HUNOT-MARTIN
Alaric
M1 Imagine
Université Montpellier
26/09/23

Compte Rendu TP1 HAI719

Introduction:

Le but de ce TP est, dans un premier temps, de nous introduire à la programmation 3D sous OpenGL moderne en nous montrant son histoire et les évolutions qui lui ont été apportées. Puis, dans un second temps, de nous pousser à faire un rendu 3D d'un maillage d'éléphant en implémentant certaines modifications comme l'affichage des normales, des couleurs, etc. Pour ce faire, nous travaillons avec des variables std::vector qui vont stocker les collections de sommets, de triangles, de normales et de couleurs que nous allons devoir implémenter tout au long de ce TP, que nous allons par la suite envoyer au GPU.

Résolution des exercices :

-on active les vertex arrays, les normales, les couleurs et les matériaux dans la fonction init de la manière suivante :

***vertex arrays(Ex2.1q1) :** glEnableClientState(GL_VERTEX_ARRAY);

***normales(Ex2.2q3) :** glEnableClientState(GL_NORMAL_ARRAY);

***couleurs (Ex2.3q3) :** glEnableClientState(GL_COLOR_ARRAY);

***matériaux(Ex2.3q4) :** glEnable(GL_COLOR_MATERIAL);

-on crée les attributs dans la classe Mesh de type std::vector que l'on va par la suite remplir de la manière suivante :

***vertex arrays(Ex2.1q2) :** std::vector<float> positionArray ;

***triangles(Ex2.1q2) :** std::vector<unsigned int> triangleArray ;

***normales(Ex2.2q2) :** std::vector<float> normalArray ;

***couleurs (Ex2.3q2):** std::vector<float> colorArray ;

-on crée trois fonction buildvertexarray, buildtrianglearray et buildcolorarray qui vont être chargée d'implémenter chaque variable tableau stockant la collection de sommets, normales, triangles et couleurs pour chacun des sommets constituant les triangles du maillage.

-enfin on envoie les différents tableaux créés au GPU(Ex2.1q2,Ex2.2q4,Ex2.3q4):

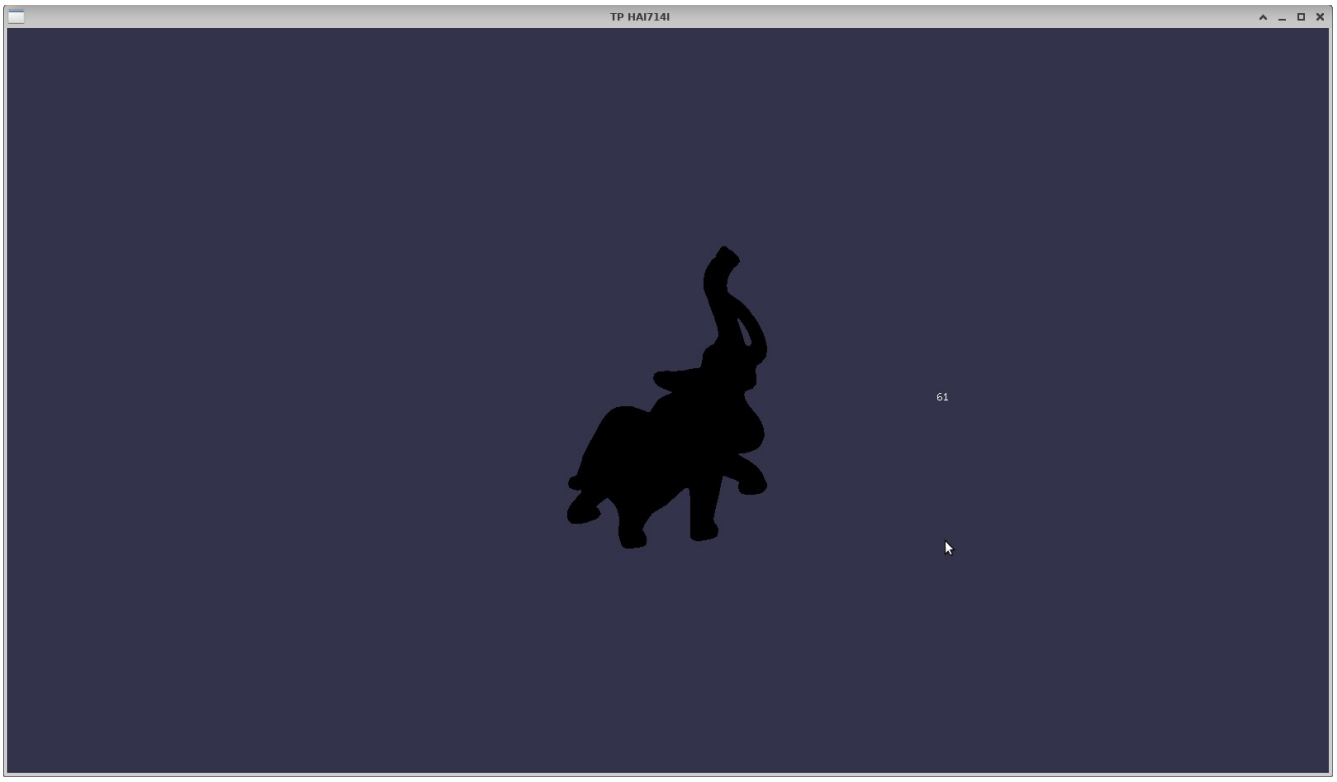
***vertex :**glVertexPointer(3, GL_FLOAT,3*sizeof(float),(GLvoid*)(&this->positionArray[0]));

***normales :**glNormalPointer(GL_FLOAT,3*sizeof(float),(GLvoid*)(&this->normalArray[0]));

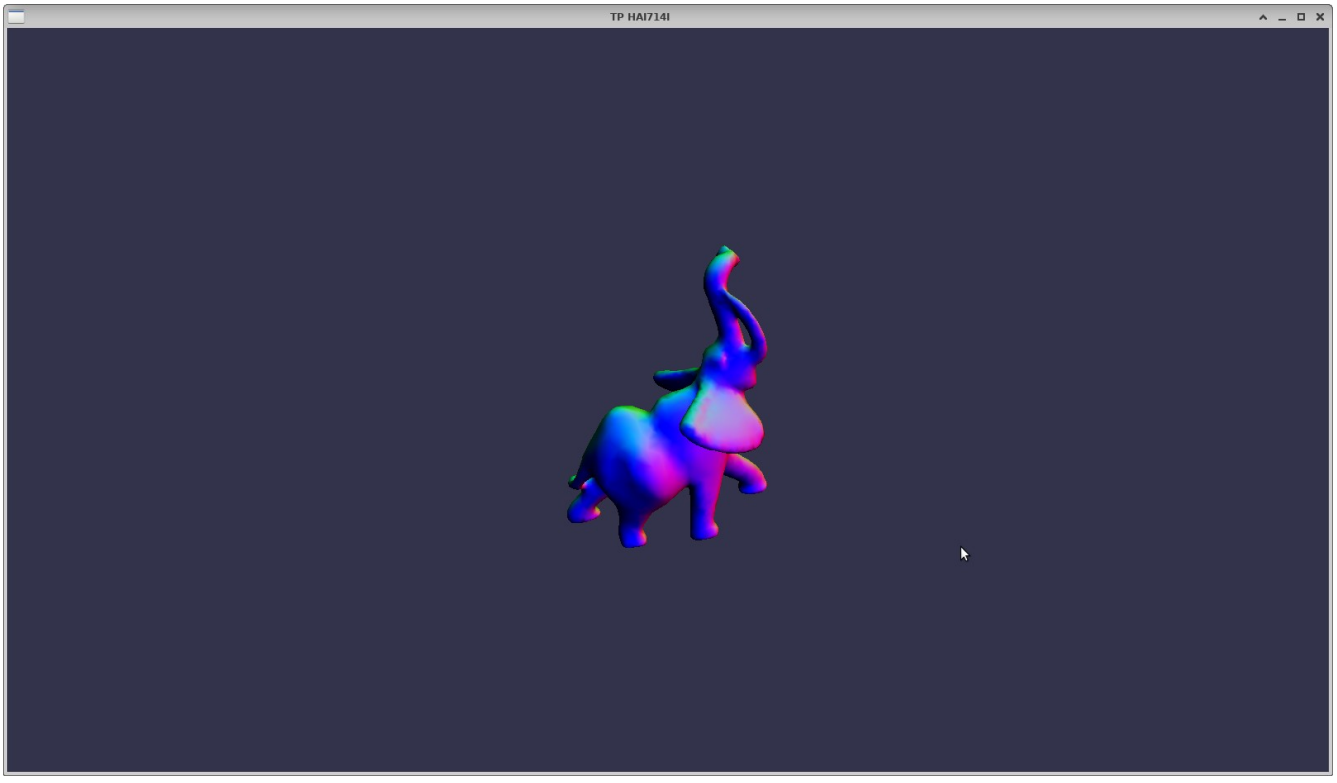
***couleurs :**glColorPointer(3,GL_FLOAT, 3*sizeof(float),(GLvoid*)(&this->colorArray[0]));

-ajout d'un compteur de FPS basé sur les différences de temps entre 2 moments donnés(facultatif).

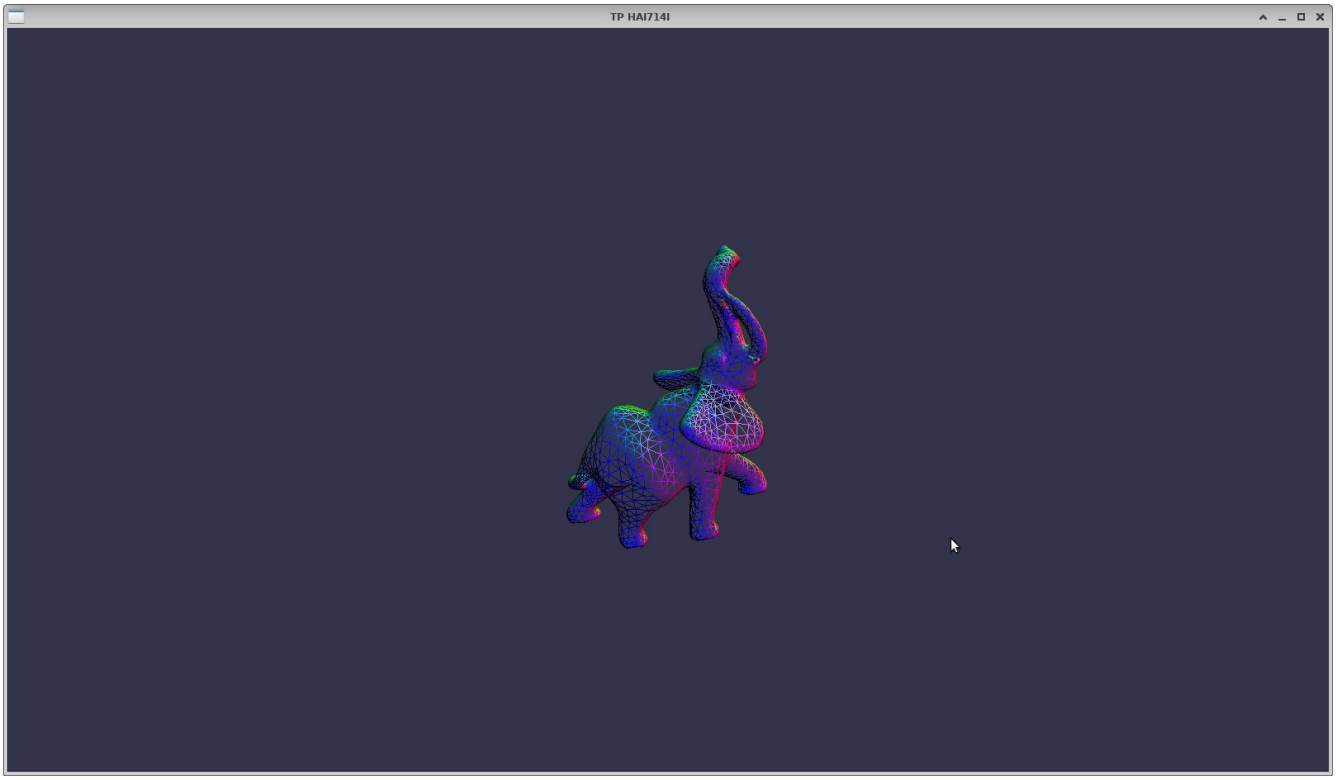
Rendu :



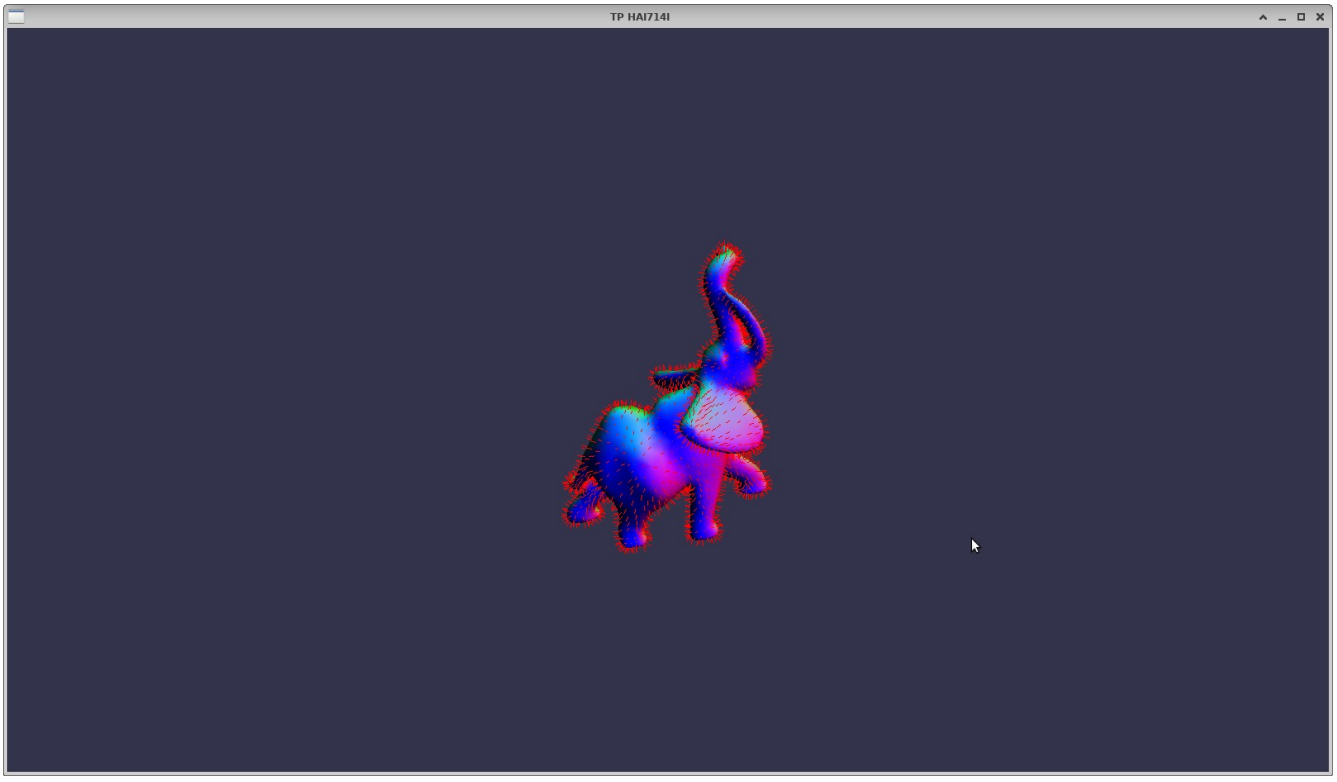
Exercice1 éléphant



Exercice 3 éléphant couleur



Exercice 3 mesh éléphant



Exercice 3 éléphant+normales

Conclusion:

Au cours de ce TP, nous avons abordé plusieurs aspects importants de la programmation 3D sous OpenGL moderne. Nous avons commencé par une introduction à l'histoire et aux évolutions d'OpenGL, ce qui nous a permis de comprendre le contexte de cette technologie.

Ensuite, nous avons exploré la création d'un rendu 3D d'un maillage d'éléphant en implémentant des fonctionnalités telles que l'affichage des normales et des couleurs. Pour cela, nous avons utilisé des variables std::vector pour stocker les collections de sommets, de triangles, de normales et de couleurs. Cette approche nous a permis de manipuler efficacement les données du maillage et de les envoyer au GPU pour le rendu.

Nous avons également activé les vertex arrays, les normales, les couleurs et les matériaux de manière appropriée dans la fonction d'initialisation. Cette étape est cruciale pour configurer OpenGL correctement avant de dessiner notre maillage.

Enfin, nous avons mis en place un compteur de FPS en calculant les différences de temps entre deux moments donnés, ce qui peut être utile pour optimiser les performances de notre application.

En résumé, ce TP nous a permis d'acquérir des compétences essentielles en programmation 3D avec OpenGL moderne. Nous avons appris à gérer les données du maillage, à configurer OpenGL pour le rendu, et même à surveiller les performances de notre application. Ces connaissances sont fondamentales pour quiconque souhaite travailler dans le domaine de la programmation graphique et de la conception 3D.