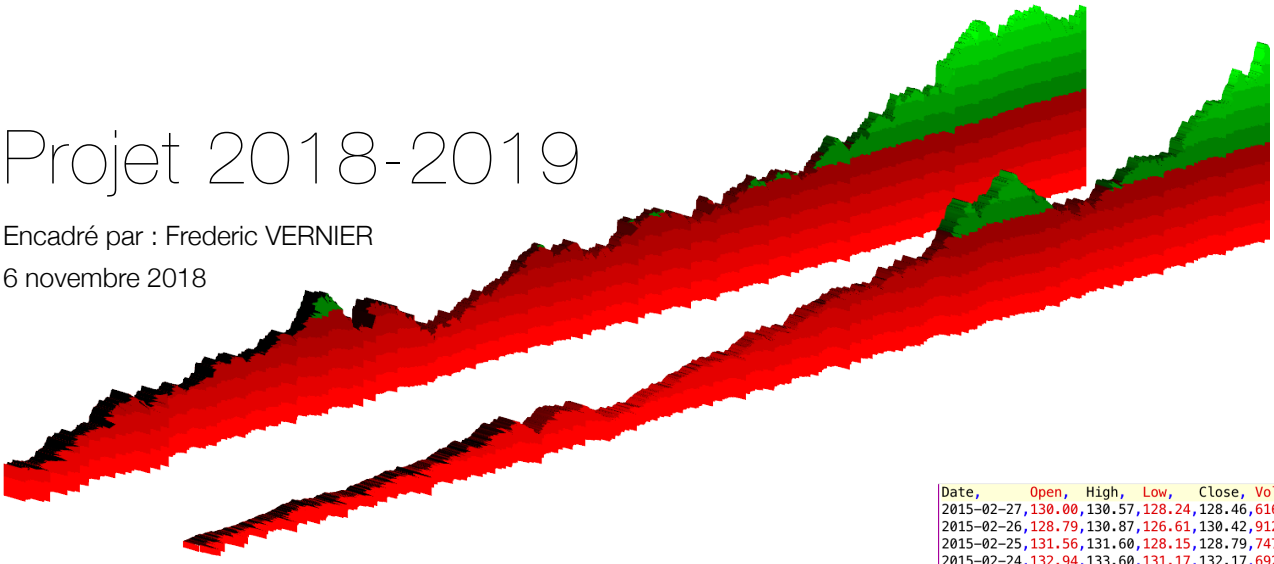
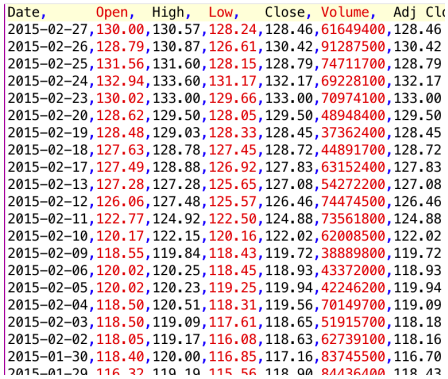
LICENCE D'INFORMATIQUE : PROJET IGSD (INFO 226 L2)



Projet 2018-2019

Encadré par : Frederic VERNIER

6 novembre 2018



Date	Open	High	Low	Close	Volume	Adj Cl
2015-02-27	130.00	130.57	128.24	128.46	61649400	128.46
2015-02-26	128.79	130.87	126.61	130.42	91287500	130.42
2015-02-25	131.56	131.60	128.15	128.79	74711700	128.79
2015-02-24	132.94	133.60	131.17	132.17	69228100	132.17
2015-02-23	130.02	133.00	129.66	133.00	70974100	133.00
2015-02-20	128.62	129.50	128.05	129.50	48948400	129.50
2015-02-19	128.48	129.03	128.33	128.45	37362400	128.45
2015-02-18	127.63	128.78	127.45	128.72	44891700	128.72
2015-02-17	127.49	128.88	126.92	127.83	63152400	127.83
2015-02-13	127.28	127.28	125.65	127.08	54272200	127.08
2015-02-12	126.06	127.48	125.57	126.46	74474500	126.46
2015-02-11	122.77	124.92	122.50	124.88	73561800	124.88
2015-02-10	120.17	122.15	120.16	122.02	62008500	122.02
2015-02-09	118.55	119.84	118.43	119.72	38889800	119.72
2015-02-06	120.02	120.25	118.45	118.93	43372000	118.93
2015-02-05	120.02	120.23	119.25	119.94	42246200	119.94
2015-02-04	118.50	120.51	118.31	119.56	70149700	119.09
2015-02-03	118.50	119.09	117.61	118.65	51915700	118.18
2015-02-02	118.05	119.17	116.08	118.63	62739100	118.16
2015-01-30	118.40	120.00	116.85	117.16	83745500	116.70
2015-01-29	116.32	119.19	115.56	118.90	84436400	118.43

Objectif

L'objectif de ce projet est de piloter un affichage OpenGL depuis un programme C++ qui affiche des courbes en trois dimensions. Vous disposerez de quatre fichiers de données (AAPL.cvs, AMZN.csv, GOOGL.csv, MSFT.csv) qui représentent la valeur en bourse de ces sociétés ainsi que les volumes d'échange d'action jour par jour. La description du projet est divisée en 4 parties (Données, Modèles 3D, Couleurs et ombrages et Interaction). Pour chaque partie nous décrivons le minimum de travail pour obtenir une note entre 8/20 et 12/20 (selon la clarté du code et de la présentation) et le travail supplémentaire pour obtenir une meilleur note.

Données

Au minimum, vous implémenterez le chargement des données avec un **`ifstream`** et un **`getlin(file, value, c);`** qui lit le fichier **`file`** et met dans la string **`value`** le résultat de la lecture jusqu'au prochain caractère **`c`**. Les fonctions **`stof(…)`** convertissent une **`string`** en **`float`**. Vous devez implémenter la fonction **`loadData`** suivante :

`void loadData(string fn, vector<int> &vols, vector<float> &vals);`

qui lit le fichier **`fn`** et remplit les vecteurs vols et vals avec les 2 dernières colonnes.

Pour aller plus loin, vous pourrez implémenter un lissage Gaussien qui remplace chaque valeur par une moyenne pondérée des valeurs voisines. Chaque voisin doit avoir comme pondération le nombre de Gauss associé :

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Dans l'illustration ci-contre la courbe de valeurs au premier plan est lissée mais pas la seconde.

Pour implémenter le lissage vous remplirez la remplirez les fonctions suivantes :

`vector<float> computeGaussianKernel(int kernelRadius);`

qui retourne un tableau de nombres Gaussiens et

`vector<float> smoothData(vector<float> &data, vector<float> &kernel);`

qui moyenne les valeurs de data et retourne un tableau de donnée lissé

Modèles 3D

Au minimum, vous implémenterez le chargement de deux VAO. Le premier VAO contient les points au dessus de la courbe. Le second VAO contient les données des bords (4). Ces chargements se feront dans deux fonctions génériques qui pourront être appelées plusieurs fois pour charger plusieurs courbes :

```
int loadModelA(vector<int> &vols, vector<float> &vals, GLuint VertexArrayIDA);
int loadModelB(vector<int> &vols, vector<float> &vals, GLuint VertexArrayIDB);
```

Chaque fonction retourne le nombre de « vertices » créés.

Pour aller plus loin, vous rajouterez une épaisseur variable en fonction de la racine carrée du volume des action échangées et vous calculerez la normale à chaque surface. Les normales devront être enregistrées dans un nouveau VBO pour être utilisées dans la partie suivante.

Couleurs et ombrages

Au minimum, le dessus de la courbe (données du premier VAO) sera coloriée en fonction de la valeur (valeur minimum = rouge, maximum=vert) ou en fonction de la dérivée (quand l'action monte elle est verte, grise quand elle stagne et rouge quand elle perd de la valeur). Les bords (les données du second VAO) de chaque courbe 3D seront coloriés uniquement dans le fragmentShader (donc pas de VBO avec des couleurs !) en fonction de la hauteur. Comme dans les illustrations les pixels en z>0.5 seront coloriés dans une variation de tons de vert et les pixels en z<=0.5 seront coloriés dans une variation de tons de rouge.
Pour aller plus loin, vous implémenterez un ombrage de Phong pour le dessus de la courbe avec une source lumineuse placée au dessus du maximum de chaque courbe et une forte composante spéculaire. Vous pouvez aussi passer le nombre de bandes vertes et rouge dans une variable uniforme des shaders.

Interaction

Au minimum, vous permettrez la rotation et l'arrêt de la rotation en appuyant sur une touche de votre choix.
Pour aller plus loin, vous rendrez les lissages (valeurs et volumes) interactifs (en appuyant sur d'autres touches le lissage changera)

Rendu final

Chaque étudiant/binôme devra remettre à l'enseignant un rapport (entre 3 et 4 pages avec les images) organisé afin de décrire comment vous avez utilisé les technologies demandées, un code C++ fonctionnel et commenté (dans une archive zip à votre nom). En outre chaque étudiant/binôme devra faire une démonstration devant les enseignants lors de la dernière séance afin de défendre son approche, ses choix et montrer les possibilités des interactions.

Bonne chance !