Projet- L2- IGSD

Université Paris Saclay

Lynda ATTOUCHE

Yacine MOKHTARI

Groupe : 01

Introduction :

lors de ce projet nous avons eu comme objectif de réaliser un GapChart représentant le championnat anglais de football à l'aide d'OpenGL. Pour se faire nous avons eu à notre disposition un fichier csv qui décrit le classement et le score de chaque équipe sur chacune des journées sachant que nous avons 39 journées et qui représentera nos primitives pour les transformations qui suivent.

Etape 01 : "Données"

Comme nous l'avons cité, afin d'avoir notre GapChart nous avons besoin d'extraire des données fournies par le fichier csv. Pour se faire il a été demandé d'implémenter la fonction LoadData.

Afin de parvenir à extraire les points et classements de chaque équipe nous nous sommes appuyés sur les modules : fstream pour lire le fichier et sstream pour manipuler les lignes de ce dernier. Nous avons parcouru chaque ligne jusqu'à avoir un saut de ligne. Puis pour obtenir les données une par une nous avons utilisé un getLine de nouveau mais cette fois ci sur les lignes et jusqu'à arriver à une virgule.

Avec des conditions que nous avons instaurés en s'appuyant sur le fichier nous avons pu extraire le rang de chaque équipe et ses points pour chaque journée et nous les avons ajouté à nos vectors. Pour réaliser cette dernière étape, nous avons utilisé un pushback.

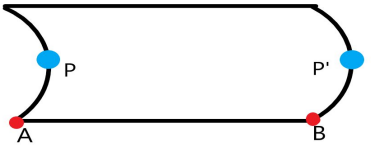
Pour aller un peu plus loin, nous nous sommes intéressés aux textures. En effet nous avons charger les textures et stocker leur IDs dans un tableau "IDs". Nous nous sommes appuyés sur la librairie "opencv" pour charger les images fournies et pour attribuer à chaque texture un ID nous avons utilisé la fonction Load Texture vue dans le "main6". Nous avons inséré cette tâche dans la fonction "LoadData" précédente, donc si lors de la lecture de nos ligne si nous nous retrouvons dans la 1ere colonne, celle qui contient les équipes nous chargeons alors pour chacune d'elle son ID.

Etape 02 : "Modèle 3D"

Dans cette étape nous avons pour but de donner forme à nos primitives soit construire notre GapChart de base.Pour se faire nous avons commencé par créer un VAO pour chaque équipe soit 20 au total. Afin de contenir nos points ainsi que leur couleur nous avons déclaré deux tableaux : g_vertex_color_data et g_vertex_buffer_data , le premier sera notre tableau de couleur et le second le tableau qui contiendra les coordonnées des points à tracer. En ce qui concerne le tableau de couleurs nous allons nous y intéresser dans l'étape suivante. Dans cette étape nous allons plutôt nous préoccuper du tableau des points à tracer.
Donc afin de tracer la succession des demi cylindres qui constituent le GapChart de chaque équipe nous avons un processus qui se répète pour chaque équipe et en chaque journée ce qui se traduit dans le code par deux boucles, la première parcourant les équipes et la sous boucle parcourant les nos journées.
Explication du processus :
Vu que nous sommes partis sur l'idée de tracé avec des triangle Strip nous avons donc besoin de 3 sommet qui se renouvellent à chaque point de périmètre, d'ailleure nous avons choisis un périmètre de 40 points.

Le demi cylindre droit :

Nous considérons les deux points A et B. Les points que nous recherchons seront tous d'abscisses xA et xB.
Ainsi on fixe ces deux valeurs, donc pour chaque point du périmètre nous allons avoir : $x_p = x_A$ et $x_{p'} = x_B$



Pour les y nous nous sommes avons trouvé la formule de trigo qui correspond soit :

$$y_p = y_{p'} = dy + \frac{l}{2} \times \frac{cos(\pi iteration)}{Nbr_{pts\ Périmètre}}$$

De même pour les z : $z_p = z_{p'} = \frac{l_2}{2} \times \frac{sin(\pi iteration)}{Nbr_{pts\ Périmètre}}$

Avec : l : Rayon de la courbe d'une équipe ; l_2 : Rayon du cylindre

dy qui est égale à la formule donnée soit :..

$$dy = \frac{\frac{19-rank[(equipe*41+journees)]}{19}. + \frac{pts[(equipe*41+journees)]}{98} + \frac{19-rank[(equipe*41+journees)]}{19} + \frac{points[(equipe*41+journees)]}{98} + l}{2}$$

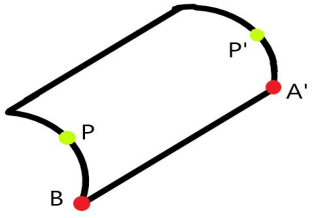
Le demi cylindre penché :

Les deux demi cylindres partagent tous les deux le point d'où le fait d'avoir une succession donc nous commençons là où le premier demi cylindre se termine. On applique le même procédé que précédemment avec de nouveaux points : soit

$x_p = x_B$ et $x_{p'} = x_{A'}$

$$y_p = y_{p'} = dy + \frac{l}{2} \times \frac{cos(\pi(Nbr_{pts\ Périmètre} - iteration)}{Nbr_{pts\ Périmètre}}$$

$$z_p = z_{p'} = \frac{l_2}{2} \times \frac{sin(\pi(Nbr_{pts\ Périmètre} - iteration))}{Nbr_{pts\ Périmètre}}$$



nous avons effectué une soustraction (Nbr(pts Périmètre) - itération) car nous nous remplissons les demi cylindres dans le sens inverse.

Avec un dy différent du précédent car ici nous passant à une autre journée, soit :

$$dy' = \frac{\frac{19-rank[(equipe*41+journees+1)]}{19}. + \frac{pts[(equipe*41+journees+1)]}{98} + \frac{19-rank[(equipe*41+journees+1)]}{19} + \frac{points[(equipe*41+journees+1)]}{98} + l}{2}$$

c'est en appliquant cette méthode que nous avons obtenu notre GapChart.

Remarque :

Dans notre code nous avons ajouté un(-1) aux expressions de dy et dy' afin qu'il ne soit pas en dehors de la fenêtre.

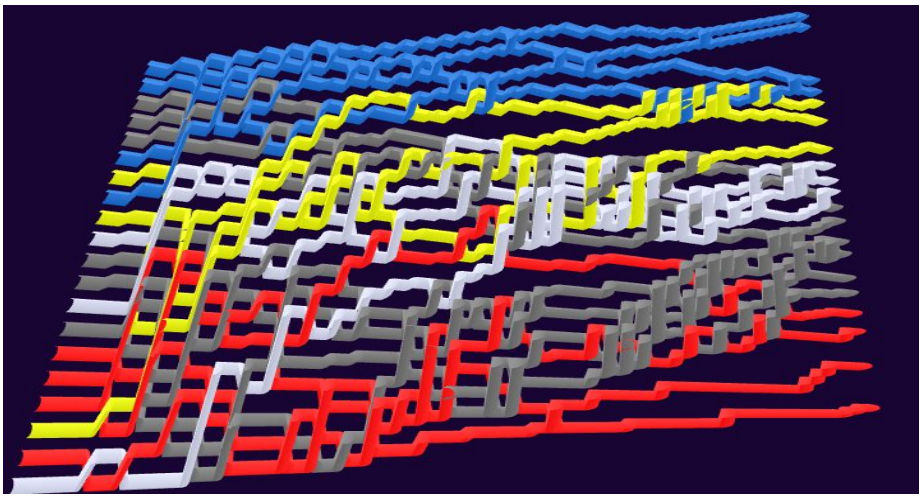
Etape 03 : " Couleurs et ombrages"

Dans cette étape, nous avons, au lieu de remplir notre tableau de couleurs avec seulement le code RGB, nous avons utilisé une règle de trois de façon à ce que les couleurs du haut de chaque courbe soit de plus en plus clair par rapport au bas (qui représente la couleur de base choisie) pour ce faire, nous avons introduit une nouvelle variable (Ombre) qui effectue le dégradé de couleur avec une simple règle de trois pour les journées, mais un

complémentaire pour la montée (ce qui est naturel car, comme expliqué ci-dessus, lors du passage d'une journée à l'autre, on remplit notre tableau des points pour les cylindres "penchés" en partant du haut du cylindre).

Nous avons aussi fait en sorte de faire passer les courbes qui "croient" vers le dessus et d'afficher par derrière (en arrière-plan) les courbes qui "décroient" et ce en introduisant un nouveau tableau de données (nommé gainLoss) qui prend la valeur 0 en tant normal (càd lors du remplissage d'une journée), et qui prend la valeur "gaLoss = 2" pendant le remplissage des courbes "penchées" si jamais la courbe de la journée d'après est supérieure à la courbe de la journée en cours. On envoie ce tableau au VertexShader qui ajoute donc la valeur (qui est soit 0 soit gaLoss = 2) à "depth".

Voici en image le résultat obtenu :



Etape 04 : "Caméra"

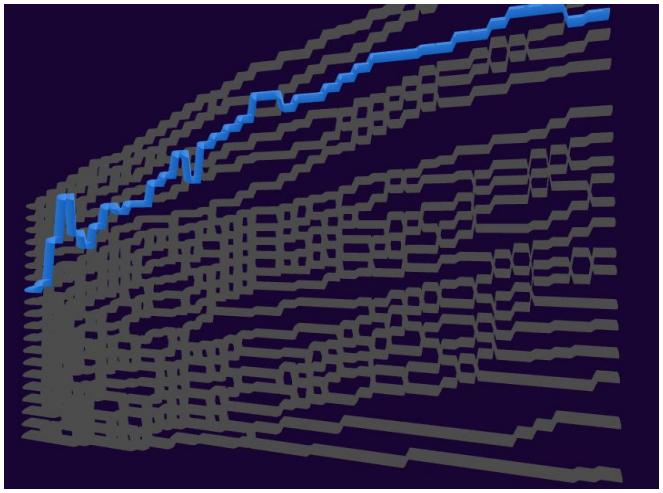
Dans cette étape, nous avons capturé les touches du clavier grâce à une fonction key_callback() qui fait appel à des fonctions natives d'OpenGL telle que glfwGetKey().

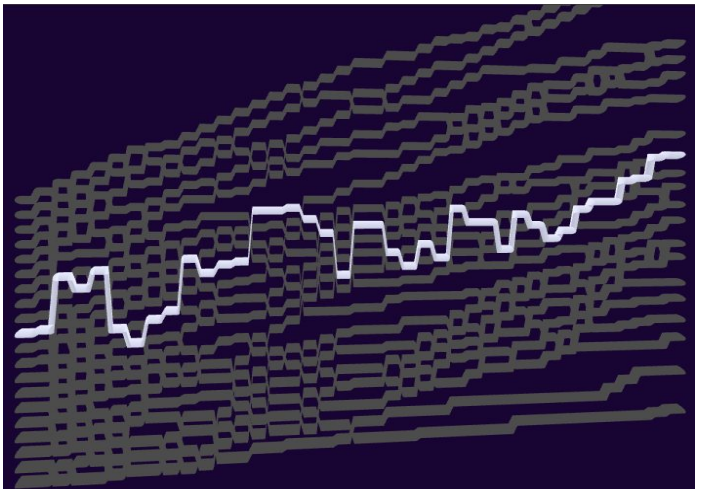
Nous faisons tourner le modèle sur lui même grâce au flèches du claviers (gauche et droite rotation sur un rayon égal à 2, haut et bas pour une semi rotation vertical), nous avons également permi à l'utilisateur de bouger non seulement la caméra mais aussi la position où elle "regarde", nous pouvons monter, descendre, aller à gauche et aller à droite grâce aux touches Z, S, Q et D respectivement. Nous avons aussi implémenté une fonctionnalité qui permet de zoomer et de dézoomer, ce grâce aux touches E et A respectivement.

Etape 05 : " La sélection d'équipe "

Ici, nous envoyons une variable nommée "compteur" (de type int) à notre vertexShader pour savoir quelle courbe doit-il colorier et quelles sont les autres courbes qu'il doit mettre en gris.
Du côté programmation, compteur est initialisé à -1, lors de l'affichage, on envoie
0 pour lui faire comprendre de colorier toutes les courbes. Sinon, grâce aux touches H et B, cela permet d'incrémenter et de décrémenter la variable "compteur" et la faire varier entre 0 et 19, ainsi un test sera effectué avant l'exécution d'un VAO. Par exemple si compteur = 5, on enverra la valeur 1 lors de l'execution du VAO n°5 au vertexShader qui se chargera de colorier cette équipe, et 2 pour tous les autres VAOs (ce qui les colore en gris par défaut).

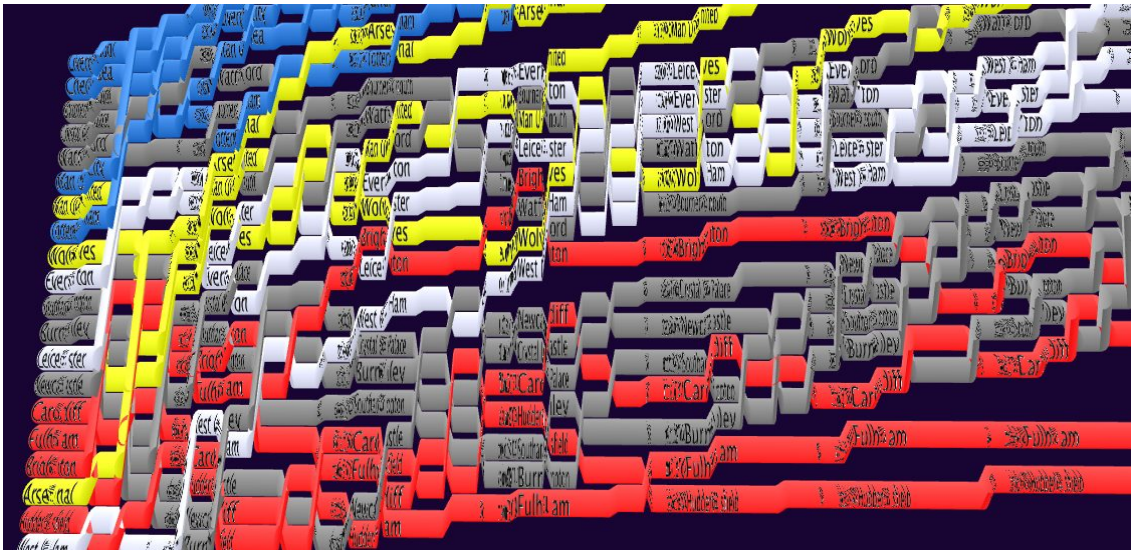
Voici un exemple du résultat :





Etape 06 : “Les textures”

Grâce au tableau teamIDs[ ] que nous a fournit la fonction loadData(), nous avons déclaré un nouveau tableau texCoord qui contiendra les coordonnées UV d’une texture, nous le passerons ainsi au vertexShaders qui lui les passera au fragmentShader pour faire correspondre chaque bout de la texture au bout du cylindre correspondant. Voici le résultat final que nous avons obtenu :



Conclusion :

Pour conclure nous avons comme annoncé à l’introduction par des données dans un fichier csv que nous avons pris comme primitives, nous avons puis passer par le chargement des données, le modèle 3D, couleurs, ombrages,caméra,.. pour finalement avoir un GapCHart qui représente assez bien le championnat anglais avec lequel nous pouvons interagir grâce aux touches du clavier.