Travaux dirigés OpenGL Avancé n°3

Informatique

—IMAC troisième année—

Shaders & Illumination

Le but de ce TD est d'apprendre à utiliser les *shaders* pour produire différents effets d'illumination plus ou moins réalistes.

TD à faire en binôme et à rendre à votre enseignant.

▸ Exercice 1. *Application du modèle OpenGL*

Objectif de l'exercice : Appliquer le modèle d'illumination d'OpenGL au niveau des pixels.
Nouvelles fonctions : glUseProgramObjectARB, fonctions de chargement des shaders avec OpenGL.

Voici, pour rappel, les spécificités du modèle d'illumination utilisé dans OpenGL. Il est constitué de trois composantes principales qui sont la lumière ambiante, la lumière diffuse et la lumière spéculaire que l'on peut définir de la manière suivante :

$$ambiant_{final} = ambiant_{lumiere} \times ambiant_{matiere}$$
$$diffus_{final} = \max(\vec{L} \cdot \vec{n}, 0) \times diffus_{lumiere} \times diffus_{matiere}$$
$$speculaire_{final} = \max(\vec{s} \cdot \vec{E}, 0)^{brillance} \times speculaire_{lumiere} \times speculaire_{matiere}$$

où $\vec{L}$ est le vecteur pointant du sommet vers la lumière, $\vec{n}$ le vecteur normal unitaire au sommet, $\vec{s}$ le vecteur symétrique de $\vec{L}$ par rapport à $\vec{n}$ appelé aussi half vector, $\vec{E}$ le vecteur pointant du sommet vers l'œil et brillance le facteur de brillance.

À cela il faut ajouter le facteur d'atténuation qui a pour valeur : $facteur\,d'atténuation = \frac{1}{k_c + k_l \times d + k_q \times d^2}$

où d est la distance entre la lumière et le sommet, k_c le facteur d'atténuation constant, k_l le facteur d'atténuation linéaire et k_q le facteur d'atténuation quadratique.

Dans du code GLSL, il est possible de récupérer des états et des propriétés d'OpenGL via des variables internes telles que gl_LightSource ou gl_FrontMaterial.

Écrivez un code OpenGL qui affiche le résultat de l'utilisation d'une lumière ponctuelle sur votre scène. Écrivez ensuite un vertex et un pixel shaders permettant de calculer le modèle d'illumination précédent et ajoutez le moyen de passer d'un éclairage standard à un

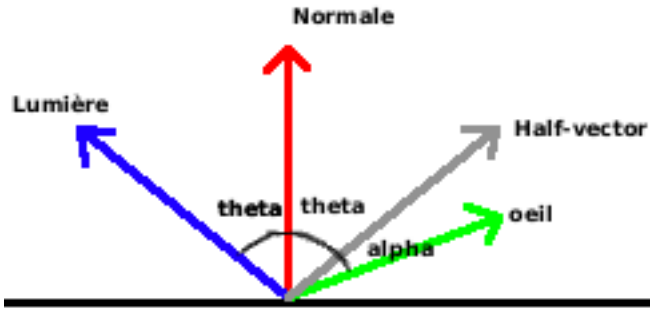

Figure 1: Modèle d'illumination OpenGL

éclairage par shaders.

▸ **Exercice 2.** ***Toon Shading Simple***

Un autre type de modèle d'illumination existant est le ***Cartoon Shading*** *aussi connu sous le nom de* ***Cel-Shading****. Il consiste en un rendu non-réaliste des couleurs qui vise à reproduire ce qu'on pourrait obtenir dans un dessin-animé ou une bande dessinée.*

Au lieu d'avoir une grande quantité de tons qui donne un aspect lissé à une surface, elle va être limitée. Cela va provoquer des transitions abruptes lors du rendu. Ainsi pour un intervalle d'intensités d'éclairage donné il va être associé un ton définit en corrélation avec la couleur diffuse de la surface.

Écrivez un vertex et un pixel shader qui permettent, en fonction d'un pas donné et de l'intensité sur la surface, d'afficher le bon ton de couleur dépendant de la couleur diffuse de la surface.

Il est possible d'afficher des contours noirs à l'affichage de l'objet pour accentuer le coté cartoon. Pour cela appliquez le morceau de code suivant :

```
glEnable(GL_CULL_FACE);
glPolygonMode(GL_FRONT, GL_FILL);
glDepthFunc(GL_LESS);
glCullFace(GL_BACK);
drawObject();
glUseProgramObjectARB(0);   /*Fin des shaders ici*/
glLineWidth(5.0);
glDisable(GL_LIGHTING);
glPolygonMode(GL_BACK, GL_LINE);
glDepthFunc(GL_LEQUAL);
```

Figure 2: Cartoon Shading

```
glCullFace(GL_FRONT);
glColor3f(0.0,0.0,0.0);
drawObject();
glDisable(GL_CULL_FACE);
glPolygonMode(GL_FRONT, GL_FILL);
```

▸ **Exercice 3.** ***Environment Mapping***

*Une autre catégorie de rendu, dite photo-réaliste, consiste à utiliser des textures de manière à ce que l'environnement d'un objet se reflète à sa surface. Il existe différentes méthodes pour arriver à cette fin et nous allons utiliser plus particulierement celle de l'**environment cube mapping**. Une **cube map** est une texture spéciale composée de six images qui se répartissent sur les faces d'un cube et dont les coordonnées de textures sont tridimensionnelles. Ces dernières représentent un vecteurs dans l'**espace objet**.*

D'une manière générale, le but de l'exercice est de calculer en chaque point de la surface de l'objet le vecteur issu de la reflexion du vecteur provenant de l'œil. Ces calculs, effectués dans un shader, necessitent certains arrangements car les calculs vont être éfféctués dans le repère de l'œil (en raison de la normale). Ainsi une fois le résultat obtenu dans le repere de l'œil, il faut le transformer pour l'obtenir dans le repère de l'objet. Un moyen pour effectuer cette opération consiste à récupérer la matrice ModelView dans le programme OpenGL, de calculer son inverse et de la transmettre au shader. Cette dernière étape peut être éffectuée via la matrice de texture qui est récupérée dans le vertex shader via la variable gl_TextureMatrix[i].

Après avoir téléchargé le programme associé à cet exercice, écrivez le vertex et le pixel shader permettant d'afficher la texture d'environnement sur l'objet.

Figure 3: Environment Mapping