POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

INF2705 Infographie

Spécification des requis du système

Travail pratique 4

Illumination et textures

Département de génie informatique et génie logiciel
École polytechnique de Montréal
Hiver 2016

Table des matières

1 Introduction

Ce document décrit les exigences fonctionnelles et non fonctionnelles du TP4 « *Illumination et textures* » du cours INF2705 Infographie.

1.1 But

Le but des travaux pratiques est de permettre à l'étudiant d'appliquer directement les notions vues en classe.

1.2 Portée

Chaque travail pratique permet à l'étudiant d'aborder un sujet spécifique.

1.3 Références

1. Site du cours INF2705
2. Site du cours INF2990
3. Guide de programmation C++ (INF2990).

2 Description globale

2.1 But

Le but de ce TP est de permettre à l'étudiant mettre en pratique les notions d'illumination et d'applications de textures en utilisant des nuanceurs en GLSL.

2.2 Travail demandé

Partie 1 : l'illumination des objets

On demande d'implanter les modèles d'illumination de Lambert, de Gouraud et de Phong dans les nuanceurs afin d'obtenir un meilleur rendu des objets (Figure 1). De plus, dans le calcul de la réflexion spéculaire, on pourra choisir entre le modèle de Phong ou celui de Blinn (Figure 2).

Le modèle d'illumination de Lambert ne demande qu'une seule normale par facette. On utilisera donc le nuanceur de géométrie où on calculera une normale à la surface en utilisant un produit vectoriel. Ce nouveau vecteur normal sera imposé à tous les sommets (au lieu de ceux spécifiés originalement).

Partie 2 : l'utilisation d'un spot

Le programme permettra aussi d'utiliser un éclairage de type « spot » selon la définition d'OpenGL ou celle de Direct3D (voir annexe D pour les formules). Les propriétés du spot, sa position, son angle maximum (`spotCutoff`) et son exposant (`spotExponent`) sont modifiables au cours de l'exécution (Figure 3). On pourra observer l'effet de l'illumination de l'objet par rapport à son orientation.

Partie 3 : l'application de textures

Le logiciel permettra d'afficher ou non des textures sur les quelques objets illuminés.

— Affichage d'un dé à jouer sur le cube. On spécifiera les coordonnées de texture afin de montrer sur le cube un dé à jouer. La texture contenant toutes les faces du cube est fournie et elle sera utilisée sans la subdiviser en 6 textures différentes. (voir Figure 4).

— Affichage d'un patron d'échiquier sur le cube (et autres objets). On spécifiera les coordonnées de texture afin de montrer sur le cube un échiquier centré sur chaque face ou répété selon l'une ou l'autre des directions. Il doit aussi être possible de changer l'allure de l'affichage, au cours de l'exécution, suivant les rendus de la Figure 6 et Figure 7 sans modifier la texture.

Enfin, on pourra choisir que les pixels noirs soient plutôt transparents (voir la Figure 4).

2.3 Fichiers fournis

Le code fourni présente un cube avec aucun éclairage. Le code pour initialiser, charger et lancer les nuanceurs est aussi fourni. Pour démarrer, on pourra aussi utiliser les nuanceurs de l'exemple du cours : www.groupes.polymtl.ca/inf2705/exemples/06-Illumination/

Deux fichiers de texture (le dé et l'échiquier) sont aussi fournis (Figure 8), de même que les fonctions pour charger les textures en mémoire.

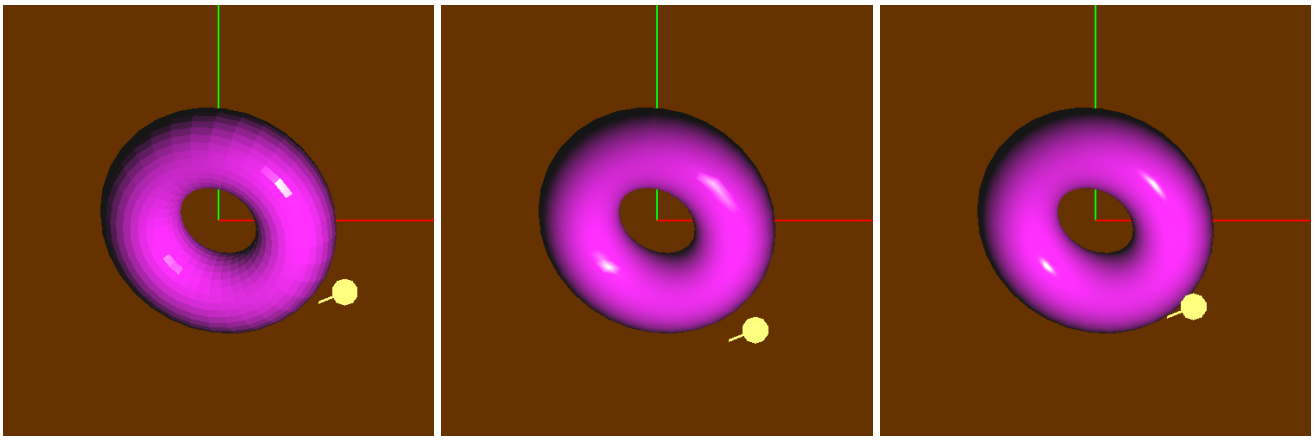

FIGURE 1 – Rendu avec illumination i) de Lambert, ii) de Gouraud, iii) de Phong. (Réflexion spéculaire de Phong dans tous les cas.)

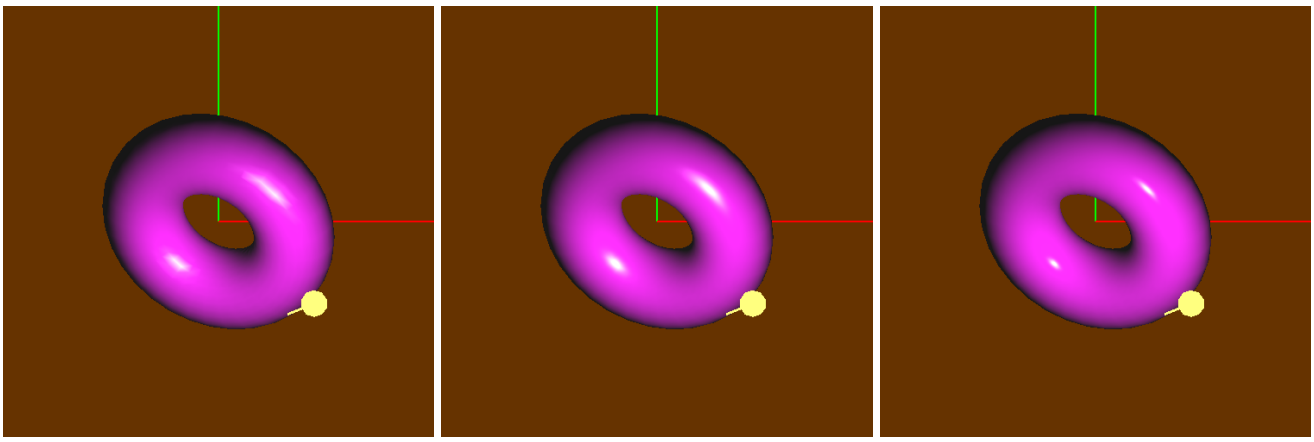

FIGURE 2 – Rendu avec i) illumination de Gouraud et réflexion spéculaire de Blinn, ii) illumination de Phong et réflexion spéculaire de Blinn, iii) illumination de Phong et réflexion spéculaire de Phong.

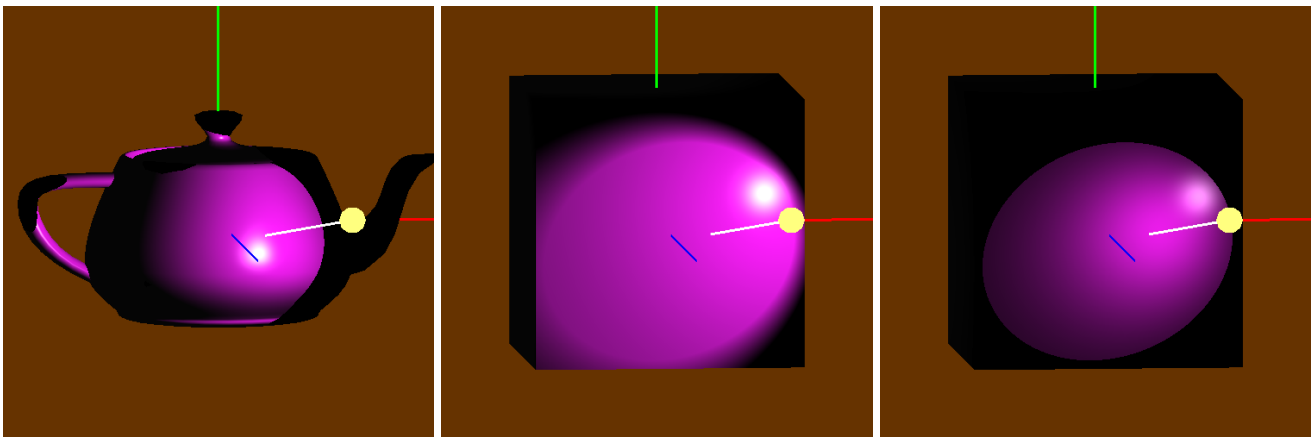

FIGURE 3 – Rendu avec le spot i) avec le modèle OpenGL, ii) avec le modèle Direct3D, iii) avec un plus grand exposant

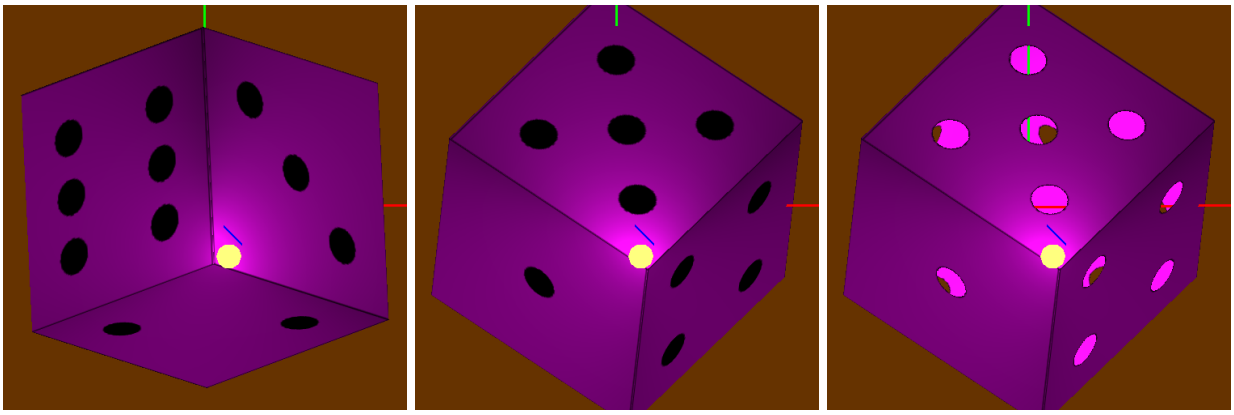

FIGURE 4 – Texture appliquée sur le dé en 3D (noir opaque ou noir transparent)

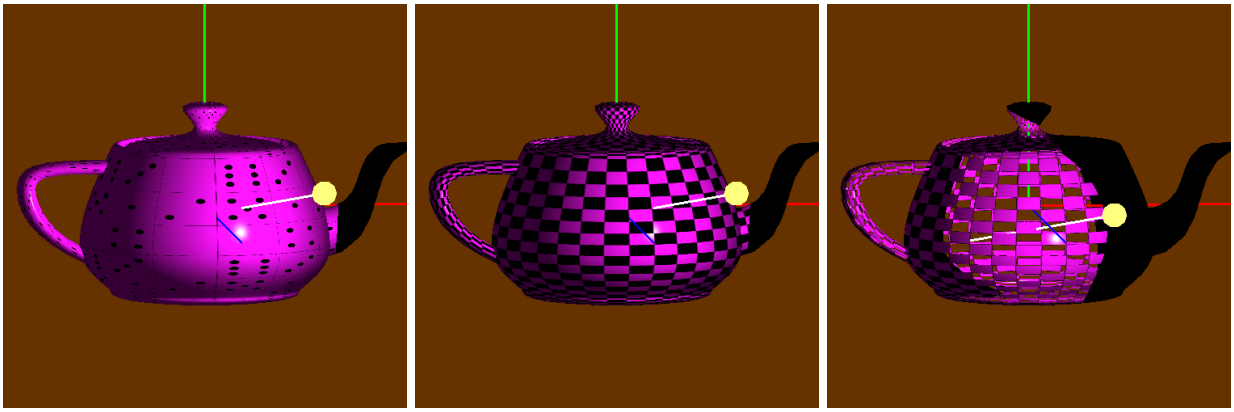

FIGURE 5 – La théière texturée (noir opaque ou noir transparent)

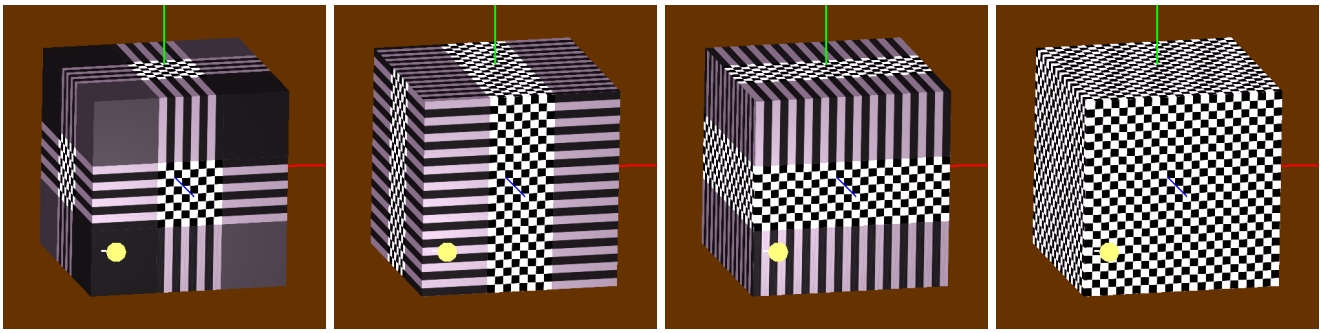

FIGURE 6 – Texture échiquier appliquée sur le cube avec différents modes de répétition

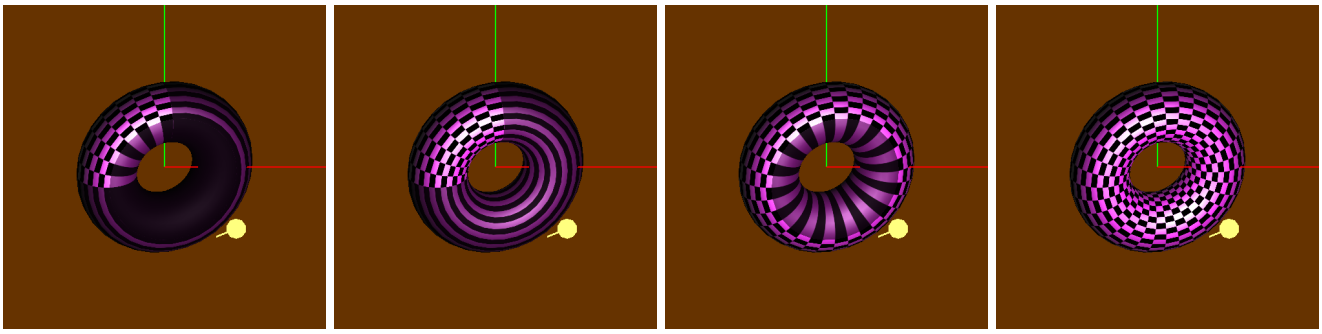

FIGURE 7 – Texture échiquier appliquée sur le tore avec différents modes de répétition (avec un plus grand coefficient de brillance)

3 Exigences

3.1 Exigences fonctionnelles

Partie 1 :

E1. Le modèle d'illumination de Phong est correctement implanté.

E2. Le modèle d'illumination de Gouraud est correctement implanté.

E3. Le modèle d'illumination de Lambert est correctement implanté.

E4. Les modèles de réflexion de Phong et de Blinn sont correctement implémentés.

Partie 2 :

E5. Le modèles de spot d'OpenGL est correctement implémenté.

E6. Le modèles de spot de Direct3D est correctement implémenté.

E7. Les modifications des propriétés du spot sont visibles (ex. : position, orientation, taille du cône).

Partie 3 :

E8. Les paramètres des textures sont bien initialisés et les objets texturés sont correctement illuminés.

E9. Le cube est affichée correctement avec la texture du dé (voir Figure 4).

E10. Le cube et le tore sont affichées correctement avec la texture de l'échiquier (voir Figure 6).

E11. L'utilisateur peut changer de texture (entre dé et échiquier) ainsi que des quatre modes de répétition de la texture (ex. : Figure 7).

E12. Permettre l'affichage de l'objet texturé avec les pixels noirs devenus transparents lorsqu'on utilise les nuanceurs (ex. : Figures 4 et 5).

3.2 Exigences non fonctionnelles

Normalement, on chargerait des nuanceurs différents pour cas d'utilisation afin d'augmenter la performance en évitant les énoncés conditionnels à la valeur d'une variable uniforme.

Toutefois, dans le contexte de TP, on utilisera sciemment de tels énoncés conditionnels aux valeurs des variables uniformes (modifiables interactivement) : le modèle d'illumination, le modèle de spot, si on veut des pixels transparents, etc. Ceci permettra de plus facilement contrôler le type de rendu, en plus de faciliter votre développement... et la correction !

Dans vos nuanceurs, on voudra donc voir des énoncés semblables à ceux-ci :

```
if ( variableUniforme == ... ) ... else ... ;

( variableUniforme == ... ) ? ... : ...
```

ANNEXES

A Liste des commandes

Touche	**Description**
q	Quitter l'application
x	Activer/désactiver l'affichage des axes
v	Recharger les fichiers des nuanceurs et recréer le programme
p	Permuter la projection : perspective ou orthogonale
i	Alterner entre le modèle d'illumination : Lambert, Gouraud, Phong
r	Alterner entre le modèle de réflexion spéculaire : Phong, Blinn
s	Alterner entre le modèle de spot : OpenGL, Direct3D
l	Alterner entre une caméra locale à la scène ou distante (localViewer)
a	Incrémenter l'angle du cône du spot
z	Décrémenter l'angle du cône du spot
d	Incrémenter l'exposant du spot
e	Décrémenter l'exposant du spot
j	Incrémenter le coefficient de brillance
u	Décrémenter le coefficient de brillance
m	Choisir le modèle affiché : cube, théière, tore, sphère, dodécaèdre, icosahèdre
t	Choisir la texture utilisée : aucune, dé, échiquier
w	Changer le mode de répétition de la texture
c	Changer l'affichage de l'objet texturé avec couleurs ou sans couleur
o	Permuter la transparence des pixels noirs
g	Permuter l'affichage en fil de fer ou plein
n	Utiliser ou non les normales calculées comme couleur (pour le débogage)
SPACE	Permuter la rotation automatique du modèle
BUTTON LEFT	Tourner l'objet
BUTTON MIDDLE	Modifier l'orientation du spot
BUTTON RIGHT	Déplacer la lumière

B Figures supplémentaires

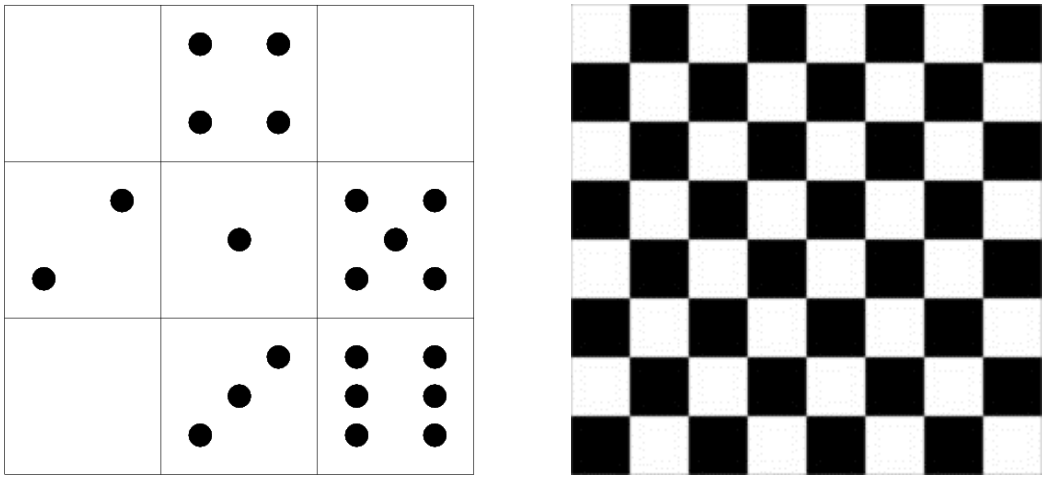

FIGURE 8 – Les textures fournies

C Apprentissage supplémentaire

Partie 1 :

1. Valider les réflexions individuelles de chaque composante de la lumière (ambiante, diffuse, spéculaire) en modifiant le nuanceur.
2. Comparer visuellement les différences entre les modèles de Phong et Blinn.
3. Comment implanteriez-vous mieux le modèle d'illumination de Lambert ?
4. Afficher une sphère au lieu du cube. (Vous devrez alors calculer et spécifier ses normales.)

Partie 2 :

5. Définissez un autre modèle de spot (d'autres paramètres et fonction de calcul) qui vous semblerait intéressant.
6. Plutôt que de mettre en noir les fragments non directement éclairés par le spot, diminuez leur intensité par un facteur de 2.

Partie 3 :

7. Modifier les coordonnées de texture pour constater l'effet sur le résultat visuel.
8. Faire glisser la texture affichée en fonction du temps.
9. Définir les bonnes coordonnées de texture pour un objet plus complexe composé de triangles.

D Formules utilisées

D.1 Modèles de spot d'OpenGL et de Direct3D

Un spot n'éclaire qu'à l'intérieur d'un cône, c'est-à-dire a une influence seulement si l'angle γ entre la direction du spot et la direction vers le point à éclairer est plus petit que l'angle d'ouverture δ du spot, c'est-à-dire si $cos(\gamma) > cos(\delta)$. La différence entre les modèles d'OpenGL et de Direct3D réside dans la formule pour calculer le facteur qui multiplie l'intensité lumineuse du spot à l'intérieur du cône :

- OpenGL utilise le facteur : $(cos(\gamma))^c$

- Direct3D utilise le facteur : $(cos(\gamma) - cos(\theta_{outer}))/(cos(\theta_{inner}) - cos(\theta_{outer}))$

où :

$cos(\delta)$:	cosinus de l'angle d'ouverture	= `cos(LightSource[0].spotCutoff)`
$\vec{L_n}$:	direction du spot	= `LightSource[0].spotDirection`
c :	exposant du spot	= `LightSource[0].spotExponent`
$cos(\gamma)$:	est obtenue par	$(\vec{L} \cdot \vec{L_n})$
$cos(\theta_{inner})$:	est remplacé *dans ce TP* par	$cos(\delta)$
$cos(\theta_{outer})$:	est remplacé *dans ce TP* par	$(cos(\delta))^{1.01+c/2}$

D.2 Modèles de réflexion spéculaire de Phong et de Blinn

Le calcul de la réflexion spéculaire fait intervenir un produit scalaire entre deux vecteurs. La différence entre les modèles de Phong et de Blinn réside dans le choix des deux vecteurs utilisés :

- Phong utilise : $\vec{R} \cdot \vec{O} = \texttt{reflect}(-\vec{L}, \vec{N}) \cdot \vec{O}$

- Blinn utilise : $\vec{B} \cdot \vec{N} = bissectrice(\vec{L}, \vec{O}) \cdot \vec{N}$

où :

$\vec{N}$:	normale à la surface	
$\vec{L}$:	direction du point vers la source lumineuse	
$\vec{R}$:	direction du rayon réfléchi	= `reflect(-L,N)`
$\vec{O}$:	direction du point vers l'observateur	
$\vec{B}$:	bissectrice entre les vecteurs $\vec{L}$ et $\vec{O}$	= `normalise(L+O)`, si $\vec{L}$ et $\vec{O}$ sont unitaires.