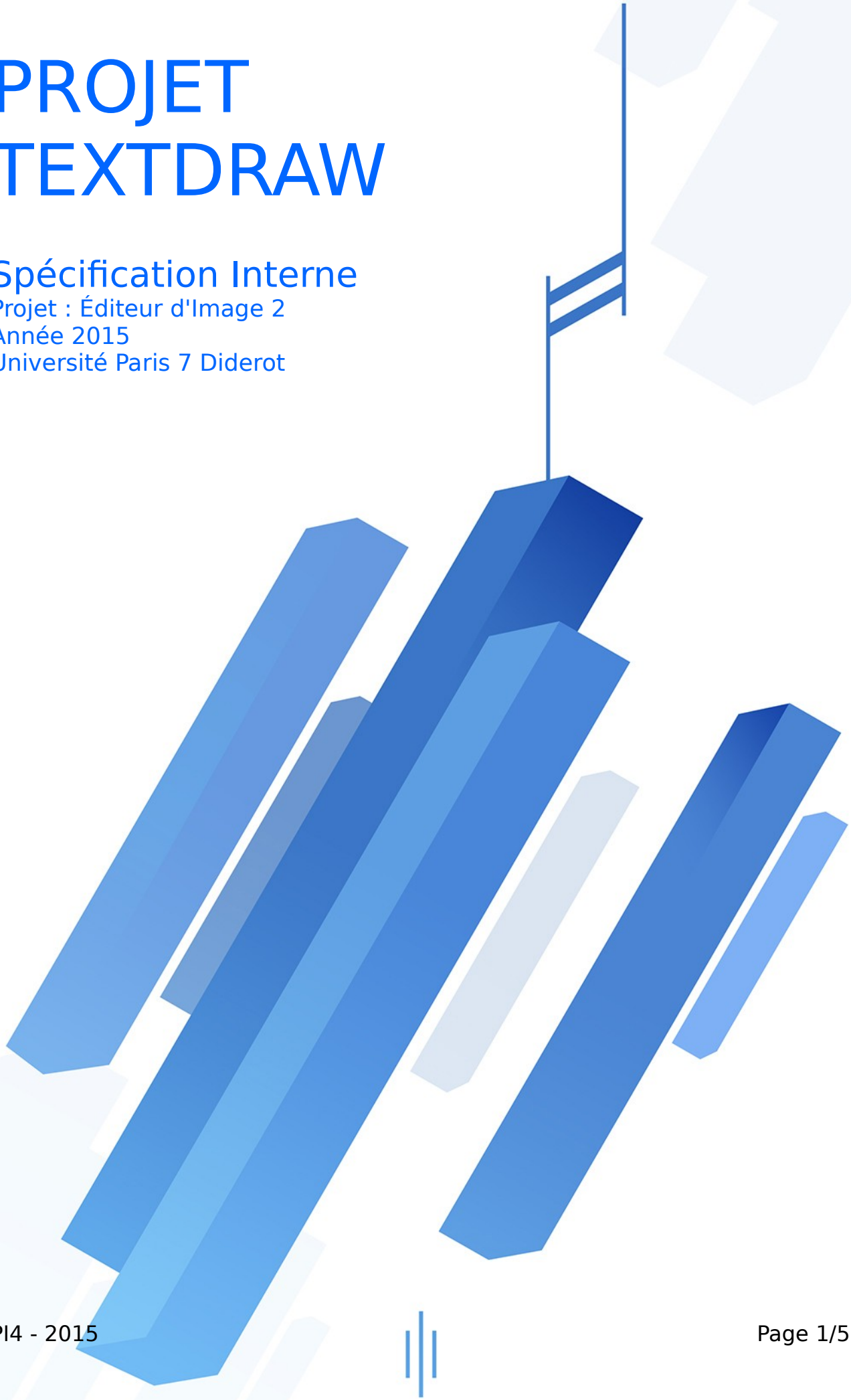
PROJET TEXTDRAW

Spécification Interne

Projet : Éditeur d'Image 2
Année 2015
Université Paris 7 Diderot



1 Présentation des principales classes :

- Main.java : permet de lire le fichier de commande et de créer l'image de sortie
- Lexer.java : permet de créer le flot de jeton qui compose la grammaire
- Parser.java : permet de reconnaître les instructions que l'utilisateur souhaite et savoir si il y a une erreur ou pas
- FormePrimitive : permet de créer les formes que l'on veut dessiner et gére toutes la parties PPM, création d'un tableau de Pixel pour chaque forme.
- CourbeDeBezier.java : permet de stocker des courbes
- Transformation.java : permet de faire les transformations sur une forme donnée

2 Entrée

Notre programme aura pour but de transformer un fichier texte (.txt) en image vectorielle (.svg) ou en image matricielle (.ppm). Notre but est de traduire les instructions en objet primitif, ce qui nous permettra de faire le moins de travail possible pour les convertir en image vectorielle ou matricielle.

Au lieu de faire toute la traduction des instructions de base jusqu'à nos 2 types d'images, nous factorisons le maximum du travail. Nos objets primitifs seront de simple classe java contenant des listes de courbe de Bézier (représenté par 2 points et leurs 2 vecteurs tangent).

Les instructions de base seront tout d'abord séparer dans une file, puis lu une par une, et mot par mot. Les commandes permettront de soit, définir une variable stocker dans une Hashmap, soit créer un objet primitif stocké dans une liste.

Au départ on lit le fichier de commande, au passage on initialise un lexer qui crée un flot de jeton. Par la suite on va passer le flot de jeton dans le parser où l'on va si nécessaire faire des transformations pour modifier directement les coordonnées dans la liste de formePrimitive.

3 Plat

Une fois la liste de commandes complètement parcourus, nous nous retrouverons avec une liste d'objets primitifs. Ces objets posséderont 2 fonctions principales : 'toSVG()' et 'toPPM()'.

Quand il s'agit d'une image au format SVG (format d'image vectorielle) on va créer une chaîne de caractère contenu du texte en XML .Puis pour chaque courbe de chaque forme on va insérer les balises du langage permettant de dessiner les formes avec la balise courbe de Bézier.

Quand il s'agit d'une image au format PPM (format d'image matricielle), il va falloir calculer chaque point que l'on doit dessiner. Je décompose chaque forme en courbe puis chaque courbe en une série de point en utilisant la formule mathématique d'une courbe de bézier :

$$\mathbf{P}(t) = \mathbf{P}_0(1-t)^3 + 3\mathbf{P}_1 t(1-t)^2 + 3\mathbf{P}_2 t^2(1-t) + \mathbf{P}_3 t^3$$

Je dessine chaque pixel que je rencontre.
Pour le coloriage d'une forme le principe est le suivant, colorier le tableau subsidiaire de la couleur du remplissage de la forme, dessiner le contour de la forme puis remettre les attributs de couleur pour chaque pixel en dehors du contour à une valeur -1. Ceci me permet par la suite de ne pas recopier le surplus de l'image de la forme créer dans l'image final.

4 Dessert

C'est le moyen de créer l'image deux possibilités : une image SVG ou image PPM :

L'image voulue doit être au format SVG, j'ai créer une chaîne de caractère final avec tous le texte qu'il faut pour créer une image SVG, je met la chaîne de caractère dans le fichier que j'ai créer.

L'image voulue doit être au format PPM, je commence par créer une image quand j'ai ma première ligne de pixel de l'image puis à chaque nouvelle ligne du tableau je l'ajoute à dans le fichier, cette méthode de le faire pas à pas et de ne pas manipuler des chaînes de caractères énormes ce qui me permet d'avoir une optimisation en temps dans la création de l'image (essentiellement sur les grandes images).