Informatique Graphique : TP3

Exercice 1 : Cylindre

Pour rendre un cylindre, on sépare celui-ci en un certain nombre de parts, à la manière d'une tarte. Plus il y a de parts plus le cylindre aura un aspect arrondi. Chaque part est composé de plusieurs trois parties, un triangle en bas, un autre en haut ainsi qu'une face rectangulaire elle-même composée de deux triangles.

Pour définir les coordonnées de chaque point on commence donc par calculer le nombre de part, puis l'angle qu'il y a entre chaque part. On aura un total de 12 points par part.

On réalise ensuite une boucle allant de 0 au nombre de points avec une incrémentation de 12, à chaque tour de boucle on va donc définir les 12 points de la part correspondante. Même si on a 12 points dans notre représentation, certains sont partagés entre plusieurs triangles et on en a donc en réalité seulement 6.

- Le point central bas : (0, 0, 0)
- Le point central haut : (0, 0, 1)
- Le point "bas-gauche" du rectangle : (cos(a), sin(a), 0)
- Le point "bas-droite" du rectangle : (cos(a+delta), sin(a+delta), 0)
- Le point "bas-gauche" du rectangle : (cos(a), sin(a), 1)
- Le point "bas-gauche" du rectangle : (cos(a+delta), sin(a+delta), 1)

On assigne donc chacun de ces points aux triangles qui leur correspondent :

```
// Exemple triangle bas
positions[i] = glm::vec3(0.0, 0.0, 0.0);
positions[i+1] = glm::vec3(cos(angle_step*i/12),
                           sin(angle_step*i/12),
                           0.0);
positions[i+2] = glm::vec3(cos(angle_step*(i+12)/12),
                           sin(angle_step*(i+12)/12),
                           0.0);
```

Une fois les positions calculées, on l'objet est traité comme n'importe quel Renderable, et on obtient par exemple le résultat présenté en annexe 1.

Exercice 2 : Normales

On souhaite maintenant calculer les normales de chaque face, on a pour ça deux méthodes à notre disposition. La première est de la calculer à partir de l'équation paramétrique (plus rapide ici mais ne marche que quand on possède l'équation.

Ici ayant un simple cylindre on trouve facilement les normales de chaque face.

- Normale de la face supérieure : (0, 0, 1)
- Normale de la face inférieure : (0, 0, -1)
- Normale des faces latérales : (cos(a+delta/2), sin(a+delta/2), 0)

```
// Exemple de calcul de normal pour chaque face (les 2 autres points
de chaque triangle ont leur normales égales au premier)
normals[i] = glm::vec3(0.0, 0.0, -1.0);
normals[i+3] = glm::vec3(cos(angle_step*(i+6)/12),
                         sin(angle_step*(i+6)/12),
                         0.0);
normals[i+6] = glm::vec3(cos(angle_step*(i+6)/12),
                         sin(angle_step*(i+6)/12),
                         0.0);
normals[i+9] = glm::vec3(0.0,0.0,1.0);
```

L'autre solution consiste à faire le produit vectoriel entre deux vecteurs composants le triangle. Cette méthode est plus coûteuse mais marche pour n'importe quel triangle puisqu'il n'y a rien besoin d'autre que la position des points.

J'ai pour cela créé une méthode utilitaire qui renvoie la normale en fonction de trois points passés en paramètre.

```
glm::vec3 getNormal(glm::vec3 a, glm::vec3 b, glm::vec3 c) {
   return glm::normalize(glm::cross(c - a, b - a));
}
```

J'utilise tout d'abord la méthode `glm::cross` qui calcule le produit vectoriel entre deux vecteurs, puis la méthode `normalize` afin de transformer ce résultat en vecteur de taille 1.

Avec cette méthode, il faut faire attention lors de la création des triangles à l'ordre des points, sinon la normale risque d'être orientée du mauvais côté.

En mettant les normales en code couleur, on obtient le résultat en annexe 2.

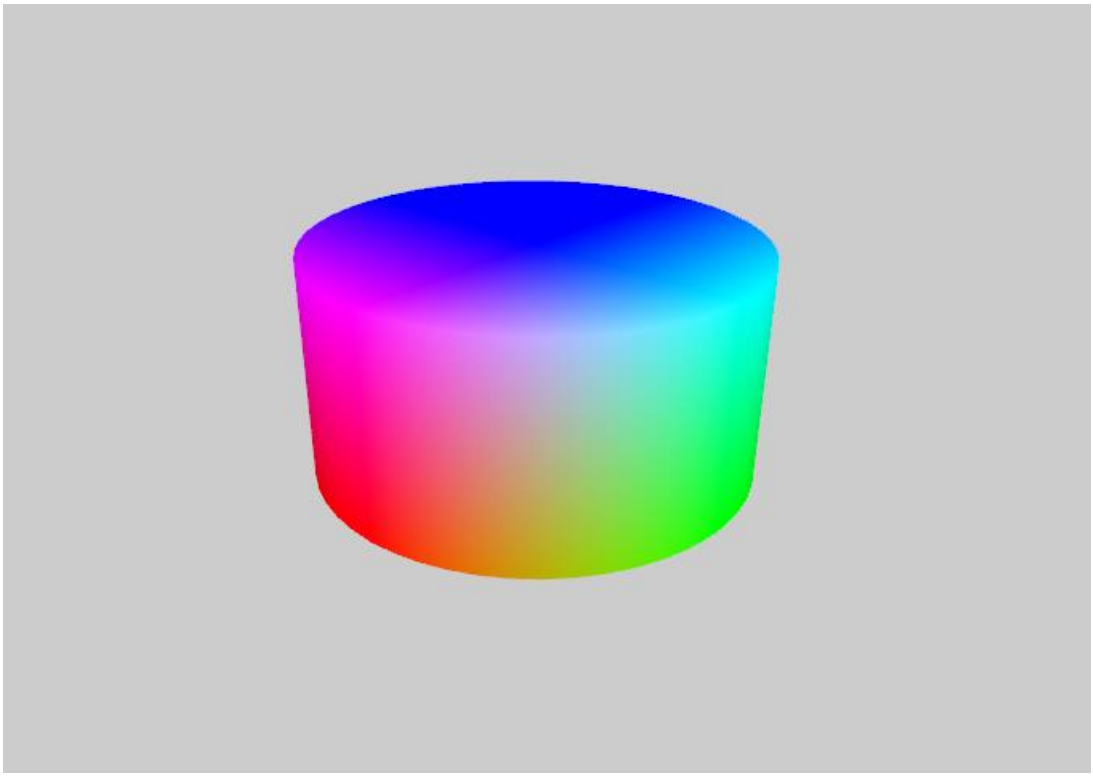
Exercice 3 : Mesh

On va ici charger un objet mesh (réalisé par exemple avec blender ou autre logiciel 3D). On appelle pour cela la méthode `read_obj` qui remplit les positions, indices et normales des points à partir du fichier `.obj`.

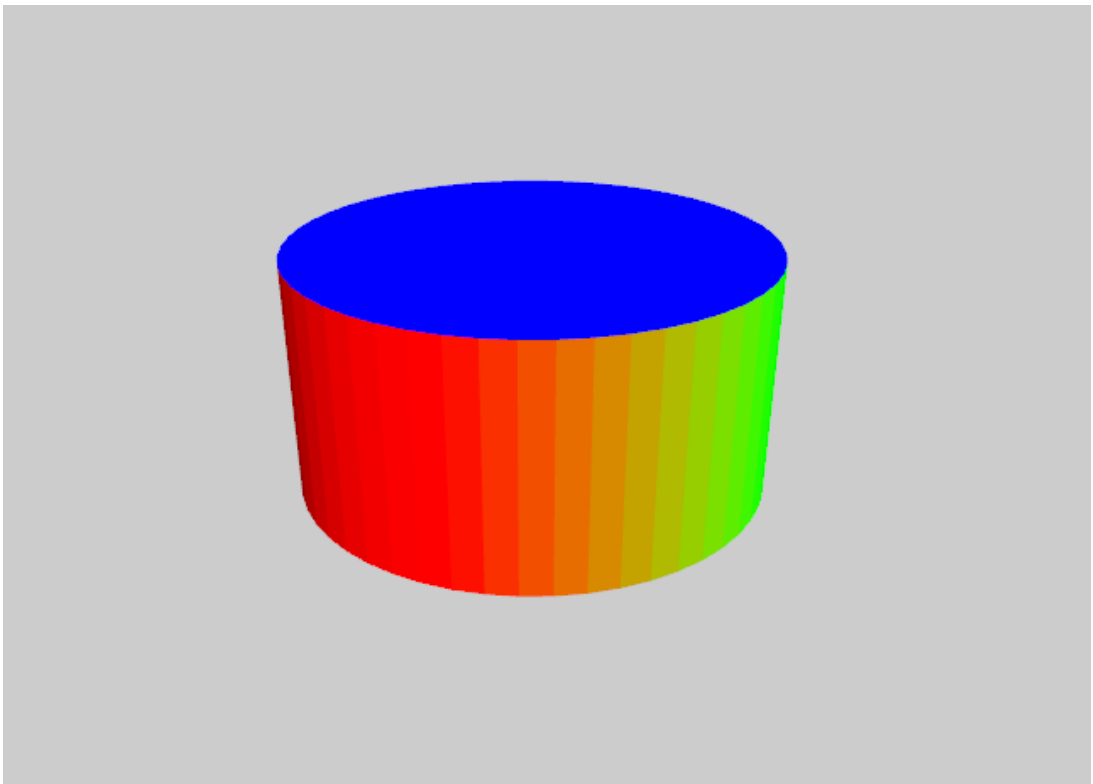
L'objet est ensuite traité comme n'importe quel `Renderable` dans le main, on crée un espace de mémoire partagé, puis on l'ajoute à la scène.

On obtient le résultat présenté en annexe 3.

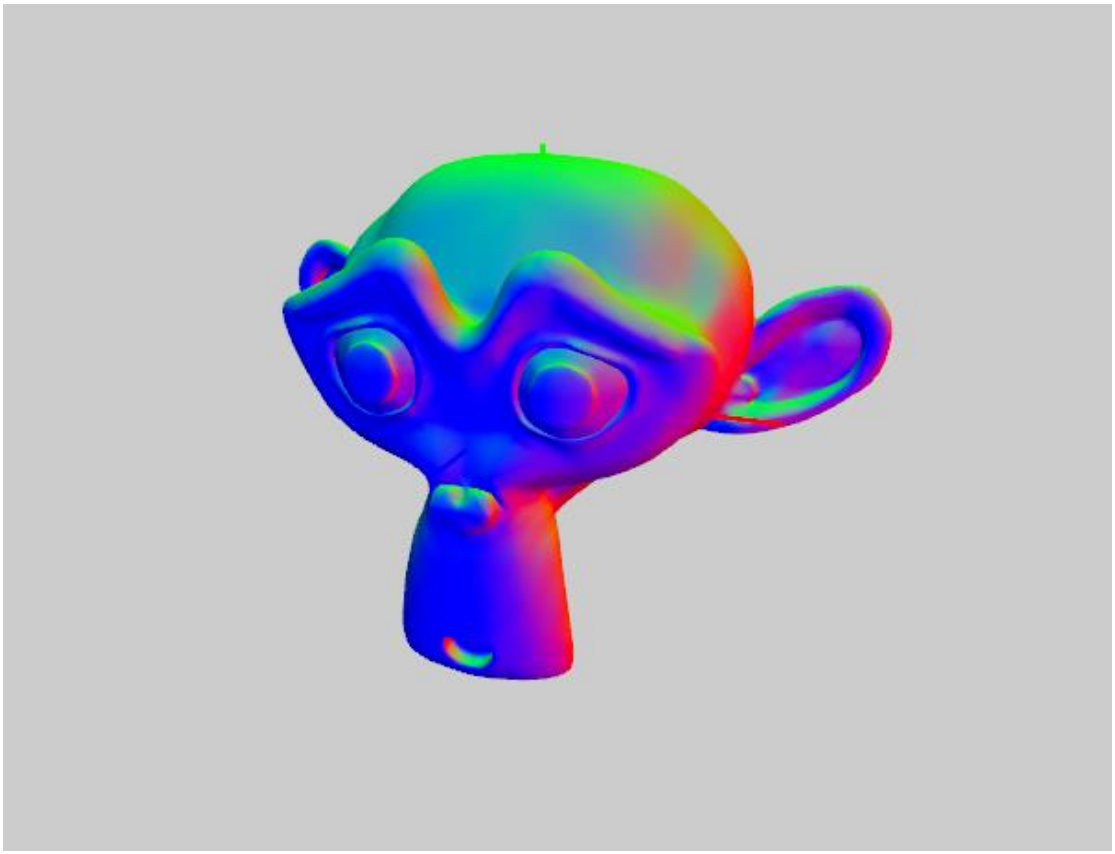
Annexes



Annexe 1 : Cylindre composé de 50 parts avec la position pour code couleur



Annexe 2 : Cylindre composé de 50 parts avec les normales pour code couleur



Annexe 3 : Objet mesh avec les normales pour code couleur