# 3 IF 2014/2015

# Andrea ACCARDO

# Cyril CANETE

# B3230

# Compte Rendu

# TP C++ n°4 : Héritage, polymorphisme



Application DrawController

Dessin vectoriel en ligne de commande

DESCRIPTION

*L’application DrawController permet de gérer des formes dans un repère. L’utilisateur peut insérer, retirer et déplacer ces formes. Il peur également créer des sélections afin d’appliquer une actions à un groupe de formes. Les formes gérées sont le cercle, le rectangle, les lignes et les polylignes. Une fonction d’annulation est également disponible et permet à l’utilisateur d’annuler jusqu’à 20 actions. La gestion de ces formes ce fait via l’invite de commande. Chaque commande entrée par l’utilisateur est suivie d’un message de validation ou d’erreur.*

COMMANDES UTILISATEUR

Après avoir démarrer l’application via la commande *tpheritage* depuis le shell, l’utilisateur peut interagir directement avec le programme.

AJOUT DE FORMES ET DE SELECTIONS

* **C Name X1 Y1 R** : *Crée un nouveau Cercle*
  + - * ***Name*** *: Chaine de caractère (sans espace), Identifiant du cercle*
      * ***X1*** *: Entier, Coordonnée X du centre du cercle*
      * ***Y1*** *:* *Entier, Coordonnée Y du centre du cercle*
      * ***R*** *: Entier, Rayon du cercle*
      * *RETOUR : OK si l’ajout à réussi, ERR + #Type Erreur, si une erreur est survenue.*
* **R Name X1 Y1 X2 Y2** : *Crée un nouveau Rectangle*
  + - * ***Name*** *: Chaine de caractère (sans espace), Identifiant du rectangle*
      * ***X1*** *: Entier, Coordonnée X du premier point diagonal du rectangle*
      * ***Y1*** *:* *Entier, Coordonnée Y du premier point diagonal du rectangle*
      * ***X2*** *: Entier, Coordonnée X du second point diagonal du rectangle*
      * ***Y2*** *:* *Entier, Coordonnée Y du second point diagonal du rectangle*
      * *RETOUR : OK si l’ajout à réussi, ERR + #Type Erreur, si une erreur est survenue.*
* **L Name X1 Y1 X2 Y2** : *Crée une nouvelle Ligne*
  + - * ***Name*** *: Chaine de caractère (sans espace), Identifiant de la ligne*
      * ***X1*** *: Entier, Coordonnée X de la première extrémité de la ligne*
      * ***Y1*** *:* *Entier, Coordonnée Y de la première extrémité de la ligne*
      * ***X2*** *: Entier, Coordonnée X de la seconde extrémité de la ligne*
      * ***Y2*** *:* *Entier, Coordonnée Y de la seconde extrémité de la ligne*
      * *RETOUR : OK si l’ajout à réussi, ERR + #Type Erreur, si une erreur est survenue.*
* **PL Name X1 Y1 …. Xn Yn** : *Crée une nouvelle Poly-line*

*Remarque: n, entier strictement positif*

* + - * ***Name*** *: Chaine de caractère (sans espace), Identifiant de la ligne*
      * ***Xi*** *: Entier, Coordonnée X du i-ème extrémité point de la poly-ligne*
      * ***Yi*** *:* *Entier, Coordonnée Y du nième extrémité point de la poly-ligne*
      * *RETOUR : OK si l’ajout à réussi, ERR + #Type Erreur, si une erreur est survenue.*
* **R Name X1 Y1 X2 Y2** : *Crée une nouvelle Sélection*
  + - * ***Name*** *: Chaine de caractère (sans espace), Identifiant de la sélection*
      * ***X1*** *: Entier, Coordonnée X du premier point diagonal de la sélection*
      * ***Y1*** *:* *Entier, Coordonnée Y du premier point diagonal de la sélection*
      * ***X2*** *: Entier, Coordonnée X du second point diagonal de la sélection*
      * ***Y2*** *:* *Entier, Coordonnée Y du second point diagonal de la sélection*
      * *RETOUR : OK si l’ajout à réussi, ERR + #Type Erreur, si une erreur est survenue.*

FONCTIONS DE GESTION DES FORMES ET SELECTIONS

* **DELETE Name1 Name2…NameN** : *Supprime les objets « Name1 » à « NameN »*

*Remarque: n, entier strictement positif*

* + - * ***Name(i)*** *: Chaine de caractère, Identifiant de l’objet/selection à supprimer*

*Remarque: Si Name(i) est une selection, tous les objets de la sélection sont supprimés.*

* + - * *RETOUR : OK si l’ajout à réussi, ERR + #Type Erreur, si une erreur est survenue.*
* **MOVE Name dX dY** : *Déplace de dX, dY l’objet « Name »*
  + - * ***Name :*** *Chaine de caractère, Identifiant de l’objet/selection à déplacer*

*Remarque: Si Name est une selection, tous les objets de la sélection sont déplacés.*

* + - * ***dX*** *: Entier, valeur de déplacement selon d’axe X*
      * ***dY*** *: Entier, valeur de déplacement selon d’axe Y*
      * *RETOUR : OK si l’ajout à réussi, ERR + #Type Erreur, si une erreur est survenue.*
* **LIST** : Enumère tous les objets du repère
  + - * *RETOUR : Liste des objets présents dans le repère sous la forme de la commande nécessaire à sa création, liste par ordre alphabétique des identifiants.*
* **UNDO** : Rétabli l’état du repère précédant la dernière commande

*Remarque: une sélection n’est pas annulable*

* + - * *RETOUR : OK*
* ***REDO*** : Rétabli l’état du repère suivant le dernier UNDO

*Remarque: en cas de nouvelle action de l’utilisateur, la fonction REDO est inaccessible.*

* + - * *RETOUR : OK*
* ***LOAD nomfichier.txt*** : Ajoute un ensemble d’objet contenu dans le fichier « nomfichier.txt » au repère
  + - * *RETOUR : OK*
* ***SAVE nomfichier.txt*** : Sauvegarde l’ensemble des objets dans le fichier « nomfichier.txt »

*Remarque: les sélections ne sont pas sauvegardées.*

* + - * *RETOUR : OK*
* ***CLEAR*** : Supprime l’ensemble des objets et sélections du repère
  + - * *RETOUR : OK*
* ***EXIT*** : Ferme l’application
  + - * *RETOUR : OK*

## Document de conception

### Spécification complètes des classes

Ici on peut trouver tous les tests effectués et leurs relatives adresses :

|  |  |
| --- | --- |
| 1. *Description*: Création d’un cercle | Test\_Cercle :   1. run : Syntaxe correcte ; 2. Test\_InvalideCenter/run : Valeur X et/ou Y non valides ; 3. Test\_InvalideRadius/run : Valeur R non valide ; 4. Test\_NameMissing/run : Nom manquante. |
| 1. *Description*: Création d’une ligne | Test\_Line :   1. run : Syntaxe correcte ; 2. Test\_InvalidePoints/run : Valeurs X1, Y1, X2 et/ou Y2 non valides ; 3. Test\_NameMissing/run : Nom manquante. |
| 1. *Description*: Création d’un rectangle | Test\_Rectangle:   1. run : Syntaxe correcte ; 2. Test\_InvalidePoints/run : Valeurs X1, Y1, X2 et/ou Y2 non valides ; 3. Test\_NameMissing/run : Nom manquante. |
| 1. *Description*: Création d’un poly-ligne | Test\_Polyline:   1. run : Syntaxe correcte ; 2. Test\_InvalidePoints/run : Valeurs X1, Y1, X2, Y2, … Xn et/ou Yn non valides ; 3. Test\_NameMissing/run : Nom manquante. |
| 1. *Description :* Création d’une sélection | Test\_Selection:   1. run : Syntaxe correcte ; 2. Test\_InvalidePoints/run : Valeurs X1, Y1, X2 et/ou Y2 non valides ; 3. Test\_NameMissing/run : Nom manquante. |
| 1. *Description*: Affichage des objets existants | Test\_List:  run : Affichage correcte (ignore les sélections). |
| 1. *Description*: Suppression de plusieurs objets | Test\_Delete:   1. run : Syntaxe |
| 1. *Description*: Déplacement d’un objet. | Test\_Move:   1. run : Syntaxe correcte ; 2. Test\_InvalidePoints/run : Valeurs δX et/ou δY non valides ; 3. Test\_NameMissing/run : Nom manquante. |
| 1. *Description*: Annulation de la dernière commande | Test\_Undo:  run : annule correctement toute commande sauf celles qui concernant les sélection. |
| 1. *Description*: Rétablir la dernière commande annulée | Test\_Redo:  run : rétablit correctement la dernière commande sauf si après le UNDO on a exécuté un autre commande. |
| 1. *Description*: Suppression de tous les objets | Test\_Clear:  run : suppression de tous les objets (remise à zéro). |
| 1. *Description*: Sauvegarde d’un fichier | Test\_Save:   1. run : Syntaxe correcte ; 2. Test\_NameMissing/run : Nom manquante ; 3. Test\_ExistentFile/run : Fichier déjà existant. |
| 1. *Description*: Chargement d’un fichier | Test\_Load:   1. run : Syntaxe correcte ; 2. Test\_NameMissing/run : Nom manquante ; 3. Test\_FileMissing/run : Fichier inexistant. |

### Architecture globale de l’application

Nous avons ainsi structuré le logiciel :

1. Une classe *Page* constituée par :
   1. Un constructeur et un destructeur de défaut ;
   2. Des méthodes :
      1. *AjoutReferer (const string refererPage)*, qui ajout un référer a cette Page ;
      2. *Afficher (), qui affiche une page : son nom et le nombre de clique qui elle a eu ;*
      3. *GetHit (), qui renvoie l'attribut nombreHits ;*
      4. *InsererDot (ofstream fichier), qui remplit le fichier avec des arcs valus en format graphviz.*
   3. Des attributs :
      1. Un unsigned int indiquant le nombre de hits ;
      2. Un *dictionnaire\** contenant des *unsigned int* ;
      3. Une string contenant le nom de la page.
2. Une classe *Navigation*constituée par :
   1. Un constructeur et un destructeur de défaut ;
   2. Des méthodes :
      1. AjoutPage ( string nomPage, string refererPage ), qui prend deux strings en entrée (respectivement le nom de la page et le page de
      2. AfficheTopPage (), qui affiche les top 10 pages sans prendre paramètre en entrée.
      3. CreeDot (string), qui crée un fichier dot représentant Navigation en prenant le nom du fichier en paramètre ;
      4. comptageTopPage (), méthode privé qui calcule les top 10 pages sans prendre paramètre en entrée, et les renvoie comme un tableau de strings.
   3. Des attributs :
      1. Un *dictionnaire\** contenant des *Pages*.
3. Une classe *LigneLog* constituée par :
   1. Un constructeur qui prend en paramètre une string (que doit avoir le bonne format), un constructeur et destructeur de défaut ;
   2. Des méthodes :
      1. Afficher (), qui affiche toutes les caractéristiques de la LigneLog ;
      2. TestExtension (), qui teste si la LigneLog contient des fichiers image, JavaScript ou CSS et renvoie faux si oui, vrai sinon ;
      3. TestOptions (vector<string>\*), qui prend en paramètre un vecteur et teste si la LigneLog respecte les options contenues dans le vecteur ;
      4. GetReferer ( ), qui return une string contenant le referer ;
      5. GetUrl ( ) qui return une string contenant l’Url Demande.
   3. Des attributs :
      1. Une string ipClient ;
      2. Une string userLogname ;
      3. Une string authenticatedUser ;
      4. Une Date\*\* date ;
      5. Une Requete\*\* request ;
      6. Une int returnCode ;
      7. Une string dataQuantity ;
      8. Une string referer ;
      9. Une string navigateur.
4. Une interface \*\*Data contenant :
   1. La structure Data qui est constituée par :
      1. Un unsigned int Jour;
      2. Une string Mois;
      3. Un unsigned int Annee;
      4. Un unsigned int Heure;
      5. Un unsigned int Minute;
      6. Un unsigned int Seconde;
      7. Une string Fuseau;
   2. La structure Request qui est constituée par :
      1. Une string Action;
      2. Une string UrlDemande;
      3. Une string Protocole;

### \*Structures de données

Navigation : Nous avons choisi un arbre qui stocke des pages en utilisant comme clé leur nom.

Nous avons opté pour cette clé car la recherche par nom et beaucoup plus employée (à chaque lecture de ligne) alors que le max sera calculé une seule fois.

Page : Nous avons choisi un arbre qui stocke des entières (nombre hits) en utilisant comme clé leur référer. Nous avons opté pour cette clé car, dans notre cas, c’était l’unique possible pour identifier le nombre de hits.

LigneLog : Nous avons choisi un vecteur qui stocke des strings (options). Nous avons opté pour cette structure de données car est la plus adapte lorsqu’on ne connais pas la dimension d’une liste d’objet, vu qu’elle modifie sa taille automatiquement.