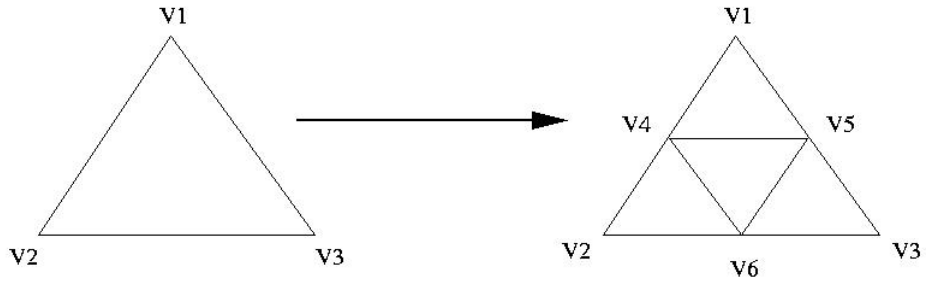
UNIVERSITÉ D'ORLÉANS **M1 Info**
Informatique Année **2021 - 2022**

Série de Travaux Dirigés : Lecteur de trajectoires : Partie 1 les sphères par icosaèdres

Ce TD est divisé en deux parties. La première consiste à créer des sphères par des icosaèdres. La deuxième consiste à visualiser la trajectoire d'une structure moléculaire où un atome est représenté par un icosaèdre. Et dans cette deuxième partie on souhaite faire le maximum de calculs sur la carte graphique via les shaders.

Exercice 1. Partie 1 : Les icosaèdres

L'icosaèdre est un polyèdre constitué de 20 triangles définis à partir de 12 sommets (cf icosaedre.hpp). Il est possible de raffiner l'icosaèdre pour se rapprocher de la sphère en subdivisant chaque triangle comme illustré par la figure ci-dessous.



- Écrivez le code nécessaire pour afficher l'icosaèdre de niveau 0.
- Proposez un premier niveau de raffinement. Attention le principe du raffinement est simple mais lorsqu'on raffine 2 triangles qui partagent un côté, il faudrait éviter de générer 2 fois le même point.
- Ajoutez une lumière afin d'améliorer le rendu (lisser la surface).

Exercice 2. Partie 2 : Le lecteur de trajectoires

Pour démarrer cette partie vous disposez d'une première version du lecteur de trajectoire. Pour cette version, les atomes sont représentés par des points dont on a augmenté la taille. Sinon les différentes parties de cette version sont

- `xdrfile` et `lecteur_trajectoire` dans *Common* qui contiennent les codes pour lire une trajectoire compressée (les fichiers .xtc). En particulier dans `lecteur_trajectoire` on trouve les fonctions pour lire les positions des atomes d'une frame enregistrée. Cette partie n'est pas à comprendre. Ce n'est pas de l'OpenGL juste ce qui permet de travailler sur des trajectoires.
- `lecteur` avec
  - `get_first_frame` qui utilise les fonctions précédentes pour lire la première frame et pour initialiser le tableau des positions des atomes ainsi que le tableau des couleurs.
  - `get_frame` qui permet de lire les frames suivantes.
- `code01_trajectoire` qui contient la partie OpenGL pour le rendu de la trajectoire. En particulier les fonctions `init` et `display` qui créent les VBO et VAO nécessaires et réalisent l'affichage sous forme de points.

Désormais on souhaite représenter les atomes par des sphères.

L'objectif de cet exercice est de modifier le code pour représenter l'atome par un icosaèdre mais en transmettant le minimum de choses à la carte à chaque frame et en faisant le maximum de calculs sur la carte graphique. Pour cela vous allez définir

1. un tableau de sommets qui va être constitué d'autant d'icosaèdres élémentaires que d'atomes. Ce tableau est transmis qu'une seule fois à la carte graphique.

2. un tableau de couleurs qui permet d'associer la bonne couleur à chaque sommet de chaque icosaèdre. Lui aussi n'est transmis qu'une seule fois.

3. un tableau qui permet d'enregistrer la translation à appliquer à chaque icosaèdre pour placer l'atome qu'il représente au bon endroit. Ce tableau contient simplement la position des atomes et est transmis à chaque nouvelle frame à la carte graphique. Pour cela vous devrez utiliser un attribut supplémentaire associé aux sommets.

4. un vertex shader qui applique le bon déplacement aux sommets de chaque icosaèdre. Le fragment shader est classique pour attribuer une couleur à partir de l'attribut couleur transmis à la carte.

Remarques supplémentaires

- le programme se lance par

  `./code01_trajectoire Data/petite_trajectoire.xtc Data/petite_trajectoire_couleurs.txt`

- Les données sont dans le dossier *Data*
  - Les fichiers `petite_trajectoire.xtc` et `trajectoire.xtc` contiennent des trajectoires compressées
  - Les fichiers `petite_trajectoire_couleurs.txt` et `trajectoire_couleurs.txt` sont respectivement les fichiers associés à la trajectoire et qui contiennent la liste des atomes qui constituent la structure. Seul le type est indiqué et c'est utile à la fonction `void choix_couleur(const char s, float *rgb)` pour attribuer une couleur (par convention) à chaque atome.

- Ci-dessous un exemple de ce que vous pouvez obtenir pour le moment alors qu'aucun éclairage n'est défini et avec le raffinement 0 de l'icosaèdre.