



POLYTECHNIQUE
MONTRÉAL

LE GÉNIE
EN PREMIÈRE CLASSE

INF2705 Infographie

Spécification des requis du système

Travail pratique 3

Éclairage de scène en trois couleurs

Département de génie informatique et génie logiciel
Polytechnique Montréal
Benoît Ozell, Automne 2021

Table des matières

1	Introduction	2
1.1	But	2
1.2	Portée	2
1.3	Remise	2
2	Description globale	3
2.1	But	3
2.2	Travail demandé	3
3	Exigences	10
3.1	Exigences fonctionnelles	10
3.2	Exigences non fonctionnelles	10
A	Liste des commandes	11
B	Textures utilisées	12
C	Apprentissage supplémentaire	13
D	Formules utilisées	14
D.1	Modèles de réflexion spéculaire de Phong et de Blinn	14
D.2	Placage de relief	15
D.3	Modèles de spot inspirés d'OpenGL ou de Direct3D	16

1 Introduction

Ce document décrit les exigences du TP3 « *Éclairage de scène en trois couleurs* » (Automne 2021) du cours INF2705 Infographie.

1.1 But

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

1.2 Portée

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

1.3 Remise

Faites la commande « `make remise` » ou exécutez/cliquez sur « `remise.bat` » afin de créer l'archive « **INF2705_remise_TPn.zip** » (ou .7z, .rar, .tar) que vous déposerez ensuite dans Moodle. (Moodle ajoute automatiquement vos matricules ou le numéro de votre groupe au nom du fichier remis.)

Ce fichier `zip` contient tout le code source du TP (`makefile`, `*.h`, `*.cpp`, `*.glsl`, `*.txt`).

2 Description globale

2.1 But

Le but de ce TP est de permettre à l'étudiant mettre en pratique l'illumination des objets, l'application de textures pour colorer ou créer l'illusion d'un relief en utilisant des nuanceurs en GLSL. Ce travail pratique permet aussi de se familiariser avec les nuanceurs de tessellation.

2.2 Travail demandé

Partie 1 : réflexion de la lumière sur les objets

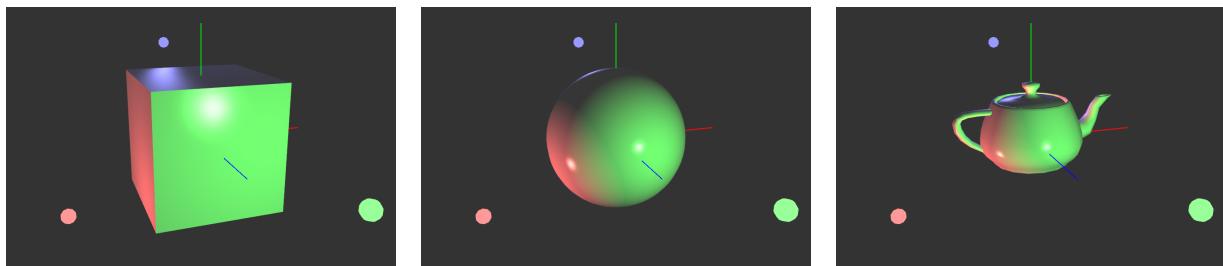


FIGURE 1 – Divers objets illuminés.

Vous calculerez les réflexions sur divers objets avec trois sources de lumière. L'effet de ces trois sources de lumière positionnelles est cumulatif et augmente les réflexions aux endroits éclairés par plusieurs sources (figure 1). Pour tous les objets, les modèles d'illumination de Gouraud et de Phong (figure 2) seront mis en œuvre et on pourra aussi choisir, pour le calcul de la réflexion spéculaire, entre le modèle de Phong ou celui de Blinn (figure 3).

Pour le cube, vous devrez définir directement les normales de chaque sommet selon la face à tracer : $(0,+1,0)$, ... $(0,0,-1)$, ... $(+1,0,0)$, etc. L'utilisation d'un facteur d'atténuation n'est pas un requis, mais libre à vous d'expérimenter !

Pour démarrer, on pourra utiliser les nuanceurs des exemples du cours :

<https://gitlab.com/ozell/inf2705-exemples/-/tree/master/06-IlluminationTheieresPhong/> et
<https://gitlab.com/ozell/inf2705-exemples/-/tree/master/06-IlluminationTheieresGouraud/..>

Vous observerez que l'illumination de Gouraud sur le cube montre bien les limites de cette technique. Les couleurs sont calculées aux sommets de chaque face du cube et interpolées sur toute la face à partir de ces valeurs aux coins. Mais puisqu'il n'y a que deux triangles par face, la réflexion spéculaire n'est jamais très « belle », voire même inexiste : ça dépend s'il y a ou non de la réflexion spéculaire aux coins du cube. Et s'il y a réflexion spéculaire au coin, alors l'intensité est alors simplement interpolée linéairement sur la face (figure 4). Ce problème de rendu dû à la faible résolution des faces sur le cube sera résolu, à la partie 3 du TP, par l'utilisation de nuanceurs de tessellation afin de subdiviser les faces du cube (aperçu à la figure 10). En attendant d'utiliser la tessellation, on acceptera ce défaut d'affichage sur le cube avec l'illumination de Gouraud.

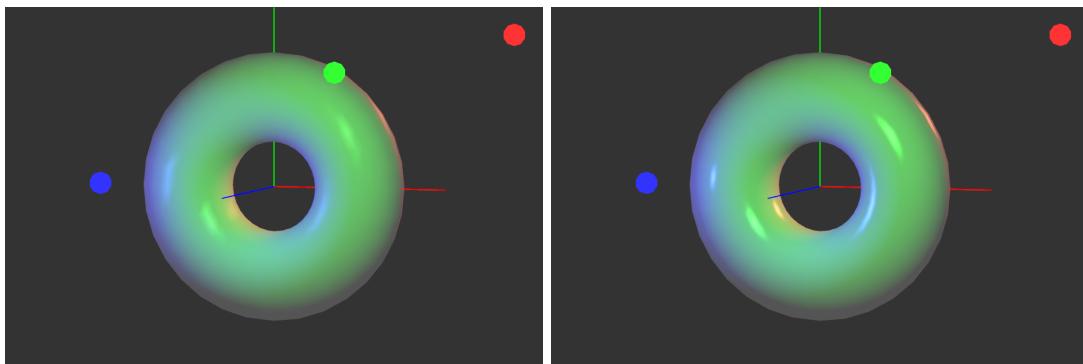


FIGURE 2 – Rendu avec réflexion spéculaire de Phong : illumination de a) Gouraud, b) Phong.

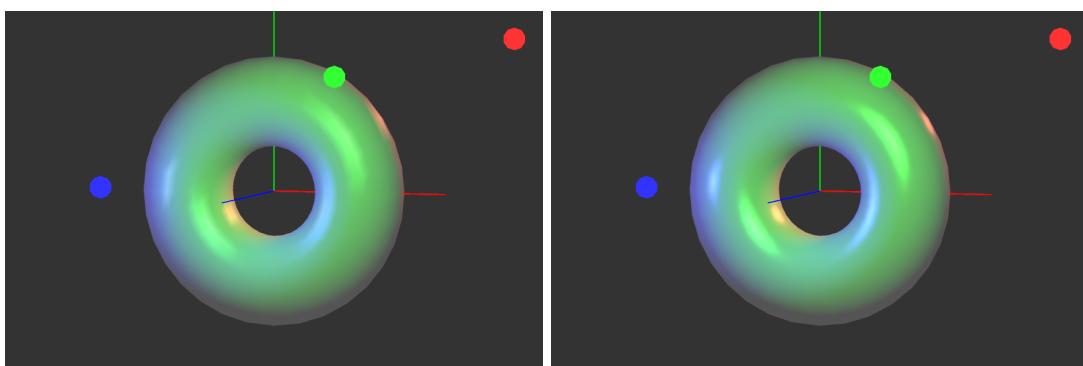


FIGURE 3 – Rendu avec réflexion spéculaire de Blinn : illumination de a) Gouraud, b) Phong.

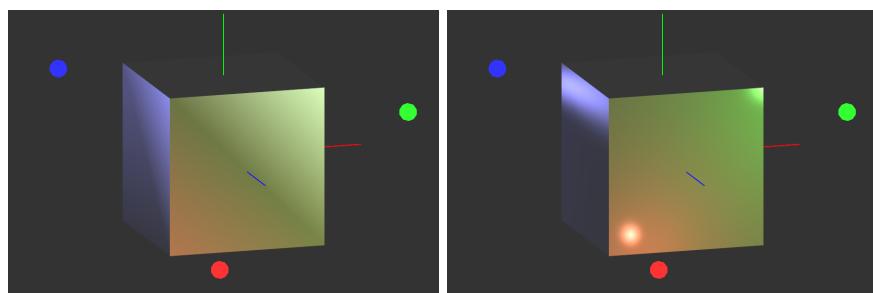


FIGURE 4 – Réflexion spéculaire a) pas très réussie avec Gouraud et b) mieux avec Phong.

Note : Ce TP utilise des « *Uniform Buffer Object* » (UBO) afin de transférer en bloc les variables uniformes aux nuanceurs. L’usage des UBO est très semblable aux VBO et permet surtout que le passage des valeurs des variables uniformes aux nuanceurs soit beaucoup plus efficace. Dans ce TP, on utilise ainsi quatre UBO qui correspondent aux quatre blocs de variables uniformes utilisés dans les nuanceurs : LightSource, FrontMaterial, LightModel et varsUnif. Les trois premiers blocs contiennent les variables uniformes servant à l’illumination, tandis que le dernier bloc contient les variables uniformes de l’application. (Les noms de variable sont inspirés de OpenGL 2.x.)

Partie 2 : l'application de textures

Le logiciel permettra d'afficher les textures fournies (voir figures 16 et 17) sur les objets illuminés.

- Une carte du monde sur le cube : on spécifiera les coordonnées de texture afin de montrer diverses parties du monde sur les faces. La texture fournie contient une mappemonde et il s'agit donc de définir les bonnes coordonnées de texture afin d'afficher sur toutes les faces : le Monde répété 3 fois sur le dessus (+Y), l'Amérique du Sud, l'Afrique + Moyen-Orient, le Québec, l'Europe (incluant pour le moment encore l'Angleterre !) sur les côtés et l'Australie sur le dessous (-Y) comme montré à la figure figure 5.
- Les autres textures sur le cube : on spécifiera les coordonnées de texture afin de montrer la texture entière directement sur chaque face comme montré à la figure 5.

Les autres objets (tore, sphère, théière, cylindre) définissent eux-mêmes leurs coordonnées de texture et l'affichage des textures devrait se faire pratiquement automatiquement (figure 6) ; c'est donc un bon point de départ.

Pour démarrer, on pourra utiliser les nuanceurs des exemples du cours :

[https://gitlab.com/ozell/inf2705-exemples/-/tree/master/08-Texture/..](https://gitlab.com/ozell/inf2705-exemples/-/tree/master/08-Texture/)

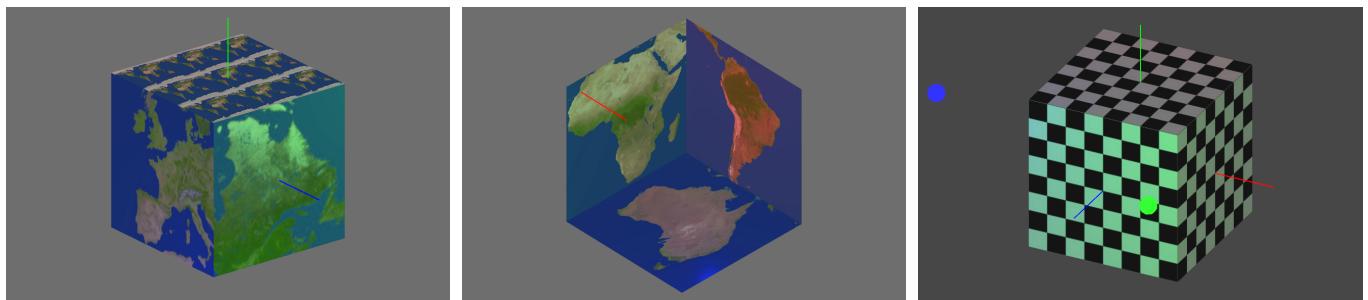


FIGURE 5 – Textures sur le cube

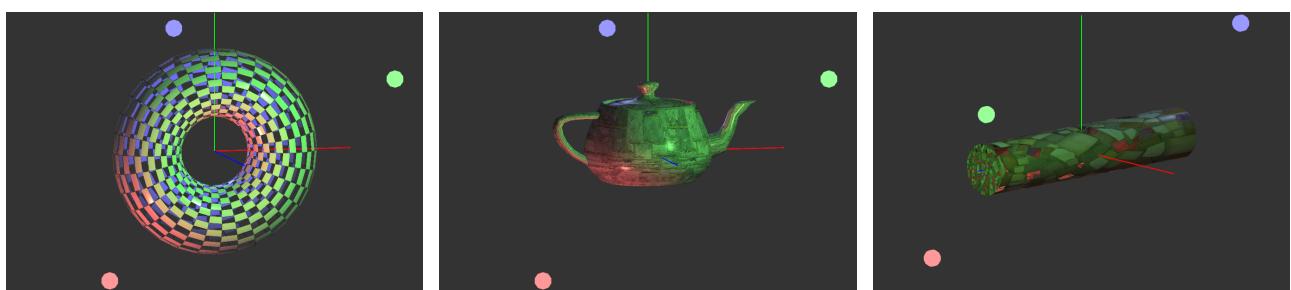


FIGURE 6 – Diverses textures sur les autres objets

On voudra aussi que la texture se « déplace » sur l'objet en fonction du temps, comme si elle glissait sur la surface dans une direction donnée. La variable d'état « tempsGlissement », communiquée aux nuanceurs dans les variables uniformes, servira à décaler une des coordonnées de texture avant d'aller chercher le texel dans la texture (figure 7). Prenez aussi soin que ce déplacement que vous ajoutez fasse en sorte que la texture de la Terre, en particulier, se déplace dans le bon sens sur la sphère : le soleil se lève à l'est et se couche à l'ouest !

Lorsqu'on presse la touche 'a' pour avancer le temps (voir annexe A), l'effet visuel sera que la texture se déplace sur la surface dans la direction du décalage.

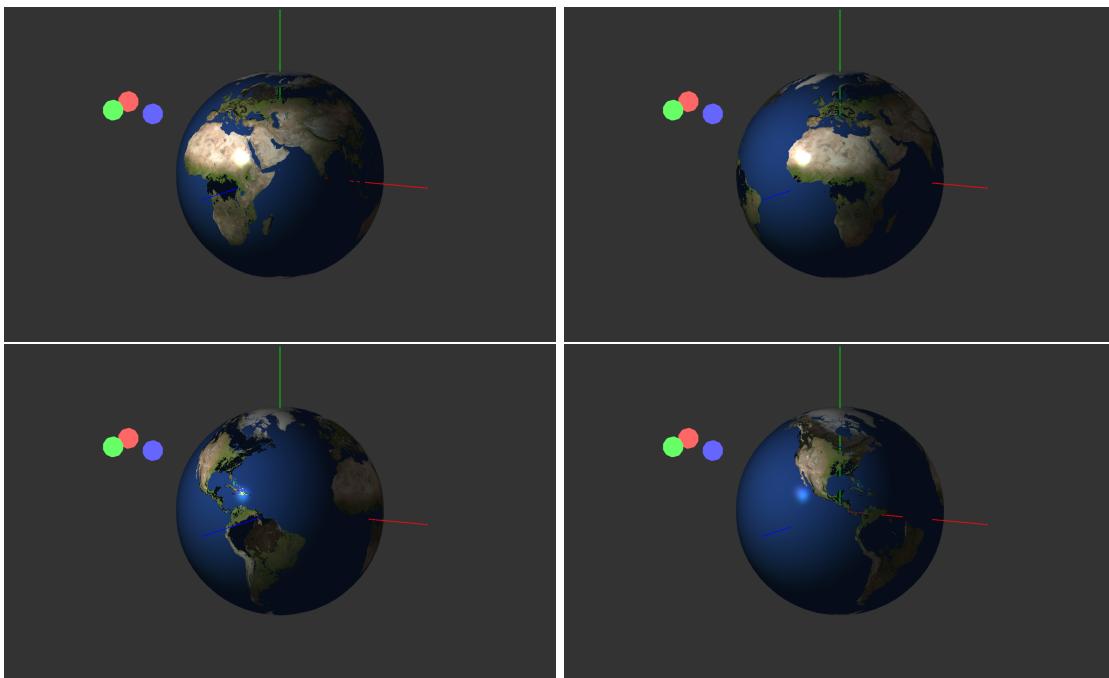


FIGURE 7 – Décalage en fonction du temps de la texture Terre sur la sphère

Enfin, pour jouer un peu plus dans le nuancier de fragments, lorsque le texel est bien foncé (lorsque « `length(couleurTexture.rgb) < 0.5` »), il sera plutôt complètement transparent (utiliser `discard`), tel qu'illustré à la figure 8.

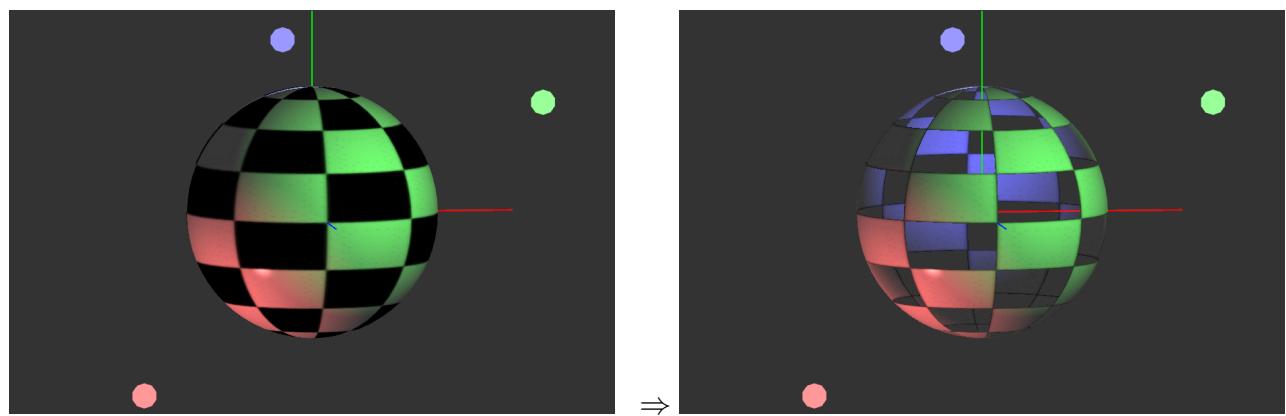


FIGURE 8 – Un texel foncé est plutôt affiché transparent

Partie 3a : l'utilisation des nuanceurs de tessellation

On aura remarqué que le modèle de Gouraud n'offre pas un très beau rendu graphique, en comparaison de celui de Phong, lorsque la surface du cube n'est composée que seulement de deux triangles (figure 9). Afin de corriger cet effet, on utilisera des nuanceurs de tessellation pour subdiviser chaque face du cube et fournir un plus beau rendu lorsque l'algorithme de Gouraud est utilisé. Il faudra d'abord activer les nuanceurs de tessellation dans le programme principal et ensuite utiliser des GL_PATCHES afin d'afficher les faces du cube qui seront alors subdivisées en un nombre variable de sous-triangles (figures 10 et 11). (Afin d'améliorer le rendu avec l'algorithme de Gouraud, il faudra bien sûr alors déplacer le calcul de l'éclairage dans les nuanceurs de tessellation.)

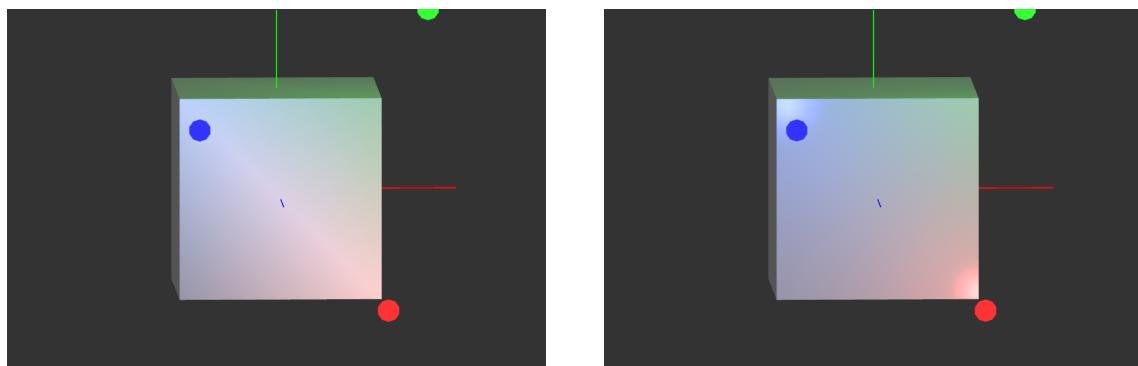


FIGURE 9 – Comparaison entre les rendus du modèle de Gouraud et du modèle de Phong

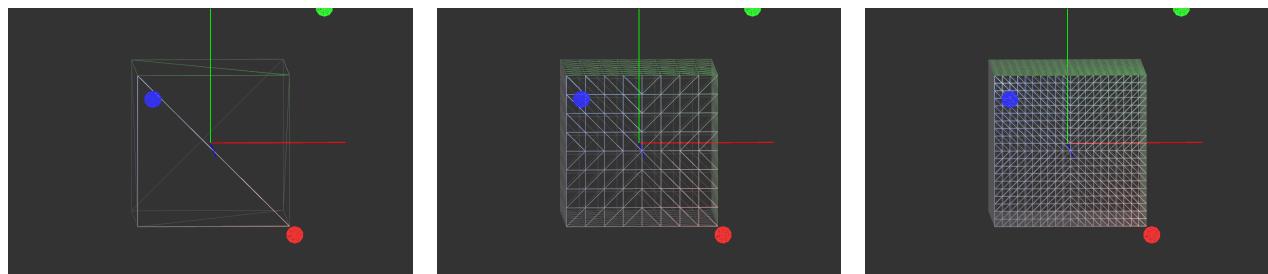


FIGURE 10 – La face du cube avec Gouraud et des niveaux de tessellation de 1, 8 ou 20 (fil de fer)

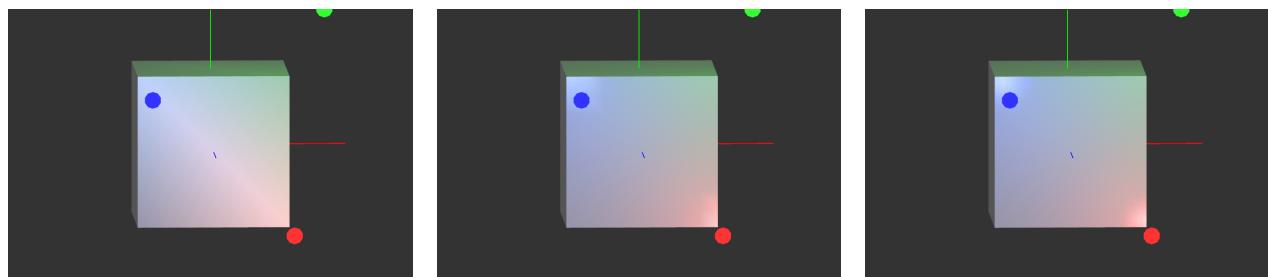


FIGURE 11 – La face du cube avec Gouraud et des niveaux de tessellation de 1, 8 ou 20 (plein)

Pour démarrer, on pourra utiliser les nuanceurs l'exemple de nuanceurs pour le quad :
<https://gitlab.com/ozell/inf2705-exemples/-/tree/master/09-TesselationPrim/>.

Partie 3b : le placage de relief

Le placage de relief permet de simuler la variation des normales à la surface en utilisant une texture spéciale qui contient des variations à appliquer aux normales calculées. Cet effet, applicable dans le nuanceur de fragments, permet ainsi de montrer un relief (simulé) à la surface de l'objet (figure 12).

En utilisant deux textures, on peut colorer un objet avec une texture pour les couleurs, puis simuler l'effet ajoutée d'un placage de relief avec une texture supplémentaire pour les normales (figure 13).

La section D.2 décrit une façon simplifiée (incorrecte mais simple !) d'appliquer un placage de relief pour ce TP.

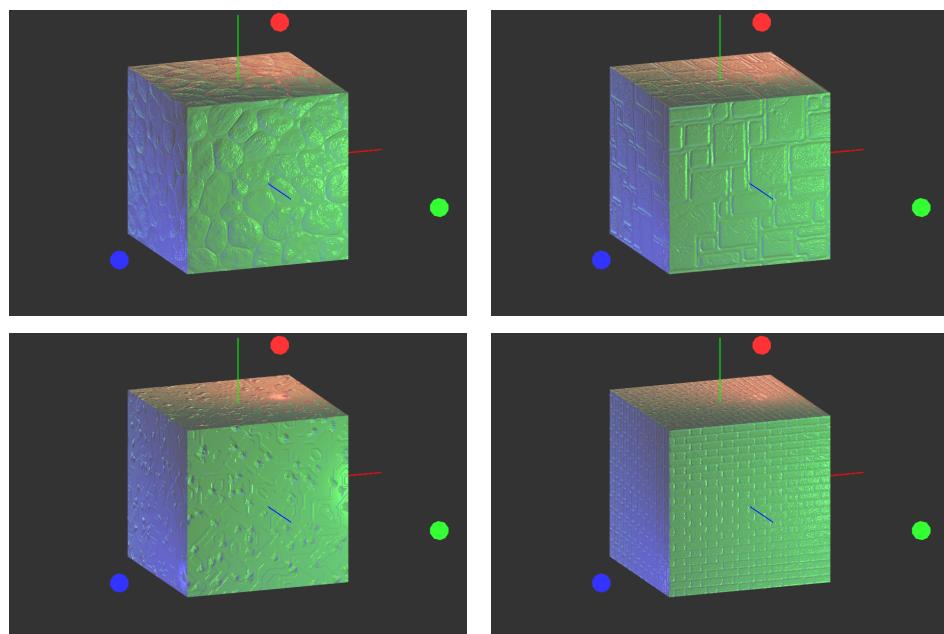


FIGURE 12 – Placage de relief seulement

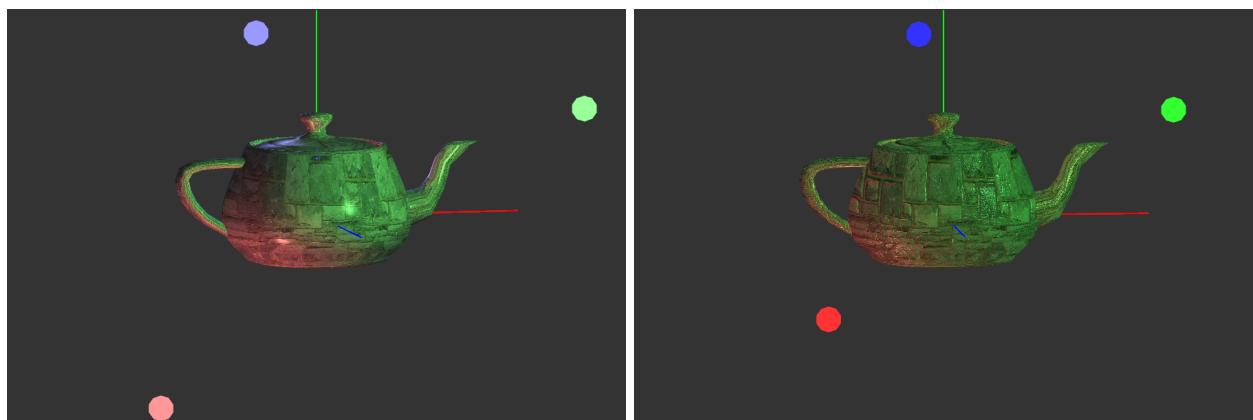


FIGURE 13 – Placage de texture, puis de texture et relief

Partie 3c : l'utilisation de spots

On peut aussi utiliser un éclairage de type « spot » où le cône de lumière est selon le style d'OpenGL ou de Direct3D (figures 14 et 15). En variant l'angle du cône de lumière, on peut illuminer certaines régions et en laisser d'autres dans le noir. La section D.3 décrit les formules applicables.

Note : afin d'afficher la ligne montrant la direction du spot, il faut changer le « #if 0 » par « #if 1 » dans la fonction `afficherLumieres()` du programme principal `main.cpp`.

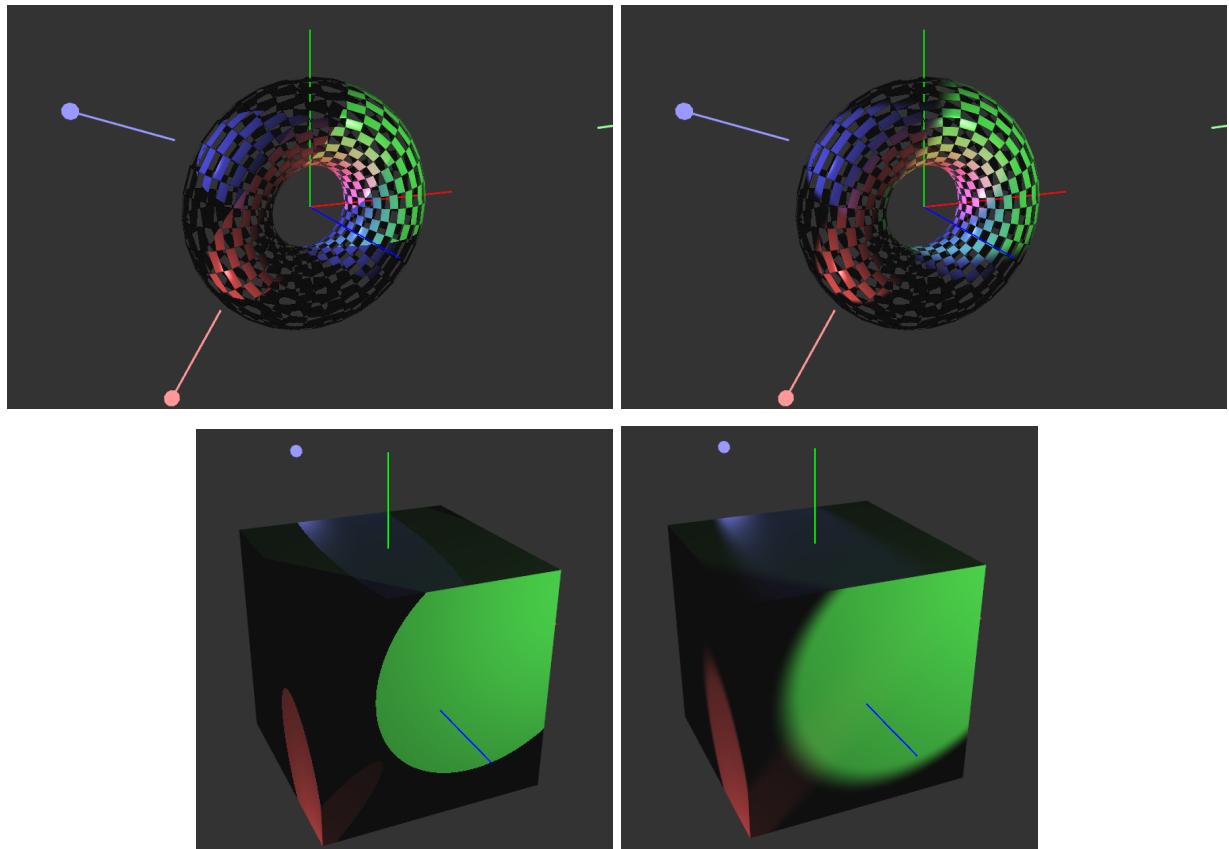


FIGURE 14 – Rendu de spots, avec le style OpenGL ou Direct3D, sur le tore et sur le cube

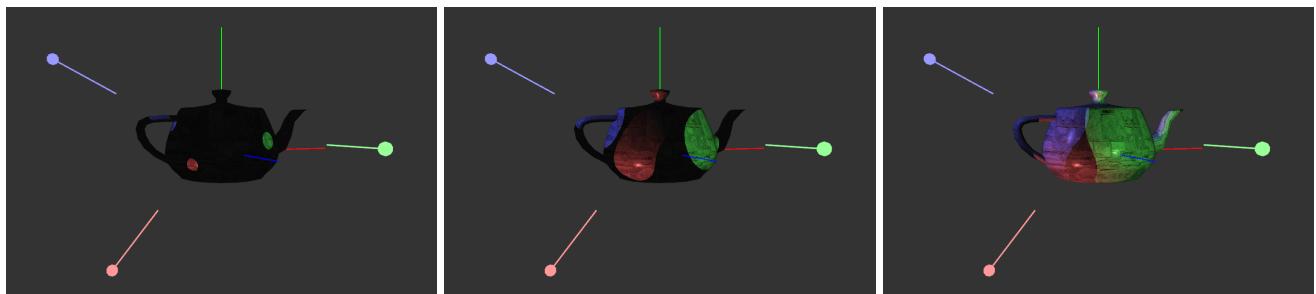


FIGURE 15 – Rendu avec des spots sur la théière un angle d'ouverture du cône de 3, 10 ou 20 degrés

Pour démarrer, on pourra utiliser les nuanceurs l'exemple de nuanceurs avec un spot :
<https://gitlab.com/ozell/inf2705-exemples/-/tree/master/06-IlluminationTheieresSpot/> .

3 Exigences

3.1 Exigences fonctionnelles

Partie 1 :

- E1. Le modèle d'illumination de Phong est implanté comme demandé. [6 pts]
- E2. Le modèle d'illumination de Gouraud est implanté comme demandé. [6 pts]
- E3. Les modèles de réflexion spéculaire de Phong et de Blinn sont bien implémentés. [2 pts]

Partie 2 :

- E4. La texture de la mappemonde est affichée correctement sur le cube (figure 5). [2 pts]
- E5. Les autres textures sont affichées correctement sur le cube (figure 5). [2 pts]
- E6. Les textures sont généralement bien affichées sur les autres objets (figure 6). [2 pts]
- E7. Tous les objets texturés sont correctement illuminés lorsque texturés. [3 pts]
- E8. Les textures se déplacent sur l'objet en fonction du temps. (figure 7). [4 pts]
- E9. Les texels foncés sur l'objet sont affichés transparents (figure 8). [2 pts]

Partie 3a :

- E10. Le rendu illuminé avec Gouraud sur le cube est corrigé avec l'utilisation des nuanceurs de tessellation (figure 11). [9 pts]

Partie 3b :

- E11. Une texture de relief peut être appliquée sur les objets (figure 12). [4 pts]
- E12. Deux textures peuvent être appliquées conjointement pour modifier couleurs et normales (figure 13). [5 pts]

Partie 3c :

- E13. Utilisation de spots selon le modèle inspiré de OpenGL ou de Direct3D (figure 14). [9 pts]

3.2 Exigences non fonctionnelles

De façon générale, le code que vous ajouterez sera de bonne qualité. Évitez les énoncés superflus (qui montrent que vous ne comprenez pas bien ce que vous faites !), les commentaires erronés ou simplement absents, les mauvaises indentations, etc. [2 pts]

Notez que, normalement, on chargerait des nuanceurs différents pour chaque 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). Ceci permettra de plus facilement contrôler le type de rendu, en plus de faciliter votre développement.

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

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

ou, mieux encore :

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

ANNEXES

A Liste des commandes

Touche	Description
q	Quitter l'application
x	Activer/désactiver l'affichage des axes
9	Permuter l'utilisation des nuanceurs de tessellation
v	Recharger les fichiers des nuanceurs et recréer le programme
p	Alterner entre le modèle d'illumination : Gouraud, Phong
r	Alterner entre le modèle de réflexion spéculaire : Phong, Blinn
d	Alterner entre le modèle de spot : OpenGL, Direct3D
h	Incrémenter le coefficient de brillance
b	Décrémenter le coefficient de brillance
c	Incrémenter l'angle d'ouverture du cône du spot
f	Décrémenter l'angle d'ouverture du cône du spot
w	Incrémenter l'exposant du spot
s	Décrémenter l'exposant du spot
m	Choisir le modèle affiché : cube, tore, sphère, théière, cylindre, cône
t	Choisir la texture de couleurs utilisée : aucune, Terre, échiquier, mur, métal, mosaique
e	Choisir la texture de normales utilisée : aucune, pierre, bulles, mur, brique, circuit
i	Augmenter le niveau de tessellation interne
k	Diminuer le niveau de tessellation interne
o	Augmenter le niveau de tessellation externe
l	Diminuer le niveau de tessellation externe
u	Augmenter les deux niveaux de tessellation
j	Diminuer les deux niveaux de tessellation
BARREOBlique	Permuter la projection : perspective ou orthogonale
g	Permuter l'affichage en fil de fer ou plein
n	Utiliser ou non les normales calculées comme couleur (pour le débogage)
0	Replacer Caméra et Lumière afin d'avoir une belle vue
z	Remettre le temps à 0
a	Le temps avance ou est à l'arrêt
ESPACE	Permuter la rotation automatique du modèle
BOUTON GAUCHE	Tourner l'objet
BOUTON MILIEU	Modifier l'orientation du spot
BOUTON DROIT	Déplacer la lumière
Molette	Changer la taille du spot

B Textures utilisées

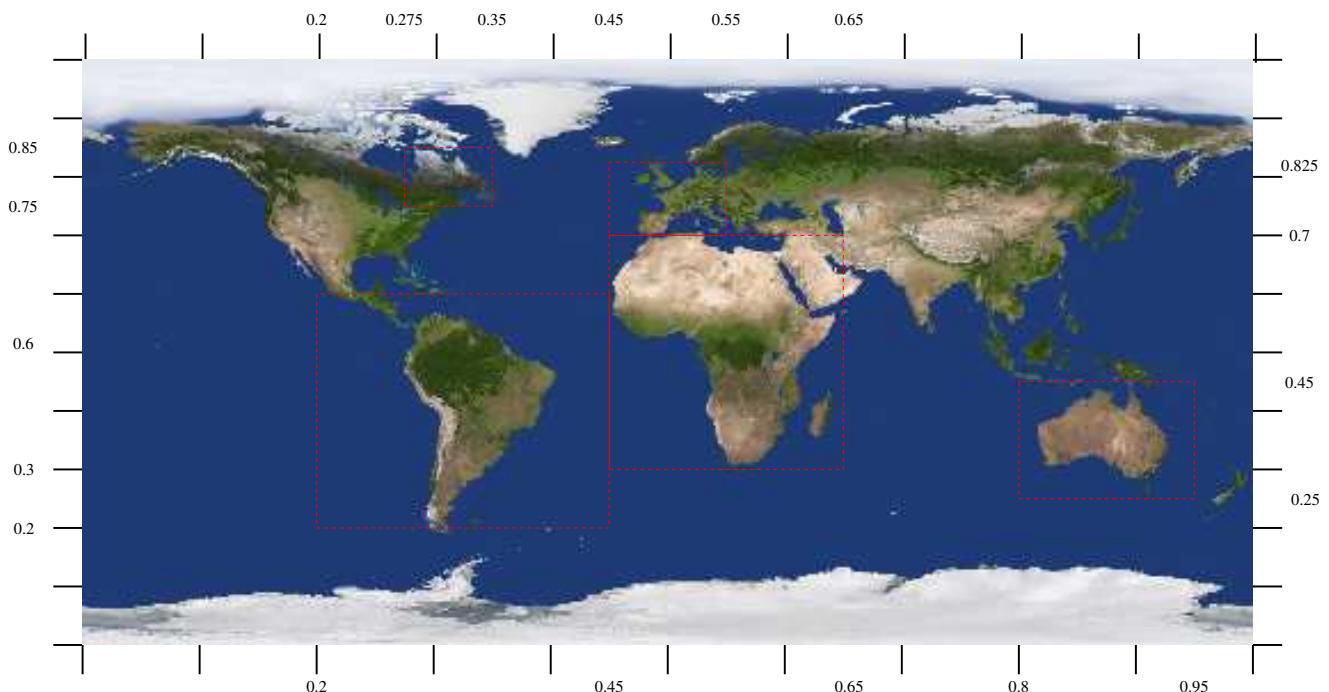


FIGURE 16 – La texture de la Terre

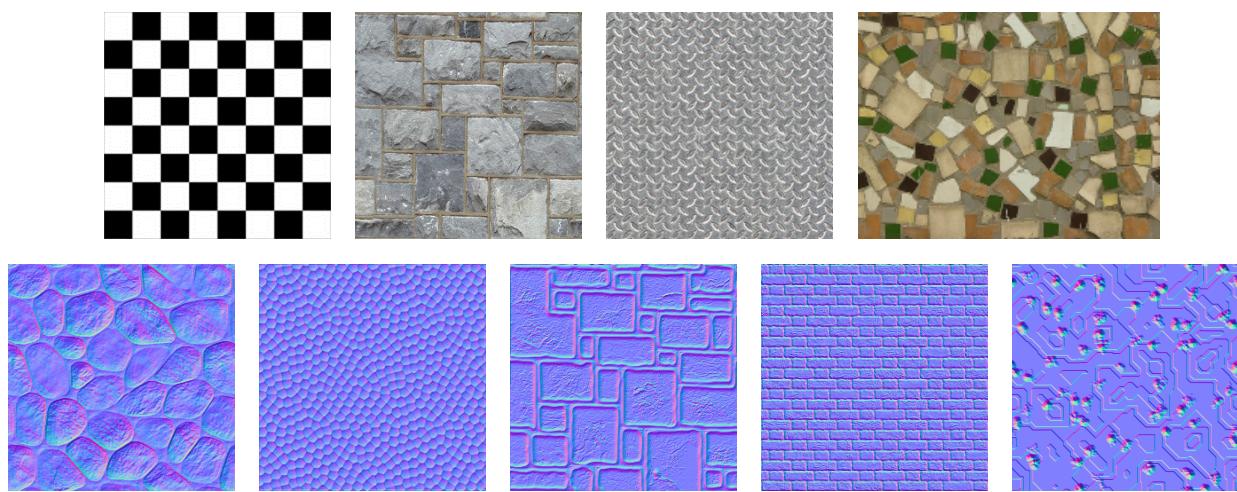


FIGURE 17 – Les autres textures fournies

C Apprentissage supplémentaire

Partie 1 :

1. Valider les réflexions individuelles de chaque composante de la lumière (ambiante, diffuse, spéculaire) en commentant sélectivement les lignes qui calculent la contribution de chaque composante dans le nuanceur.
2. Comparer visuellement les différences entre les modèles de Phong et Blinn, en particulier pour une lumière rasante.
3. Comment ferez-vous pour implanter le modèle d'illumination de Lambert ?

Partie 2 :

4. Modifier les coordonnées de texture pour constater l'effet sur le résultat visuel.
5. Définir les bonnes coordonnées de texture pour un objet plus complexe composé de triangles.
6. Utiliser deux textures pour appliquer leur combinaison sur l'objet.

Partie 3a :

7. Rapetisser légèrement chaque triangle afin de voir à l'intérieur du cube entre les triangles.
8. Déformer la surface du cube différemment, par exemple pour la convertir en ellipsoïde.
9. Utiliser un facteur de déformation `facteurDeform` pour mettre en œuvre une déformation variable.

Partie 3b :

10. Que se passe-t-il lorsque vous déplacez la source lumineuse à l'intérieur de l'objet ?
11. Plutôt que de mettre en noir les fragments non directement éclairés par la source de lumière, diminuez simplement leur intensité par un facteur de 2.

Partie 3c :

12. Définissez vos propres modèles de spot (autres paramètres et fonction de calcul) qui vous semblent intéressants.

D Formules utilisées

D.1 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 : $\vec{R} \cdot \vec{O} = \text{reflect}(-\vec{L}, \vec{N}) \cdot \vec{O}$
- Blinn utilise : $\vec{B} \cdot \vec{N} = \text{bissectrice}(\vec{L} \text{ et } \vec{O}) \cdot \vec{N} = \text{normalise}(\vec{L} + \vec{O}) \cdot \vec{N}$

où :

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

Tous les calculs d'illumination se font dans le repère de la caméra en GLSL.

- Le calcul de la direction vers l'observateur (\vec{O}) :
- (un vecteur qui pointe vers le (0,0,0), c'est-à-dire vers la caméra)

```
AttribsOut.obsVec = (-pos); // =(0-pos)
```

- La direction de la lumière (\vec{L}) :

```
AttribsOut.lumiDir[j] = (matrVisu * LightSource.position[j]).xyz - pos;
```

D.2 Placage de relief

Afin de produire un effet de relief, on doit modifier la normale N utilisée pour l'illumination selon le contenu d'une texture un peu spéciale qui encode les variations de la normale dans $[-1,+1]$ sous la forme de couleurs qui varient dans $[0,1]$ stockées dans la texture.

Pour récupérer la variation recherchée de la normale dN , on doit obtenir la « couleur » dans la texture des normales à la position donnée par `AttribsIn.texCoord`, puis convertir la valeur RGB lue :

```
vec3 couleur = texture( ... ).rgb;
vec3 dN = normalize( ( couleur - 0.5 ) * 2.0 );
```

Ensuite, une façon simple (bien qu'incorrecte) est de produire une nouvelle normale N utilisant la variation dN :

```
N = normalize( N + dN ).
```

Dans ce TP, nous accepterons cette simplification qui évite d'avoir à calculer aussi la tangente et la binormale à chaque point. Dans le nuanceur de fragments, un simple test pour savoir si une texture de normale est activée est de vérifier la valeur de `iTexNorm` et de modifier la normale N en conséquence :

```
if ( iTexNorm != 0 ) N = modifierNormale( N );
```

D.3 Modèles de spot inspirés d'OpenGL ou 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. Lorsque c'est le cas, on a « $\gamma < \delta$ » et mais on vérifiera plutôt si « $\cos(\gamma) > \cos(\delta)$ » en évaluant des *produits scalaires* entre les vecteurs appropriés.

Pour déterminer la direction du spot dans le repère de la caméra, on peut calculer ce vecteur direction directement dans le nuanceur :

```
AttribsOut.spotDir[j] = mat3(matrVisu) * -LightSource.spotDirection[j];
```

La différence entre les modèles inspirés d'OpenGL et de Direct3D que nous utiliserons réside dans la formule pour calculer le facteur qui multiplie l'intensité lumineuse du spot à l'intérieur du cône :

- OpenGL utilise le facteur : $\text{fact} = (\cos(\gamma))^c$
- Direct3D utilise le facteur : $\text{fact} = \text{smoothstep}(\cos(\theta_{\text{outer}}), \cos(\theta_{\text{inner}}), \cos(\gamma))$

où :

$\cos(\delta)$:	cosinus de l'angle d'ouverture	= $\cos(\text{LightSource.spotAngleOuverture})$
\vec{L}_n :	direction du spot	= $\text{LightSource.spotDirection}[j]$
c :	exposant du spot	= $\text{LightSource.spotExponent}$
$\cos(\gamma)$:	est obtenue par	$(\vec{L} \cdot \vec{L}_n)$
$\cos(\theta_{\text{inner}})$:	est remplacé <i>dans ce TP</i> par	$\cos(\delta)$
$\cos(\theta_{\text{outer}})$:	est remplacé <i>dans ce TP</i> par	$(\cos(\delta))^{1.01+c/2}$