**Compte Rendu TP4 C++ : Héritage, polymorphisme**

**Document de spécification et de conception**

**I\ Spécifications**

Le but de ce TP est de manipuler des figures géométriques différentes. L’utilisateur utilisera l’application à travers des lignes de commandes. Il pourra ainsi créer, modifier et supprimer des figures. Il lui sera aussi possible de sauvegarder son travail ou de le charger.

**Définitions**

**Cercle :** Figure géométrique caractérisé par son origine et son rayon

**Rectangle :** Figure géométrique caractérisé par deux points : son coin supérieur gauche et son coin inférieur droit. Un rectangle ne peut pas avoir ses arêtes qui se croisent.

**Ligne :** Figure géométrique caractérisé par deux points.

**Polyligne :** Figure géométrique caractérisé par n points : Chaque doublet de points peut être assimilé à une ligne. Une polyligne peut avoir ses arêtes qui se croisent.

**Figure :** Ensemble regroupant les quatre formes énoncées ci-dessus.

**Sélection :** Objet capable d’englober plusieurs figures.

**Container :** Espace dans lequel toutes nos figures et sélections sont créées.

**Graphics :** Ensemble regroupant les figures et les sélections

**Commande :** Chaine de caractères permettant de créer un Graphics. Elle est entrée par l’utilisateur, contient le nom du Graphics et toutes les informations nécessaires pour le positionner dans le container.

**Choix généraux**

Spécifications du cahier des charges :

Pour pouvoir gérer tous les cas de configuration possibles, il a été nécessaire d'éclaircir le cahier des charges. Ainsi, un cercle peut avoir un rayon nul mais pas négatif. Une polyligne est composée d'un point au minimum tandis qu'une ligne est toujours composée de deux points.

La sélection ne contient que les figures qui sont entièrement incluses dans celle-ci (et non pas partiellement).

Seules les commandes propres aux figures sont affichées si l’utilisateur fait appel à la méthode LIST. Les sélections ne sont pas affichées. Les commandes de chaque figure sont actualisées de manière à afficher les coordonnées actuelles de celles-ci (après un MOVE par exemple).

Si l'utilisateur effectue un UNDO après un DELETE sur un objet faisant partie d'un sélection, ce dernier est recréé mais ne fait plus partie de la sélection.

**II\ Conception**

Nous avons choisi une structure composée de 18 classes.

**La classe main:** Gère la saisie des commandes saisies par l’utilisateur et fait appel aux méthodes de la classe container en fonction de celles-ci.

**La classe Container:** Classe qui contient toutes les méthodes exigées dans le cahier des charges. Ces méthodes sont appelées depuis le main en fonction des entrées saisies par l'utilisateur.

**La classe Graphics:** Classe abstraite définissant toute forme que l'on peut implémenter dans le container (Cercle, Rectangle, Polyligne, Ligne ou Sélection). Contient les méthodes/attributs communs aux figures et aux sélections . Les classes Selection et Figure héritent de cette classe.

**La classe Figure:** Classe abstraite définissant toutes les formes géométriques (les sélections ne sont pas des figures). Elle contient les attributs/méthodes communs à toutes les figures. Les classes Circle, Rectangle, Line et Polyline sont héritées de cette classe.

**La classe Selection:** Classe permettant d'implémenter les sélections comme prévu dans le cahier des charges. Une sélection peut contenir plusieurs figures mais n'est pas persistante.

**La classe Circle:** Classe permettant d'implémenter des cercles. Un cercle est caractérisé par son origine et son rayon.

**La classe Rectangle:** Classe permettant d'implémenter des rectangles. Un rectangle est caractérisé par deux points:Son coin supérieur gauche et son coin inférieur droit.

**La classe Line:** Classe permettant d'implémenter des lignes. Une ligne est caractérisée par deux points.

**La classe Polyline:** Classe permettant d'implémenter des polylignes. Une polyligne est caractérisée par une série de points.

**La classe Command:**

**La classe ComposedCommand:**

**La classe SimpleCommand:**

**La classe DeleteCommand:**

**La classe DeleteElementCommand:**

**La classe CreateElementCommand:**

**La classe MoveCommand:**

**La classe LoadCommand:**

**La classe Tools:** Classe qui contient des méthodes pour manipuler les chaines de caractères. Nous aurions pu nous passer de cette classe mais en mettant ces méthodes dans une même classe, cela améliore la visibilité et la clarté du code.

**Structures de données et algorithmes**

**Structure de donnée**: Map ?

Comment savoir si une Figure est incluse dans une sélection ?

Afin de savoir si toute figure est contenue ou non dans une sélection, il a été décidé de mettre en place un rectangle « fictif » propre à chaque figure. Ce rectangle est définit comme étant le plus petit rectangle dans lequel la figure est entièrement incluse. C'est en comparant les coordonnées de ce rectangle pour chaque Figure avec les coordonnées de la sélection que nous pouvons déterminer les figures qui sont incluses dans la sélection.

Pourquoi séparer Selection et Figure ?

Bien que les classes Selection et Figure soient proches (elles permettent d'implémenter des objets dans le container), nous avons choisit de les séparer au lieu d'en faire une seule et unique classe. En effet, une Figure est undo-able alors qu'une sélection ne l'est pas. De plus, une sélection ne doit pas apparaitre dans la liste des objets présents dans le container si l'utilisateur appelle la commande LIST. Pour finir, les figures sont sauvegardées dans un fichier à l'appel de la commande SAVE tandis que les sélections ne le sont pas. Tous ces éléments nous ont conduit à séparer ces deux types d'objets dans deux classes différentes.

Comment sauvegarder uniquement les figures(et non pas les sélections) ?

Afin de ne sauvegarder uniquement les figures et non pas les sélections, il a été décidé de placer l'attribut commande dans la classe Graphics afin que les deux classes héritières le possèdent. La commande d'une figure est la chaine de caractères saisie par l'utilisateur alors que la commande d'une sélection est toujours vide (chaine nulle). Par conséquent, dans le fichier, nous sauvegardons les commandes de tous les Graphics. Ainsi, les commandes des sélections sont sauvegardées mais n'introduisent aucun caractère dans le fichier de sauvegarde. Seules les commandes figures sont écrites dans le fichier.

Fonctionnement UNDO/REDO ?

Afin de pouvoir effectuer des UNDO/REDO, nous avons implémenté deux liste de pointeurs de commandes : Une liste UNDO et une liste REDO. Ainsi, à chaque modification du container (par des ajouts, déplacements, suppression...), ces listes sont mises à jour de la manière suivante :

-Pour chaque nouvelle commande du type création ou suppression de Figure, suppression de Selection, ainsi que MOVE, LOAD et CLEAR, la commande exécutée vient s'ajouter à la liste de UNDO et la liste de REDO devient vide.

-Pour chaque nouvelle commande du type création de sélection, ainsi que LIST, SAVE et EXIT, les listes UNDO et REDO ne sont pas modifiées.

--->Parler du execute unexecute et du whichlist

Argumenter Héritage ?

Particularité pour la création de rectangles et de sélections

Lorsque l'utilisateur crée un rectangle ou une sélection, il définit ce nouveau Graphics en indiquant les coordonnées de deux points: Le coin supérieur gauche et le coin inférieur droit. Cependant, si l'utilisateur se trompe et inscrit d'autres coordonnées (les deux autres points par exemple), la commande n'est pas modifiée mais les coordonnées des deux points qui nous intéressent (coin supérieur gauche et inférieur droit) seront calculées automatiquement pour permettre de créer le rectangle dans de bonnes conditions.

**Diagramme UML :**

**Spécification des tests fonctionnels**

Index des tests :

0\_x=Test de création

1\_x=Test de déplacement

2\_x=Test de suppression

3\_x=Test UNDO/REDO

4\_x=Test LOAD/SAVE

5\_x=Cas limites/ cas d’erreur

**Test 0\_1 :** Création d’un cercle

On vérifie que le cercle à bien été créé avec les bons paramètres

**Test 0\_2 :** Création rectangle

On vérifie que le rectangle à bien été créé avec les bons paramètres

**Test 0\_3 :** Création ligne

On vérifie que la ligne à bien été créée avec les bons paramètres

**Test 0\_4 :** Création polyligne

On vérifie que le polyligne à bien été créé avec les bons paramètres

**Test 0\_5 :** Création sélection

On vérifie que la sélection a bien été créée et qu’elle n’apparait pas si on LIST

**Test 1\_1 :** Déplacement figure

On vérifie que les coordonnées de la figure passée en paramètre ont été incrémentées de dx et dy

**Test 1\_ 2:** Déplacement sélection

On vérifie que les coordonnées de toutes les figures présentes dans la sélection passée en paramètre ont été incrémentées de dx et dy

**Test 1\_3 :** Déplacement d’une figure qui se trouve dans une sélection

On vérifie que les coordonnées de la figure passée en paramètre ont été incrémentées de dx et dy mais que les autres figures de la sélection n’ont pas bougé

**Test 1\_4 :** Déplacement d’une sélection qui englobe une autre sélection

On vérifie que les coordonnées de toutes les figures présentes dans les deux sélections ont été incrémentées de dx et dy

**Test 1\_5 :** Déplacement d’une sélection qui ne contient pas entièrement une figure

On vérifie que les coordonnées de la figure ne figurant pas entièrement dans la sélection n’ont pas été incrémentées

**Test 1\_6 :** Déplacement d’une figure contenue dans une sélection qui vient à dépasser les limites de la sélection

On vérifie que les coordonnées de la figure sont correctes, que seule cette figure s’est déplacée et qu’elle est toujours présente dans la sélection

**Test 2\_1 :** Suppression figure

On vérifie que la figure passée en paramètre ne fait plus partie de la liste des figures présentes dans le container

**Test 2\_2 :** Suppression plusieurs figures

On vérifie que les figures passées en paramètre ne font plus partie de la liste des figures présentes dans le container

**Test 2\_3 :** Suppression sélection

On vérifie que les figures présentes dans la sélection passée en paramètre ne font plus partie de la liste des figures présentes dans le container

**Test 2\_4:** Suppression figure faisant partie d’une sélection

On vérifie que la figure passée en paramètre ne fait plus partie de la liste des figures présentes dans le container mais que la sélection existe toujours

**Test 2\_5 :** Suppression d’une sélection suivie de la suppression d’une figure qui faisait partie de cette sélection dans la même commande ?

On vérifie que toutes les figures présentes dans la sélection ont été retirées de la liste des figures présentes dans le container comme si la sélection seule avait été supprimée

**Test 3\_1 :** UNDO création figure

On vérifie que la figure créée précédemment ne fait plus partie de la liste des figures présentes dans le container

**Test 3\_2 :** UNDO déplacement figure

On vérifie que la figure a récupéré les mêmes coordonnées qu’elle possédait avant son déplacement

**Test 3\_3 :** UNDO suppression figure

On vérifie que la figure fait a nouveau partie de la liste des figures présentes dans le container

**Test 3\_4 :** UNDO CLEAR

On vérifie que toutes les figures qui ont été supprimées par le CLEAR font à nouveau partie de la liste des figures présentes dans le container

**Test 3\_5 :** UNDO LOAD

On vérifie que toutes les figures introduites dans le container par le LOAD ont disparues

**Test 3\_6 :** UNDO création sélection

On vérifie que le UNDO s’applique à la commande précédant la création de la sélection car une sélection n’est pas undo-able

**Test 3\_7 :** UNDO suppression figure qui faisait partie d’une sélection

On vérifie que la figure est bien revenue mais qu’elle ne fait plus partie de la sélection

**Test 3\_8 :** UNDO suivi d’une commande puis d’un REDO

On vérifie que l’on ne peut pas REDO

**Test 3\_9:** 21 UNDO

On vérifie que l’on ne peut pas faire le 21ème UNDO

**Test 4\_1 :** SAVE

On vérifie qu’un nouveau fichier a été créé et qu’il contient les commandes associées aux Graphics présents dans le container

**Test 4\_2 :** LOAD

On vérifie que les commandes contenues dans le fichier sont exécutées dans le container

**Test 5\_1 :** Pas le bon nombre de paramètres

On obtient un message d’erreur et un commentaire explicatif

**Test 5\_2 :** Rayon négatif

On obtient un message d’erreur et un commentaire explicatif

**Test 5\_3 :** SAVE avec nom de fichier déjà existant

On obtient un message d’erreur et un commentaire explicatif

**Test 5\_4 :** LOAD avec un nom de fichier inexistant

On obtient un message d’erreur et un commentaire explicatif

**Test 5\_5 :** Nom déjà pris

On obtient un message d’erreur et un commentaire explicatif

**Test 5\_6 :** DELETE dont un des noms n’existe pas

On vérifie qu’aucun des noms valides de la commande n’a été supprimé. On obtient un message d’erreur et un commentaire explicatif

**Test 5\_7:** Création sélection vide

On vérifie que la création d’une sélection vide ne pose aucun problème

**Test 5\_8**: Sélection qui contient un objet + MOVE Selection + UNDO + UNDO + REDO +REDO

On vérifie que l'objet est recrée mais ne fait plus partie de la sélection

**Test 5\_9** : Création Figure + Sélection qui contient l'objet + DELETE Figure + DELETE Selection + UNDO + UNDO

On vérifie que la Figure a été UNDO et a donc été recréée mais pas la séléction (n'est pas undo-able)