Lumière !

Familiarisation avec le code

Le code actuel devrait être familier. Le programme charge un objet, le stocke dans des buffers sur GPU (model.js) puis l'affiche. Un fichier trackball.js est ici utilisé pour contrôler plus facilement la caméra. Il permet d'appliquer des rotations, translations, et mises à l'échelle en fonction des déplacements de la souris (et du bouton cliqué).

Observez model.js. Un troisième buffer a été créé pour envoyer les normales (en plus des positions des sommets et de leurs coordonnées de texture). Ces normales sont donc accessibles dans le vertex shader, comme les positions et les coordonnées.

Regardez le vertex shader. Que contient la variable « vPosition » ? Dans quel repère ? Que contient la variable « vNormal » ? Dans quel repère ? Que signifient les couleurs obtenues lorsqu'on lance l'application ? Observez le fragment shader pour voir ce qui a été dessiné dans le framebuffer.

Phong lighting

Implémentez le modèle de Phong avec une lumière directionnelle dans le fragment shader. Rappel de la formule :

$$L_o = \left[k_a + k_d\left(\boldsymbol{n}\cdot\boldsymbol{l}\right) + k_s\left(\boldsymbol{v}\cdot\boldsymbol{r}\right)^q\right]\frac{L_i}{r^2}$$

K_a, K_d et K_s correspondent respectivement aux couleurs ambiantes, diffuses et spéculaires du matériau. $\mathbf{L}_i$ est la couleur de la lumière. Vous les choisirez vous-même dans le shader dans un premier temps. L correspond à la direction de la lumière (que vous choisirez aussi comme un vecteur 3D dans le shader.) **n** est la normale, **v** est le vecteur vue (vers où la caméra regarde) et **r** est le vecteur reflété entre la direction de la lumière et la normale. r (le scalaire au dénominateur) est la distance entre le point de la surface et la position de la lumière (dans le cas d'une lumière directionnelle, r=1). Enfin, q permet de contrôler la taille du lobe de réflexion (plus il est grand, plus la surface est brillante et plus la tâche spéculaire sera petite).

Interactivité

On souhaite pouvoir modifier les paramètres de matériaux et d'éclairement en temps réel. Pour cela, il vous faut :

- Créer un formulaire dans la page html contenant des champs où l'on peut modifier les variables
- Ne plus définir ces paramètres directement dans le shader, mais les récupérer sous forme de variables de type « uniform »
- Envoyer les variables au shader à chaque fois que l'on dessine.

Un exemple d'envoi d'un vecteur 3D, contenue dans les champs « vecX », « vecY » et « vecZ » du formulaire « parameters » :

gl.uniform3f(shaderProgram.LA_LOCALISATION_DE_LA_VARIABLE,

parseFloat(document.parameters.vecX.value),

parseFloat(document.parameters.vecY.value),

parseFloat(document.parameters.vecZ.value));

Texture

Ajoutez une texture dans la scène (faire comme au TP précédent). Combinez ensuite cette texture avec le rendu actuel en utilisant par exemple la couleur à la place de la variable K_d. Les couleurs (ou intensités) contenues dans les textures peuvent en effet servir à contrôler tous les paramètres de matériaux de la scène. Vous pouvez par exemple aussi essayer de contrôler l'exposant avec la valeur de luminance obtenue (rouge+vert+bleu par exemple). C'est ce que font la plupart des artistes dans les applications actuelles.