École nationale supérieure d'informatique et de mathématiques appliquées de Grenoble

BezierToSTL

Robin VINCENT-DELEUZE & Floran NARENJI-SHESHKALANI

20 janvier 2016

1 Prélude

Les références sont situées en fin de `README.md`. Nous utilisons le système de gestion de version **git** sur la plateforme **github** afin de maintenir l'historique de nos modifications et de synchroniser le code entre les différents lieux et membres de l'équipe.

L'application se configure à l'aide des différents constantes présentes dans la fonction principale (fichier `src/beziertostl.adb`). L'exécutable est `bin/beziertostl`.

Des tests unitaires (couverture partielle) sont présents dans `src/test_unitaires.adb` et sont exécutables via `bin/test_unitaires`.

2 Fonctionnement

2.1 Machine séquentielle à mots

Le package `Iterateur_Mots` est une machine séquentielle pour la lecture mot à mot (séparateur paramétrable) d'une `string`. Il permet aussi un `look ahead` sur le mot suivant.

2.2 Parser SVG

2.2.1 Types & Constantes

Deux constantes définissent les séparateur : mot (espace) & coordonnées (virgule), décimal (point). Le type `Op_Code` énumère les instructions supportées (MLHCQVmlhcqv). Pour préserver la sensibilité à la casse, le type `character` est utilisé pour les valeurs de

l'énumération. On définit deux sous types, `Op_Code_Absolute` (majuscules) et `Op_Code_Relative` (minuscules).

2.2.2 Algorithme

Le parser_svg ouvre le fichier et cherche la ligne avec le marqueur (constante du package). La lignée est nettoyée à l'aide d'un `trim` (`Ada.Strings`). `Iterateur_Mot` est utilisé.

L'analyse de la courbe est un cycle :

1. Lecture d'un opcode
2. Selon l'opcode, lecture de N arguments (coordonnée seule ou pair)
3. Ajout de la courbe à la liste (cf. 2.3)
4. Si il y a encore des arguments, aller à l'étape 2
5. Si il y a encore des mots, aller à l'étape 1
6. Renvoie la liste de courbes décrivant la figure

On supprime le positionnement relatif en y ajoutant la position courante.

2.3 Courbes

On souhaite découpler la gestion des courbes de celle du SVG. On choisit une représentation intermédiaire en arbre (héritage) dont la classe de base `Courbe` est abstraite. Les différentes courbes supportées (incluant la droite et le point seul nommé singleton) en héritent. Le mécanisme du polymorphisme permet au système de fonctionner avec une liste de `Courbe`. Cette classe `Courbe` fournit deux contrats :

— `Obtenir_Point` renvoyant $(x, f(x))$ avec f étant la fonction de la courbe.

— `Accepter` du patron de conception visiteur, permettant d'accepter une visite (cf. 2.4).

Cette représentation tente d'approcher le principe SOLID.

Brièvement, une classe en Ada est un `tagged record`. Un `record` héritant est instancié avec `new`. Le polymorphisme se fait à travers une instanciation d'une variable du type donné par l'attribut `'Class` d'un `tagged record`, au quel on assigne l'instance de l'objet sur lequel on souhaite bénéficier du polymorphisme (`redispatching`).

2.4 Interpolation linéaire

L'interpolation linéaire fournit un ensemble de segments approchant une courbe. On choisit le patron de conception visiteur car l'interpolation :

— Est un comportement que l'on peut ajouter à une courbe mais ne relève pas de la responsabiltié de la courbe.

— Se comporte différement selon le type de courbe et le type d'interpolation souhaitée (`double dispatch`).

Réalisée dans le package `Interpolation_Lineaire`, l'interpolation linéaire d'une courbe consiste à la diviser en N points équidistants l'axe des abscisses et à calculer la valeur de la courbe aux points donnés.

2.4.1 Algorithme de De Casteljau

L'algorithme de De Casteljau est une approche récursive de type diviser pour mieux reigner au calcul des courbes de Bézier :

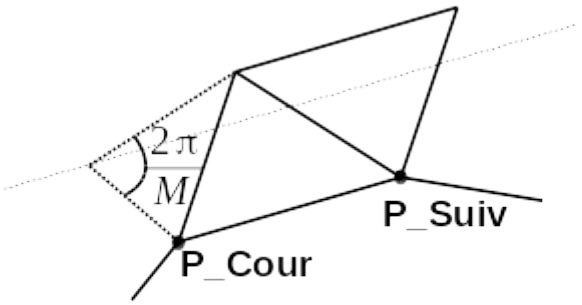
— La courbe est-elle suffisamment plate ?
— Si oui, interpolation de la courbe comme une droite
— Sinon, division de la courbe en deux courbes de Bézier et applications de l'algorithme aux deux courbes.

La tolérance est le seul paramètre de l'algorithme : elle définit quand un courbe de bézier peut être considérée comme étant droite.

Nous ne proposons cette que pour les courbes de Bézier. C'est une implémentation de la méthode proposée dans *Piecewise Linear Approximation of Bézier Curves*, par Kasper FISCHER (lien dans le README).

2.5 Génération et Sauvegarde du STL

On dispose de la valeur `M`, le pas de la rotation, calculée à partir du nombre de facette souhaité. L'idée est de parcourir `Segments` par couples : on construit un "cercle" de facettes pour chaque paire de points 2D. Ainsi, pour les points consécutifs `P_Suiv` et `P_Cour`, on crée deux facettes conformément à l'image suivante :



et on effectue une rotation de $\frac{2\pi}{M}$ pour chaque nouveau point. La figure est ensuite exportée à l'aide de la procédure `Sauvegarde` vers un fichier STL. Le fichier sera crée ou tronqué si nécessaire.

2.6 Cas d'erreurs

3 cas d'erreurs sont relevés :

— Le fichier source n'a pas pu être ouvert
— Le fichier source est mal formé (suivi d'une indication)
— Le fichier source ne contenait pas de courbe

Les erreurs sont remontées à travers des exceptions.