Rapport du projet d’algorithmique

I Paquetage Parser_SVG

Plusieurs implémentations ont été envisagées pour le parser. La première méthode possible était d’effectuer un Get_Line de data puis de parser la string. Le problème de cette méthode est qu’aucun contrôle n’est effectué sur la taille de la string ce qui pourrait créer des problèmes de sécurité/stabilité évidents.

Nous avons choisi de lire le fichier nombre flottant par nombre flottant et donc traiter les données en même temps que leur lecture ce qui diminue la quantité de mémoire utilisée et accélère l'exécution du programme. Le problème rencontré était de détecter si le prochain caractère sera une lettre ou bien le premier chiffre d’un nombre flottant, le paquetage Text_IO ne permettant pas de revenir en arrière dans la lecture (le curseur ne peut reculer) nous avons choisi de lire parfois un Float et parfois une suite de caractères destinée à être convertie en Float pour résoudre ce problème.

Le paquetage Stream_IO nous aurait permis une implémentation plus simple, mais il n’a malheureusement était découvert qu’après la création du parser. Notre solution reste tout de même plus rapide et moins coûteuse.

Interprétation des attentes:

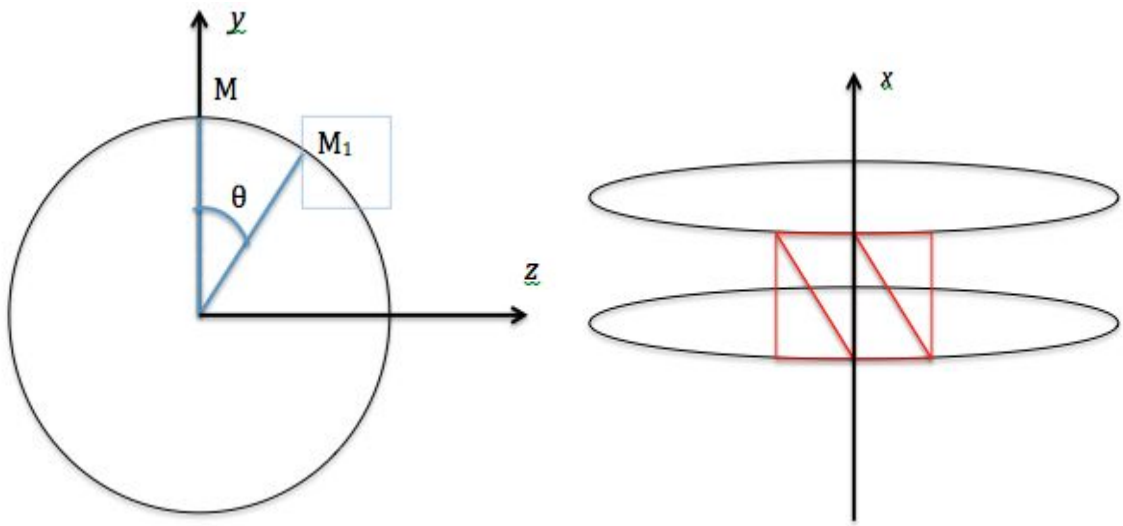
Le parser doit prendre en compte la lettre M permettant de lever le stylo. Cette lettre permet donc de créer de la discontinuité dans un path SVG si utilisée plusieurs fois dans le même d=. Il semble que l’énoncé suppose qu’un seul M ne peut être dans le path et qu’il se trouve au début de d=.

Nous avons cependant choisis de prendre en compte cette possibilité de discontinuité en rajoutant un header, footer STL et en appelant Création et Sauvegarder pendant la lecture du SVG. La liste renvoyée par parser_svg est donc la liste des points se trouvant après le dernier M lu par le parser (ce qui permet de ne pas modifier le Main fourni et que le programme se comporte comme semble le souhaiter l’énoncer s’il n’y a qu’un seul M).

II.Paquetage STL

Le codage de la partie STL a été difficile car elle nécessitait la mise en place de connaissances mathématiques et informatiques. Alors que le pré-traitement des points était relativement aisé, la création des facettes nous a pris un peu plus de temps que prévu.
Nous avons rencontré plusieurs problèmes au cours du codage de ce package, notamment une mauvaise orientation de nos facettes dû à une confusion entre les différentes coordonnées et une représentation erronée de la figure dans l'espace.

Afin de corriger ceci, nous avons procédé à plusieurs tests assez simplistes, afin de mieux visualiser l'influence de chaque paramètre sur le fichier STL. Par exemple, nous avons visualisé la mise en 3D d'un simple segment par notre procédure, ce qui nous a permis de mieux saisir le problème rencontré.



Schémas représentant la création de facettes

III Comportement séquentiel général

Afin de plus facilement rentrer dans le code d'un projet, il est intéressant de pouvoir lire ce que fait le programme dans l'ordre.

1: Appelle à Header pour créer le header du fichier STL.

2: Appelle à Chargement_Bezier pour parser le fichier SVG.

Dans Chargement_Bezier:

1: On lit tous les caractères blancs, si le premier caractère non-vide n'est pas un 'd' on saute de ligne et on recommence.

*** On a trouvé la bonne ligne ***

2: On lit un caractère et on regarde s'il s'agit d'une lettre. Si oui on modifie la variable permettant d'indiquer comment les points doivent être traités (Ligne, courbe de bézier…). Sinon, on vient de lire un chiffre, et c'est là que revenir en arrière aurait été intéressant pour pouvoir lire un float, on va cependant continuer la lecture caractère par caractère afin de finir de lire notre nombre flottant puis convertir la string en Float.

3: En fonction de la valeur de notre variable nous indiquant comment traiter les points on va soit appeler notre chargeur de points (cas simple: L,V,H) soit continuer la lecture et décoder une courbe de bezier.

4: Dans le cas de la courbe de bezier on continue la lecture selon une méthode précise par exemple 5 nombre de plus (un déjà lu) pour Bezier cubique et seulement trois de plus pour Bezier quadratique.

*** Traitement d'une courbe de Bezier ***

4bis: On transforme notre courbe de Bezier en une suite de points avec la fonction appropriée.

5: Si jamais un nouveau M est rencontré, on est dans un cas de discontinuité et il faut donc marquer une séparation dans notre liste (qui est parcourue par couple sous la forme P1-P2 puis P2-P3 puis P3-P4) qui est traité de manière continue. On appelle donc Création et Sauvegarder pour créer le fichier STL avec la liste déjà lu puis on vide la liste de points et on recommence la lecture avec une nouvelle liste (ce qui permet donc de créer de la discontinuité).

*** Fin de lecture du fichier SVG ***

6: La liste renvoyée par Chargement_Bezier est la dernière liste de points rencontrée après la dernière occurrence d'un M.

3: On appelle création avec la liste de points.

*** Création ***

4: On commence par translater tous nos points après avoir trouvé le Xmin et le Ymin.

5: On créé toutes nos facettes en pensant à inverser les coordonnées afin de bien tourner autour de Y.

6: On appelle enfin sauvegarder et Footer. Fin du programme.