Projet d'OpenGL
Eclairage et materiaux

Dans ce projet, nous allons implementer divers modeles de materiaux en utilisant uniquement les fonctions standard d'OpenGL. Pour une implementation plus efficace, on pourrait utiliser les nouvelles possibilites des cartes graphiques programmables (on parle alors de shader GPU). Bien que nous n'ayons pas acces a de telles fonctionalites, les notions abordees sont independantes du langage de programmation utilise.

1. Affichage d'un objet et eclairage

Recuperez le code source fourni (http://artis.imag.fr/~Pascal.Barla/opengl/projet.zip) et decompressez-le. Une classe Vec3 est fournie pour toutes les manipulations sur les points en 3D. L'essentiel du code se trouve dans prog.c. Des methodes de chargement pour fichiers .obj (objet 3D) et fichiers .bmp (textures) sont fournies. En plus des methodes permettant de manipuler la camera et la lumiere, des methodes (incompletes) permettant le controle du materiau d'un objet sont donnees, mais nous reviendront sur ce point ulterieurement.

1. Affichage d'un objet au format OBJ

Un objet en format OBJ peut etre donne en parametre (3 objets a diverses resolutions sont fournis dans le repertoire *objects*), ainsi qu'une texture au format BMP, que nous utiliserons plus tard. Un parser est fourni qui essaie de charger l'objet et la texture et utilise des versions par defaut s'il echoue. Une fois un objet charge, il est stocke sous forme de tableaux de sommets (*vertices*), de normales (*normals*) et de triangles (*faceIdx[1|2|3]*, un pour chaque sommet du triangle). Un element de faceIdx*i* est une paire d'indices correspondant a un sommet et a une normale.

Question 1 : Completer la fonction *drawObject()* en utilisant les sommets, normales et faces chargees depuis le modele OBJ.

2. Manipulation de l'eclairage

On aimerait ajouter une manipulation intuitive de l'eclairage afin de mieux visualiser notre objet. Pour cela, nous allons creer une lumiere rotative dont on pourra controler la vitesse (touches '*' et '/'), mais aussi interompre a tout moment pour controler manuellement la rotation (touches ',' et '.' correspondant aux carateres '<' et '>' en QWERTY), la rotation automatique pouvant etre a nouveau enclenchee (au moyen de la touche 'r').

Question 2 : Completer la fonction *updateAnimationParam()* afin de mettre en oeuvre la rotation de la lumiere telle qu'elle est decrite ci-dessus.

2. Modele de Phong

Le modele d'eclairage de Phong est celui utilise par OpenGL, et nous l'avons deja manipule dans les TPs precedents. Il est defini par trois composantes: ambiante, diffuse et speculaire. Chaque composante est un vecteur de 3 coefficients, un pour chaque canal de couleurs (Rouge, Vert et Bleu). Un parametre supplementaire appele *shininess* est utilise pour controler la taille de la tache speculaire (plus il est eleve, plus la tache lumineuse est petite).

Nous allons dans cette section utiliser le systeme d'eclairage d'OpenGL pour creer deux types de materiaux: plastique (active par la touche 'p') et metal (active par la touche 'm'). Les materiaux plastiques se caracterisent par une petite tache speculaire de couleur blanche, tandis que les materiaux metalliques ont une large tache speculaire d'une couleur proche de la couleur diffuse.

Question 3 : Completer les fonctions *setMaterial()* et *unsetMaterial()* afin d'obtenir les deux types de materiaux: metallique et plastique.

3. Modele Toon

Apres avoir directement utilise les fonctions de materiaux d'OpenGL, nous allons manipuler de maniere plus explicite les composantes du modele de Phong dans le but d'obtenir un rendu de type cartoon. Afin de pouvoir comparer les modeles que nous allons creer, veillez a **conserver les resultats des sections precedentes** (le passage d'un modele a l'autre se fait au moyen des touches 1 a 4 du clavier).

1. Modele hybride Phong/Toon

Nous commencons par modifier le modele de Phong en utilisant la fonctionalite *glColorMaterial()* d'OpenGL. Des lors que le mode GL_COLOR_MATERIAL est active (avec *glEnable()*), la valeur d'une composante (on utilisera *diffuse*) peut etre manipulee explicitement par le biais de l'appel a la fonction *glColor()* (voir le man de glColorMaterial).

Un rendu cartoon peut etre obtenu en **segmentant** la composante diffuse en differents intervalles, creant ainsi des bandes de couleurs lors du rendu. Nous allons creer une telle segmentation au moyen d'une methode simple qui a l'avantage d'etre controlable par un unique parametre: le nombre d'intervalles (qui devra etre controlable par les touches '+' et '-' du clavier). Chaque intervalle aura pour valeur le milieu de l'intervalle. La composante diffuse initiale est calculee par le produit scalaire entre la normale au sommet et la direction de la lumiere, les deux vecteurs devant etre unitaires: ***Diff = N.L***

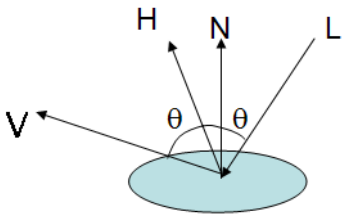
Question 4 : Completer les fonctions *setMaterial()* et *unsetMaterial()* afin d'activer et desactiver le mode GL_COLOR_MATERIAL, mais aussi creer la segmentation. Il est egalement conseille d'utiliser des coefficients ambiants nuls. Completer ensuite la fonction *drawObject()* avec des appels a *glColor()* en specifiant explicitement le coefficients diffus et pour cela, utiliser la fonction *getColor()* (a completer bien sur).

Conseil: afin de calculer la composante diffuse initiale, utiliser les fonctions *dot()* et *normalized()* de la classe Vec3.

2. *Modele Toon complet*

Une variante du modele Toon consiste a manipuler egalement la composante speculaire du modele de Phong. Cette fois-ci, nous ne pouvons plus utiliser le mode GL_COLOR_MATERIAL, puisque nous desirons controler deux composantes simultanement. Nous sommes donc forces d'abandonner les fonctionalites de materiaux d'OpenGL et de calculer completement le modele toon, pour finalement passer le resultat via *glColor()*.

La composante speculaire initiale peut etre calculee efficacement par la formule de Blinn-Phong: $Spec=(N.H)^s$ ou H represente le Half-vector: $H=L+V$, avec L et V les vecteurs correspondant a la lumiere et au point de vue respectivement (tous les vecteurs doivent etre unitaires).



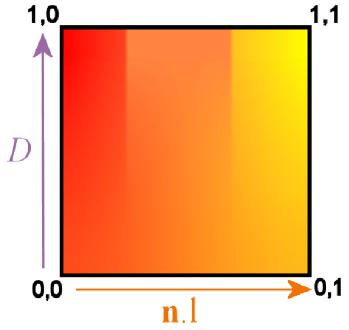
Question 5 : Completer les fonctions *setMaterial()* et *unsetMaterial()* afin d'activer et desactiver le mode GL_LIGHTING. Puis completer la fonction *getColor()* afin d'implementer un modele toon complet ou la composante speculaire sera **tronquee** par un seuil a definir.

Question 6 : Essayez les differents modeles de materiaux crees jusqu'a present (PHONG, PHONG_TOON et FULL_TOON) avec des modeles a differentes resolutions. Proposez une explication aux differences de performance et de qualite de rendu.

4. Modele X-Toon

Nous avons vu comment implementer un modele toon en calculant explicitement un nombre d'intervalles diffus et en tronquant la composante speculaire. Une autre approche consiste a utiliser une texture toon 1D pre-segmentee pour la composante diffuse. L'avantage est que l'on peut facilement realiser des transitions douces entre chaque intervalle et faire varier la couleur, comme montre ci-dessous.

Nous allons egalement en profiter pour implementer un modele cartoon plus elabore, nomme X-Toon (X pour eXtended). L'idee est simple: utiliser une texture toon 2D, illustree ci-contre, ou la nouvelle dimension correspond a une notion de detail sur laquelle nous allons jouer de differentes manieres dans la suite.



1. *Toon shader classique*

On commence par ignorer la dimension *detail* (on fixe *D=1.0*). La texture donnee en parametre du programme est automatiquement chargee et activee dans le code qui vous est fourni. Des textures en format BMP sont fournies dans le repertoire *textures*.

Question 6 : Completer les fonctions *setMaterial()* et *unsetMaterial()* afin d'activer et desactiver le mode GL_TEXTURE_2D. Puis completer la fonction *getTexCoord()* afin d'implementer un modele X-Toon simple ou la coordonnee de detail est fixee a 1.0.

2. *Extensions*

Nous allons maintenant assigner a la coordonnee *D* des valeurs differentes calculees par 3 fonctions differentes, auquelles on ajoute la fonction constante de la question precedente (on pourra visualiser les differentes fonctions via les touches F1 a F4).

Question 7 : Completer la fonction *getTexCoord()* afin d'assigner a la coordonnee *D* la distance du sommet considere a la camera. Proposer un controle (au moyen des touches '+' et '-') sur la distance a laquelle le detail minimal (*D=0*) est atteint.

Question 8 : Completer la fonction *getTexCoord()* afin d'assigner a la coordonnee *D* l'orientation du sommet considere par rapport a la camera. Proposer un controle (au moyen des touches '+' et '-') permettant de considerer des orientation plus ou moins rasantes.

Question 9 : Completer la fonction *getTexCoord()* afin d'assigner a la coordonnee *D* le coefficient speculaire (*Spec*) au sommet considere par rapport a la camera. Proposer un controle (au moyen des touches '+' et '-') permettant de faire varier le shininess.

3. *Analyse*

Question 10 : Essayez les differentes extensions avec differentes textures et des objets a de multiples resolutions. Proposez une explication aux differences de performance et de qualite de rendu en comparaison des modeles toon de la section 3. D'apres vous, en quelle mesure l'utilisation du GPU pourrait ameliorer qualite et/ou performance ?