Contrôle de l'Apparence

Travaux sur Machine Encadrés

Introduction

Ce TP vise à implémenter plusieures techniques permettant le contrôle de l'apparence des objets. Tout d'abord, en définissant les propriétés des matériaux et de la lumière. Ensuite, en s'intéressant à la question du contrôle de l'éclairage. Pour cela, plusieures méthode d'intéraction seront développées pour faciliter la tâche de la définition de l'éclairage pour un artiste. La dernière partie consistera à modifier localement les propriétés de la surface à la volée pour raffiner l'affichage.

Familiarisation avec le code

La fonction *computeLighting* va être utilisée pour la définition des shaders. Le résultat de cette fonction va définir l'apparence de la surface pour chaque sommet du modèle. Le maillage, issu des triangles, va être éclairé automatiquement à l'aide de l'information calculée localement dans la fonction. En entrée celle-ci reçoit une position sur la surface, une normale, décrivant l'orientation locale de la surface, un numéro de lumière (dont la valeur initiale vaut toujours 0) et un index pouvant être utilisé pour identifier le sommet/l'endroit éclairé du maillage.

Compilez le programme ("g++ *.cpp -l glut -I . "). Permettre de changer le mode d'affichage en appuyant sur les touches 1-4. Dans un premier temps, ceci va simplement changer la couleur de votre modèle.

Que la lumière soit !

On va simuler l'apparence d'objets et l'impact de l'éclairage sur celle-ci à l'aide des "shaders". Ceux-ci sont des petits programmes permettant d'enrichir la représentation virtuelle.

Rappel : Modèles d'illumination

Dans cet exercise, nous allons développer plusieurs modèles d'éclairage. Regardez la classe Vec3Df (à la base de Vec3D dans le fichier Vec3D.h) pour vous habituer à son utilisation (produit scalaire - dot product -, produit vectoriel - cross product -, additions, normalisations etc.).

Pour déplacer la lumière à la position de la caméra, appuyez sur l. La position de la lumière est disponible dans la variable globale *LightPos[0]*, la position de la camera dans *CamPos*.

Modèle Lambertien

Il s'agit d'un modèle d'un matériel diffus celui-ci est donné par la formule : $dot(N, L)$, où L est la direction du point sur la surface vers la lumière et N la normale.

`http://en.wikipedia.org/wiki/Lambertian_reflectance`

Implementez le modèle en supposant que la surface à une réflectance de 1.

Modèle Blinn-Phong

Il s'agit d'un modèle permettant d'ajouter des effets spéculaires celui-ci est donné par la formule : $dot(H, N)^s$, où H est le vecteur exactement entre la direction de vue vers le point sur la surface et la direction vers la lumière et s une valeur de spécularité.

`http://en.wikipedia.org/wiki/Blinn-Phong_shading_model`

Quel est l'impact de s.

Qu'observez-vous pour une lumière placée derrière le modèle ?

Comment corriger ce problème ?

Combinez les deux modèles.

Toon shading

Le toon shading donne une apparence non-continue qui quantise l'illumination. Dans notre cas, à cause de l'interpolation, le résultat n'est pas parfaitement quantisé. La figure ci-dessous illustre le résultat souhaité.

Implementez cette fonction. Traitez les spécularités séparements du reste.

FIGURE 1 – Toon Shading

Essayez également de transformer la caméra en lumière particulière. Cette lumière n'illumine pas mais elle attribuez la valeur zero dans la fonction *computeLighting* aux points pour lequels la lumière aurait été très faible : inférieure à un certain seuil - threshold - noté t.

Qu'observez vous ? Quel est l'effet de t ?

Vous pouvez changer la variable globale *BackgroundColor* pour mieux voir le résultat.

Et la lumière fut.

Maintenant, nous nous intéresserons au contrôle de la lumière. Pour un artiste, contrôler la lumière directement peut être problèmatique pour cela nous proposerons un algorithme pour définir des positions de lumière de manière indirecte.

Nous utiliserons dans la suite la fonction *userInteraction*. Cette fonction reçoit des informations liée à l'endroit où l'utilisateur a *cliqué* (comme la souris est utilisée pour la navigation, la touche espace va déclencher cette fonction à la place d'un clique). L'information fournie par la fonction est le sommet sur lequel l'utilisateur a cliqué, sa position, sa normale et l'index du sommet (ce dernier, *selectedIndex*), se verra attribué une valeure négative lorsque l'utilisateur clique à coté du maillage). Aussi, cette fonction fournit un rayon sous la forme d'une origine et d'une direction. *origin* est le centre de la caméra et le rayon passe par le pixel sur l'écran.

Dans la suite, nous utiliserons la fonction *userInteraction* pour déplacer la lumière (pour cela, changez la variable globale *LightPos[0]*).

Placer une lumière avec la souris

La lumière sera placée sur une sphère (rayon 1.5) centrée sur l'origine. Sa position doit coïncider avec l'endroit indiqué par le *clique* de l'utilisateur. Faîtes attention à choisir le point d'intersection le plus proche à l'observateur - pouvant se retrouver derrière l'observateur.

Placement basé sur l'ombrage

Cette fois, la nouvelle position sera choisie de telle sorte que la lumière soit orthogonale à l'endroit *cliqué*. Ceci a pour conséquence que la bordure d'ombrage va se retrouver (exactement) à cette endroit précis (utilisez le mode Toon-Shading pour vérifier votre solution).

Placement basé sur la spécularité

Le placement d'une specularité n'est pas facile car sa position dépend de la direction de vue ET de la lumière. **Trouvez une solution pour assurer que la spécularité apparaisse à l'endroit cliqué.**

Éclairage multiple

La touche *shift+L* permet d'ajouter des lumières dans la scène. Dans un premier temps, diminuez la puissance de la lumière en multipliant le résultat de votre calcul dans *computeLighting* avec une constante par exemple 0.4. Pour supprimer toutes les lumières sauf une, appuyez sur *shift+N*.

Éclairer comme un pro

Placez plusieures lumières afin de créer des effets comme Rim Light, Back Light, Fill Light... (ils savent ce que c'est ? ?)

Comment pourrait-on simuler une lumière large ?

Contrôle des paramètres

Ensuite, nous souhaiterons ajouter des lumières colorées. Pour cela, nous utiliserons le tableau LightColor.

Remplissez dans la fonction *keyboard* les fonctions pour les touches R, G, B, r,g,b. Pour les charactères en miniscule, nous diminuerons, sinon nous augmentons la puissance de la lumière courante légèrement. La lumière courante est indiquée par la variable globale *SelectedLight* pouvant être modifiée avec les touches + et -. La lumière courante est affichée avec une bordure.

Adaptez votre fonction *computeLight* (L'index de la lumière *light* correspondra à la source en question) pour tenir compte de la couleur de lumière.

Adaptez les fonctions de contrôle pour influencer uniquement la lumière courante. Essayez de recréer l'éclairage de la figure ci-dessous, servez vous des fonctions de placement de lumière.

FIGURE 2 – Cette scène utilise un rimlight bleu, une lumière rose, une lumière blanche et une lumière spéculaire blanche (avec une puissance de 2.0)

Expérimentez avec une lumière négative et des lumières qui ne seront utilisées que pour créer la composante spéculaire.

Edition des matériaux

Dans cet exercice, nous mettrons en place un système simple permettant l'éditer des matériaux. Pour cela, nous utiliserons à nouveau la fonction *userInteraction*.

Pour chaque sommet choisi, nous remplirons le tableau *customData*. Le shader utilisera cette information pour choisir entre différents types d'ombrage.

Changement d'un matériel

Offrez la possibilité de choisir des sommets des yeux/lèvres pour leur attribuer une spécularité comme le montre la figure ci-dessous.

FIGURE 3 – Les yeux montrent un effet de spécularité

Changement de l'influence de la lumière

Très souvent les artistes choisissent la contribution exacte de chauqe lumière à chaque partie de la scène. Nous nous intéresserons à une situation simplifiée où nous disposons de trois lumières.

Préparation : Changez la fonction *keyboard* en ajoutant les touches p, P pour changer le poids de l'influence de la lumière active. Pour cela, utilisez la variable globale *SelectedVertex* indiquant le dernier sommet choisi par l'utilisateur (-1 s'il aucun sommet n'est choisi). Nous changerons ensuite la valeur dans le tableau globale *customData* correspondant à l'indice du sommet actuel.

Utilisez l'information *customData* dans votre fonction *computeLighting* pour diminuer l'influence de la lumière autour des yeux.

Exercices supplémentaires

Avez-vous des idées pour proposer une meilleur contrôle de la lumière ? Quelle serait une bonne interface pour changer la couleur de lumière (vous pouvez ajouter des modes si nécessaire) ? Pouvez-vous définir d'autres modèles d'illumination ? Comment pourrait-on créer des ombres projetées ?