Projet FIG

Binôme : Antoine THIEL & Lucas MORAGUES

Outils choisis

- QCustomPlot pour l'affichage 2D (lien vers le site)
- OpenGL pour l'affichage 3D
- Qt Creator pour l'IDE
- C++ pour le langage de programmation

Les difficultés

Le démarrage a été rapide, grâce à l'aide de la librairie qcustomplot. Elle nous permet de créer un graphique 2D facilement. L'algorithme de bresenham est relativement simple à implémenter. La plus grande difficulté est de faire une fonction capable de gérer les différents octants / cadrans sans faire 300 lignes de code.
Concernant la partie 3D, OpenGL a été utilisé. Le projet nous a permis de créer une version "basique" utilisant un maximum les objets CPP. Chaque objet (point, repère, cube etc) possède son VBO. Faire tout fonctionner de manière automatique dans les opérations openGL a prit beaucoup de temps. Pas de difficulté importante rencontrée pour les opérations de matrice, bien qu'il y est eu quelques difficultés lors de la rotation. Il ne faut pas oublier aussi de convertir les angles en radian pour certaines opérations.
A noter que nous utilisons des matrices 4x4 pour le stockage mais nous n'utilisons que nos fonctions, et non celle de la classe Qt pour faire les opérations.
La partie des quaternions a été la partie la plus compliquée, notamment pour la compréhension, et l'implémentation, qui est loin d'être trivial. Les opérations sur la matrice principale n'ont pas été faciles. Trouver l'ordre pour les interpolations de quaternions a demandé beaucoup de tests.

L'espace projectif

OpenGl va utiliser une matrice dans les shaders pour calculer la position de chaque point. C'est cette matrice que l'on va utiliser pour faire les rotations et les translations. Cette matrice de transformation sera ensuite automatiquement appliquée à tous les points de l'affichage. Les opérations se passent dans le paintGL, dans glarea.cpp. Les points sont placés comme dans un repère fixe, puis en fonction de la souris et des touches, des valeurs d'angles et de positions vont être changées. À chaque rafraîchissement de l'écran, les points vont subir des rotations et des translations.
Toutes les fonctions permettant le calcul des matrices se trouvent dans espaceprojectif.cpp.

Les quaternions

Pour faire le changement de position et de rotation de caméra entre deux positions choisies, nous avons utilisé deux matrices de rotations. Premièrement, nous choisissons la position de départ. Le programme va enregistrer l'angle actuel et la position actuelle. Juste après nous bougeons la caméra pour la nouvelle position. En appuyant sur le bouton "quaternions", la deuxième position est enregistrée (avec la rotation). Un premier quaternion est créé avec la matrice de rotation, créée elle-même avec les valeurs de la première rotation enregistrée. Le deuxième quaternion est créé comme la première, mais avec les valeurs d'angle de la deuxième position.
Une fois les deux quaternions créés, il faut utiliser la méthode "slerp" qui va interpoler selon un indice t compris entre 0.0 et 1.0. Cela va rendre un 3ème quaternion, que l'on transforme en angle et trois coefficients pour les trois axes X, Y et Z. Il suffit après d'appliquer ces valeurs avec les fonctions de espaceProjectif.cpp.

Les rotations ne suivront pas le chemin que l'utilisateur a choisi (s'il a fait des vagues avec la caméra par exemple), car l'interpolation des quaternions donnera "le chemin le plus cours" sur la sphère de rotation.
La translation se fait avec un indice t similaire à celui de la rotation (le même en fait), et utilisera la fonction translation de espaceProjectif.cpp. Le programme fera une translation de (nouveauPosition - anciennePosition)*t. Ce qui permet de bouger sur un segment, en même temps que la rotation.

Sources et algorithmes

Bresenham : https://fr.wikipedia.org/wiki/Algorithme_de_trac%C3%A9_de_segment_de_Bresenham

Espace projectif : https://fr.wikipedia.org/wiki/Coordonn%C3%A9es_homog%C3%A8nes

Matrice 4x4 : http://www.opengl-tutorial.org/assets/faq_quaternions/index.html
http://www.opengl-tutorial.org/fr/beginners-tutorials/tutorial-3-matrices/

Quaternion : - https://fr.wikipedia.org/wiki/Quaternions_et_rotation_dans_l%27espace
https://www.euclideanspace.com/maths/algebra/realNormedAlgebra/quaternions/slerp/index.htm