PG208, Projet n°2 : Dessin vectoriel

Bertrand LE GAL*, Serge BOUTER†et Clément VUCHENER‡

Filière électronique 2^{eme} année - Année universitaire 2011-2012

1 Introduction

1.1 Objectif du projet

L'objectif du projet est de mettre en œuvre les notions appréhendées durant l'enseignement de de PG208. Durant cet enseignement vous avez normalement acquis les notions de base de la programmation orientée objets appliquées au langage C++. Le point de départ du projet est constitué du présent document. Ce dernier correspond au cahier des charges spécifié par le client (votre enseignant). Afin de mener à bien ce projet, vous allez devoir :

- lire le cahier des charges,
- identifier les besoins exprimés par le client,
- comprendre ce que vous devez développer,
- élaborer un modèle de solution,
- développer et mettre au point votre application.

Pour mener à bien les quatre premières étapes, il vous est demandé de mettre en pratique les notion d'UML vues en cours. Cela vous permettra de structurer vos idées et de les partager (avec votre binôme et votre enseignant). De plus, étant donné que le projet est un projet de conception objets, il vous est demandé dans les deux dernière étapes de mettre en œuvre les notions vues en cours (héritage, classes, polymorphisme, surcharge, etc.).

1.2 Déroulement du projet

La durée estimée du projet est d'environ 18 heures. Six séances de TP de 3 heures sont programmés dans votre emploi du temps. Toutefois, elles ne sont pas toutes encadrées : les 2 premières et les 2 dernières

*bertrand.legal(at)ims-bordeaux.fr
†serge.bouter(at)u-bordeaux1.fr
‡clement.vuchener(at)inria.fr

séances sont encadrées tandis que les 2 séances intermédiaires sont planifiées mais se dérouleront sans encadrement.

Le sujet du projet a été écrit de manière à prendre en considération les différences de niveaux entre les groupes. Pour cela le sujet du projet est écrit sous forme de cahier des charges à tiroir. Cette particularité vous permettra d'avancer à votre rythme.

1.3 Évaluation de votre travail

La notation du projet se basera sur :

- La qualité de l'analyse du cahier des charges. Cette analyse doit être réalisée à l'aide d'UML.
- L'utilisation adéquate des possibilités offertes par le langage C++ (héritage, surcharge, ...).
- Le respect des fonctionnalités spécifiées dans le cahier des charges.

L'évaluation du projet sera réalisée en deux parties :

- Lors d'une présentation orale de votre projet.
- Dans un rapport papier détaillant le développement de votre application.

Lors de la présentation orale de votre application, vous êtes en charge de présenter à votre enseignant (le client) l'application développée. Cette présentation d'une durée approximative de 10 minutes vise à démontrer au client que vous avez développé ce qu'il vous a commandé. Pour cela, vous présenterez vos choix de conception, vous expliquerez les différentes étapes par lesquelles vous êtes passées et enfin une réaliserez démonstration des fonctionnalités de l'outil (démonstration que vous aurez pris soin de préparer).

En ce qui concerne le rapport écrit, une attention particulière sera portée à la présentation des choix d'implantation que vous aurez réalisés lors de votre développement. Ces choix ainsi que votre cheminement devront être détaillés clairement à l'aide du langage UML. De plus une analyse des avantages et inconvénients du langage C++ par rapport au langage C devra être réalisée.

2 Cahier des charges

L'objectif du projet est de développer une application permettant de générer d'images (BITMAP) à l'aide d'une description vectorielle de des images. Cette application doit permettre de créer une image (fichier de type BMP) à partir d'une description des formes géométriques la composant : un cercle de diamètre 3 et de couleur rouge placé aux coordonnées (6,8). Votre application devra supporter un nombre conséquent de formes géométriques de base afin de permettre le tracé d'images complexes.

L'application à développer devra analyser un fichier textuel contenant la description des formes géométriques présentes sur l'image a générer. Les différents éléments possèdent une taille, des coordonnées,

un facteur d'échelle (pour permettre la gestion d'un zoom) ainsi qu'un coefficient de transparence. Dans un premier temps (afin de simplifier la mise en oeuvre d'une première version fonctionnelle), nous supposerons que le facteur d'échelle impact uniquement sur la taille des formes géométriques et sur leurs positions (pas sur les contours de objets).

Lors du développement de l'outil et tout particulièrement lors de la phase de modélisation des classes à mettre en œuvre vous réfléchirez aux types de données que vous allez manipuler avec pour objectif principal de développer des classes faciles d'utilisation et intuitives (la contrainte de performance étant volontairement ignorée ici). De plus vous veillerez à bien analyser les liens qui peuvent exister entre les différentes classes afin de minimiser la taille du code à écrire et à maintenir.

Votre outil sera exécuté en mode ligne de commande avec un passage des arguments lors de l'appel du programme. Afin de simplifier l'utilisation de l'outil, nous partirons sur la convention suivante :

Exemple d'utilisation

```
Programme.exe fichier_initial.vec fichier_final.bmp echelle
```

Dans cet exemple :

- Le paramètre nommé *fichier_initial.vec* permet de spécifier le fichier qui contient la description des formes géométriques contenues dans l'image.
- Le paramètre nommé *fichier_final.bmp* permet de spécifier le nom du fichier image que vous devez créer à partir du contenu du fichier *fichier_initial.vec*.
- Le paramètre nommé *échelle* permet de spécifier le niveau du zoom souhaité par l'utilisateur. En cas d'omission de ce paramètre, sa valeur par défaut sera 1.

2.1 Les formes géométriques de base

L'objectif du projet est de permettre la génération d'images plus ou moins complexes à partir d'une description des formes géométriques qui les composent. Pour cela, il est nécessaire de gérer au minimum les formes géométriques suivantes :

- **Le Point** : le point sera la forme géométrique la plus simple. En effet, la définition d'un point pourrait se résulmer à paire de coordonnées (X, Y). La déclaration d'un objet point dans le fichier d'entrée sera faite de la manière suivante *[POINT : X, Y, COULEUR, TRANSPARENCE ;]*
- **La Ligne** : la ligne est un élément géométrique possédant 2 couples de coordonnées permettant de spécifier son point de départ et son point d'arrivé (X_1, Y_1) et (X_2, Y_2). La déclaration d'une ligne dans le fichier d'entrée sera faite de la manière suivante : *[LIGNE : X_1, Y_1, X_2, Y_2, COULEUR, TRANSPARENCE ;]*
- **Le Rectangle** : le rectangle est une forme géométrique composée normalement de 4 coordonnées représentant chacun de ses sommets. Pour une plus grande facilité de programmation, nous définirons un Rectangle à l'aide de sa coordonnée inférieure (la plus petite), sa longueur et sa hauteur. La déclaration d'un rectangle dans le fichier d'entrée sera faite de la manière suivante : *[RECTANGLE :*

FIGURE 1 – Résultat de l'affichage d'un Disque et d'un DisqueS (disque plein)

X, Y, LONGUEUR, HAUTEUR, COULEUR, TRANSPARENCE ;]

- **Le Carré** : le carré est une forme géométrique composée normalement de 4 coordonnées représentant chacun de ses sommets. Pour une plus grande facilité de programmation, nous définirons un carré à l'aide de sa coordonnée inférieure (la plus petite), sa longueur et sa hauteur. La déclaration d'un carré dans le fichier d'entrée sera faite de la manière suivante : *[CARRE : X, Y, LONGUEUR, HAUTEUR, COULEUR, TRANSPARENCE ;]*
- **Le Cercle** : le disque est une forme géométrique circulaire composée de la coordonnée de son centre ainsi que de la taille de son rayon. La déclaration d'un cercle dans le fichier d'entrée sera faite de la manière suivante : *[CERCLE : X, Y, RAYON, COULEUR, TRANSPARENCE ;]*

Les 3 dernières formes géométriques (rectangle, carré et cercle) posséderont des variantes. Ces variantes nommées respectivement (RectangleS, CarreS et CercleS) permettront de tracer le contour de la forme ainsi que de colorer l'intérieur de la forme. L'exemple présenté dans la Figure 1 illustre la différence entre un Cercle et un CercleS.

Pour l'ensemble des formes géométriques, vous devrez considérer le facteur d'échelle, spécifié à l'appel du programme, point qui sera expliqué plus en détails dans la section suivante. Chaque forme géométrique supportera l'attribution d'une couleur. La couleur sera spécifiée à l'aide de mots clefs représentant un assemblage des composantes rouge, vert et bleu.

De plus afin d'autoriser la création de figures assez complexe vous prendrez en compte un dernier paramètre pouvant être facultatif dans la déclaration des formes géométriques et qui représentera le niveau (plan Z) ou se situe la forme par rapport aux autres. Les formes géométriques ne possédant pas cette information seront considérées comme appartenant à l'arrière plan (z=0). L'information liée aux plans permettra de connaître l'ordre optimal de tracé des formes lors de la génération du rendu final.

2.2 Le facteur d'échelle

Le facteur d'échelle que l'on souhaite employer lors de la génération des images à partir de leur description vectorielle va permettre de choisir la taille de l'image résultante. Le coefficient va jouer le rôle de facteur de grossissement de l'image.

La Figure 2 présente un exemple d'une image composée d'un rectangle et d'un disque. L'image de gauche a été obtenues avec un facteur d'échelle fixé à la valeur 1 tandis que celle de droite a été obtenue

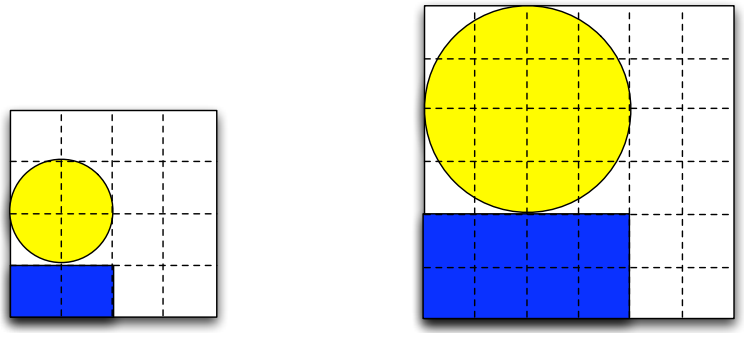

FIGURE 2 – Exemples d'une même figure obtenue avec un facteur d'échelle 1 et 2

avec un facteur d'échelle fixé à 2.

La première différence entre ces 2 tracés est liée à la taille de l'image générée à partir d'une description vectorielle. Cette dernière est multipliée par 2 pour l'image de droite. En effet, la taille de l'image dépend directement des coordonnées maximales des objets contenus dans l'image à tracer.

2.3 La gestion de la transparence

Dans les précédentes parties du sujet, il est fait mention de transparence. En effet, afin de permettre la génération d'images complexe, il est nécessaire de gérer l'ordre de superposition des formes géométrique ainsi que la transparence possible des certaines formes.

Dans le cadre du projet, nous nous limiterons a une gestion basique de la transparence des formes géométriques. Pour cela, a chaque forme géométrique possèdera un facteur de transparence. Cette information spécifiée dans le fichier décrivant les formes géométriques utilisées dans l'image, sera une valeur entière comprise entre 0 et 100. Par convention, la valeur 0 signifiera que l'objet est invisible tandis que la valeur 100 signifiera qu'il est opaque.

Le facteur de transparence impactera uniquement lors du tracé et du remplissage des formes géométriques dans votre image. La valeur du pixel résultant du tracé d'une forme géométrique possédant un facteur de transparence est fourni dans l'équation suivante :

$$Pixel'(x,y) = \frac{(100 - transp) \times Pixel(x,y) + transp \times CouleurForme}{100}$$

3 Objets fournis par votre enseignant

L'objectif de votre projet est de réaliser une gestion des formes géométriques déclarées dans le fichier d'entrée et de les afficher (dessiner) dans une image. Afin de simplifier votre développement votre enseignant met à votre disposition un ensemble de 4 classes écrites en C++ : CBitmap, CImage, CLigne et

CPixel.

L'ensemble de ces 4 classes vous permettra de créer une image vierge (toute blanche) en fonction des dimensions que vous préciserez à la création d'une instance de cet objet. Cette classe vous permettra aussi de lire et de modifier les pixels contenus dans l'image et finalement de mémoriser le résultat dans un fichier sur le disque dur. Voici les fonctions réalisées par les différentes classes fournies : CBitmap, cette classe permet de charger ou sauvegarder à partir ou depuis le disque dur une image au format BMP. Cette classe nécessite afin de travailler une image au format CImage.

CImage, cette classe modélise une image en couleurs dont la longueur et la largeur sont spécifiées à la construction de l'objet. Cette classe fournie un certain nombre de méthodes permettant d'accéder aux CPixel composants l'image. CLigne, cette classe est utilisée dans la classe CImage afin de réaliser la gestion de l'image sur les 2 dimensions (une image est une colonne composée de lignes). Cette classe peut être utilisée dans certains cas afin d'accélérer l'accès aux objets de type CPixel. CPixel, cette classe est responsable de la modélisation d'un point de l'image. Cette classe stocke les informations relatives à la couleur d'un point de l'image.

4 Extensions envisageables

Pour ceux qui trouveraient le sujet du projet trop court, il est possible de rajouter à ce dernier une dimension graphique. Cette partie ne doit être considérée que si vous avez terminé proprement la première partie (développement de l'application avec une interface de type terminal).

Afin d'améliorer l'ergonomie de l'outil, on peut souhaiter lui adjoindre une interface graphique. Afin de développer de telles interfaces en C++, il existe la bibliothèque QT de Trolltech qui propose un certain nombre d'objets graphiques facilement utilisables. La bibliothèque QT est multi-plateforme et libre de droits pour des projets personnels. A l'heure actuelle elle doit être installée par défaut sur les environnement Linux de l'ENSEIRB.

Vous trouverez des tutoriaux pouvant vous aider dans le développement d'une interface graphique pour l'outil aux adresses suivantes :

`http://www.digitalfanatics.org/projects/qt_tutorial/index.html`

`http://www.digitalfanatics.org/projects/qt_tutorial/chapter05.html`

`http://www.digitalfanatics.org/projects/qt_tutorial/chapter06.html`

`http://www.digitalfanatics.org/projects/qt_tutorial/chapter07.html`