Compte-rendu du visualiseur 3D

Hadrien Titeux - Thomas Peutipaynisse

I. Implémentation

I.1. Structure du programme

Bien évidemment, nous avons suivit le plan déjà tracé par les fichiers donnés dans le sujet. Quelques modifications ont été faites (des ajouts en particulier) dans scene.ads (rajout de paramètres pour optimiser les appels de frame.adb vers une fonction de scene.adb).

Dans stl.adb

La première fonction `Nombre_Facettes` a nécessité la création de 2 fonctions de traitement de chaînes, ainsi que l'utilisation d'une bibliothèque qui simplifie légèrement l'utilisation des `unbounded_strings` , nommée ustrings. Elle vérifie simplement la présence des chaînes "endloop" à chaque ligne, comme suggéré. Une optimisation a consisté à enlever les espaces au début de chaque ligne pour accélérer la recherche.
A propos de la fonction "`dans_chaine`", il aurait été possible de la substituer par une regexp en Ada, qui semble-t-il existent d'après Internet. Nous avons préféré recoder une petite fonction simple pour apprendre à mieux manipuler les chaînes.

La deuxième fonction, `chargement_ASCII`, un peu plus complexe, utilise elle-même une fonction qui va "charger" les coordonnées du vecteur dans des float, stockés ensuite dans des facettes.
Ces deux fonctions utilisent des machines à états très simples, qui permettent de trouver les découpages à faire dans la chaîne pour extraire les valeurs, ou savoir entre quelles balises du fichier STL on se trouve.

Dans scene.adb

Rien de tout à fait nouveau dans ce fichier qui laissait assez peu de place à l'improvisation, hormis la fonction "`Projection_Facettes`", un peu modifiée pour optimiser les calculs en évitant une projection si un vecteur a sa coordonnée z derrière la caméra.

Dans frame.adb

Fonction qui nous a demandé de créer un type `Vecteur_Entier`, pour stocker les valeurs des coordonnées des projections avant de les passer à la fonction de traçage.
Une petite amélioration qui empêche les plantages quand la figure déborde du cadre a consisté à ne pas tracer les arêtes reliées aux points dont la projection est en dehors du plan.

I.2. Optimisations et améliorations possibles

I.2.a. Optimisation des calculs

Afin d'alléger les calculs, on pourrait enregistrer le maillage courant et incrémenter a chaque rotation. Cela éviterait le calcul de la matrice de rotation à chaque déplacement.

I.2.b. Lecture des fichier binaires STL

Le programme ne lit, étant donné les prérequis du projet, que des fichiers STL ASCII. Nous ne connaissons pas la structure d'un fichier binaire, donc il nous était impossible de le parser. Néanmoins, en interfaçant visualiseur.adb avec un script en ruby trouvé sur thingiverse.com, il serait possible de les convertir pour les lire (ce que nous avons fait à la main, pour tester les solides plus complexes fournis sur le site).

I.2.c Amélioration de l'affichage

Pour l'instant, nous avons décidé que les arêtes des points hors du cadre ne seraient pas tracées. Ce problème est géré par le fichier frame.adb. On pourrait imaginer de modifier ligne.adb qui pourra alors décider de ne pas tracer les pixels hors du cadre.

II. Validation

Nous avons vérifiés la fonctionnalité du programme avec différents fichiers `.stl` trouvés sur internet via le site : http://www.thingiverse.com. Le programme fonctionne correctement, le chargement du fichier est s'effectue normalement. Néanmoins, certains fichiers sont composés de beaucoup de facettes, le temps de calculs de chaque projection est long.