Compte-Rendu
Projet Synthèse d'image

Arbres de scènes en résolution adaptative

I - Introduction

Ce projet est à réaliser et à rendre dans le cadre de la première année du master informatique de l'université Gustave EIFFEL pour la matière synthèse d'images. Ce projet a été réalisé dans le but d'appliquer les notions vues en cours et de les mettre en pratique pour en faire des scènes d'objets en passant par des arbres. L'entièreté de ce projet a été coder en langage C. La bibliothèque utilisée pour ce projet est la librairie G3x.

II - Modélisation des formes canoniques et affichage

Pour cette partie du projet, il consistait simplement à créer des formes canoniques simples telles que le cercle, le cube ou encore le cylindre ou le tor. Pour ce faire nous allons nous servir de vertex. En effet nous allons remplir pour chaque forme un tableau de vertex qui va correspond au résultat des équations paramétriques des différentes formes.

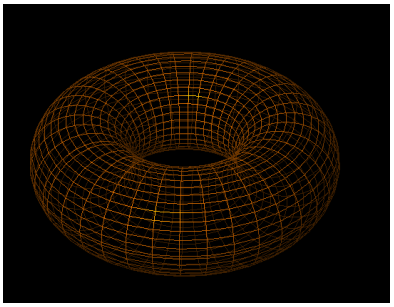
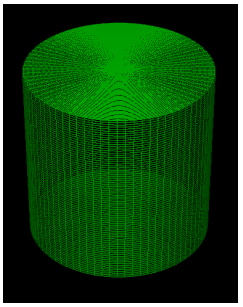
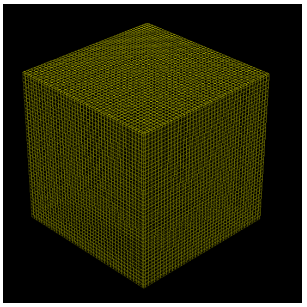
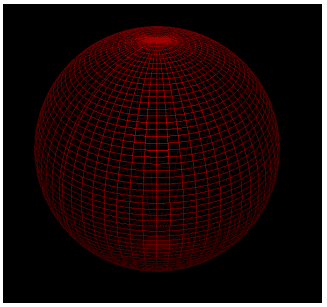
Toutes les formes suivent le même schéma, elle possède une fonction d'initialisation qui va permettre de remplir nos tableaux de vertex et de normales, et une fonction de dessins.

La fonction de dessins va alors simplement parcourir nos tableaux de vertexs et de normales et ainsi appeler les fonctions pour effectuer les dessins de nos formes.

```
g3x_Normal3dv();
g3x_Vertex3dv();
```

La structure des "Shapes", nous a été donnée dans le sujet du projet pour nous simplifier la réalisation de celui-ci.

Voici un petit aperçu des différentes formes que nous avons pu observer à ce stade du projet.

III - Arbres de scène & Transformations géométriques

Maintenant que nous avons toutes nos instances ou formes qui fonctionnent et qu'elles peuvent être affichées à l'écran, il faut maintenant pouvoir réaliser des scènes en créant des objets.

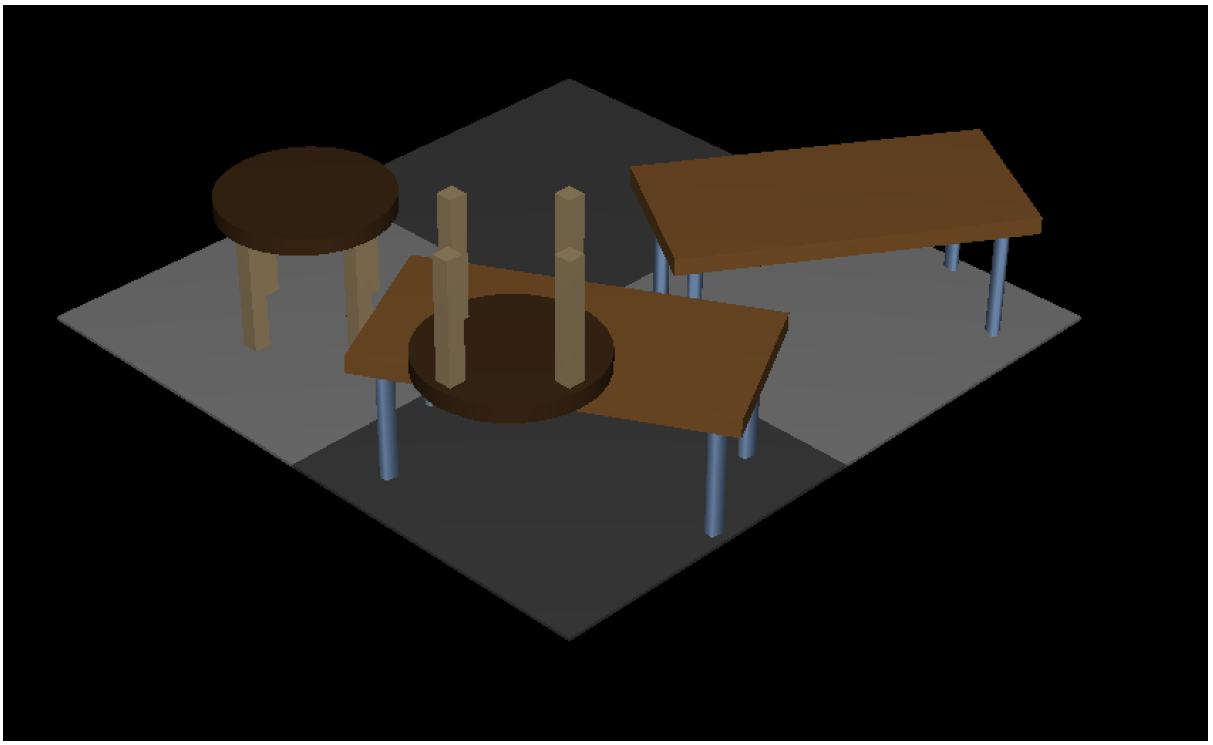
Dans notre projet, un objet peut être ramené à une succession de formes simples et mises bout à bout rendre un objet plus ou moins réaliste. L'application de transformations, d'homothétie et de rotation va nous permettre d'avoir un rendu encore plus réaliste.

Concrètement, un objet est un arbre où chaque nœud possède un fils et/ou un frère.

Ici encore, la structure d'un nœud à été fournie dans le sujet du projet pour faciliter la compréhension et la mise en forme de nos scènes.

Il est important de noter qu'un nœud va transmettre à tous ses fils, sa matrice de transformation, ainsi que toutes ses propriétés relatives à l'affichage de celui-ci. De ce fait, lorsque l'on va appliquer une quelconque transformation a un objet, toutes les formes qui en découlent auront la même transformation d'appliquer.

Voici l'exemple de "scène" que nous avons réalisé pour illustrer nos propos, cette scène est trouvable à partir de l'exécutable "scene" :



IV - Adaptation dynamique : prise en compte des positions relatives Objet/Caméra

Pour cette dernière partie du projet, il était question ici de faire entrer la distance de la caméra dans la modélisation des scènes.

Pour ce faire, nous allons récupérer la position de la caméra grâce à la commande :

```
g3x_GetCamera()->pos
```

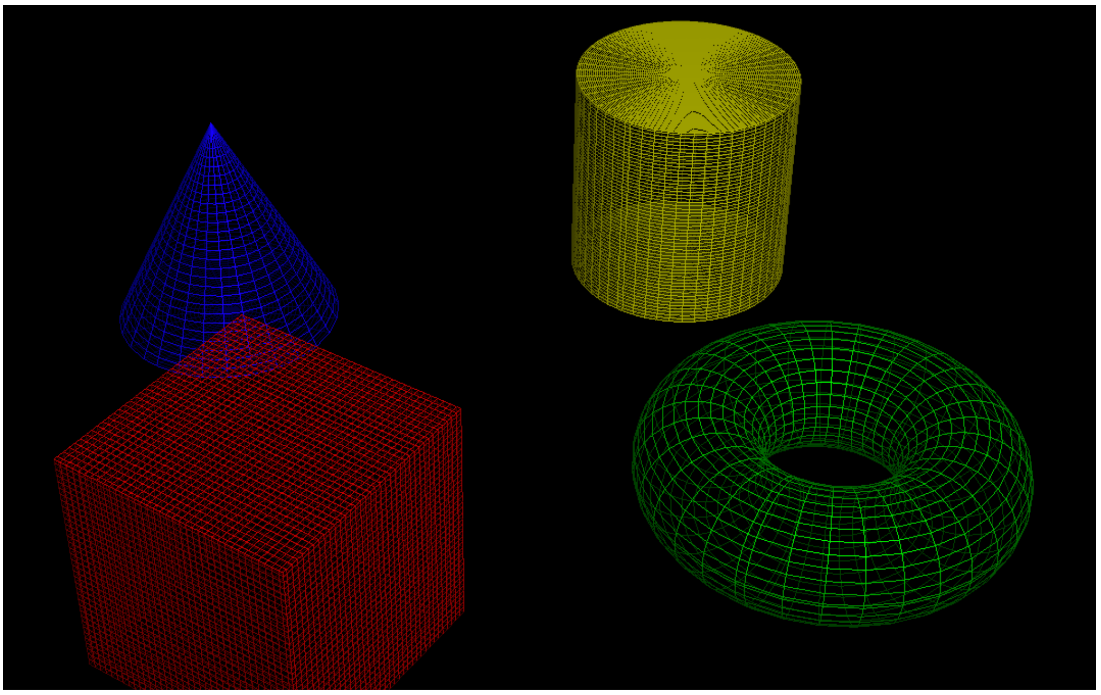
Puis grâce à celle-ci, nous allons pouvoir ouvrir la matrice de notre Shape, pour analyser et comparer la position qui se situe aux indices 12, 13 et 14 de notre matrice pour avec la position de la caméra. Cela est géré par la fonction getdistance

```
double get_distance(Node *node, G3Xpoint *cam)
{
   return sqrt(pow(node->Md.m[12] - cam->x, 2) + pow(node->Md.m[13] - cam->y, 2) +
pow(node->Md.m[14] - cam->z, 2));
}
```

Cette valeur de distance que nous récupérons aura de l'influence dans notre fonction de parcours lors du dessin car il aura une influence directe sur le pas de parcours et sur la résolution de l'objet. Plus il sera éloigné, moins la résolution sera haute. L'application de cette valeur se fait de la façon suivante :

```
int step1 = max(1, (int)(1. / (scale_factor.x * (1. / distanceCam))));
```

Nous avons mis en place une scène avec 4 Shapes à l'intérieur pour illustrer cela ainsi qu'une vidéo avec un cube dans l'archive du rendu :



V - Utilisation du programme

Pour lancer le programme principal, il faut ce rendre le dossier **makefileScene**. Puis lancer la commande

make

Cela va alors créer un exécutable nommé "scène".
Pour le lancer, tapez la commande :

./scene

Pour lancer le programme qui mettra en lumière la distance caméra. Rendez-vous dans le dossier **makefileDist** puis lancer la commande :

make

Cela va alors créer un exécutable nommé "sceneDistance".
Pour le lancer, tapez la commande :

./sceneDistances