POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

INF2705 Infographie

Spécification des requis du système

Travail pratique 3

Illumination et textures

Département de génie informatique et génie logiciel
École polytechnique de Montréal
Hiver 2017

Table des matières

1 Introduction

Ce document décrit les exigences fonctionnelles et non fonctionnelles du TP3 « *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 Remise

Vous remettrez un fichier `zip` contenant tout le code source du TP et le rapport (`*.cpp`, `*.h`, `*.glsl`, `makefile`, `*.txt`).

Faites « `make remise` » pour créer l'archive « `remise.zip` ».
Vous déposerez ensuite ce fichier dans Moodle.

2 Description globale

2.1 But

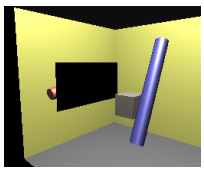
Le but de ce TP est de permettre à l'étudiant mettre en pratique les notions d'illumination et d'applications de textures en utilisant des nuanceurs en GLSL.

2.2 Travail demandé

Partie 1 : l'illumination des objets

On demande d'implanter les trois modèles d'illumination de Lambert, de Gouraud et de Phong sur divers objets (Figure 1). De plus, on pourra aussi choisir, pour le calcul de la réflexion spéculaire, 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. Normalement, le programme sur CPU spécifiera alors les normales par facette plutôt qu'à chaque sommet. On peut toutefois utiliser le nuanceur de géométrie pour calculer une nouvelle normale à la surface en utilisant un produit vectoriel de vecteurs construits à partir des sommets. Ce nouveau vecteur normal sera alors imposé à tous les sommets (et remplacera ainsi les normales spécifiées par le programme principal). Le nuanceur de fragments fera peut-être alors un peu trop de calculs inutiles, mais le but est surtout de se familiariser avec le nuanceur de géométrie.



Pour démarrer, on pourra utiliser les nuanceurs des exemples du cours :
`www.groupes.polymtl.ca/inf2705/exemples/06-IlluminationMiroir/` et
`www.groupes.polymtl.ca/inf2705/exemples/06-Illumination/`

Note : Ce TP utilise des « *Uniform Buffer Object* » (UBO) afin de transférer en bloc les variables uniformes aux nuanceurs. Le passage des valeurs des variables uniformes est ainsi beaucoup plus efficace. Même ces objets OpenGL n'ont pas été assez étudiés en classe, leur usage est très semblable aux VBO. 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 d'état de l'application. (Les anciennes `struct` de OpenGL 2.x deviennent ainsi des blocs de variables uniformes en OpenGL 4.x.)

Partie 2 : l'utilisation d'un spot

Le programme permettra aussi d'utiliser un éclairage de type « spot » selon une définition semblable à celle d'OpenGL ou à celle de Direct3D (Figure 3). Les propriétés du spot, sa position, son angle maximum (`spotCutoff`) et son exposant (`spotExponent`) sont modifiables en cours de l'exécution et seront correctement utilisés. La section 7 décrit les formules applicables.

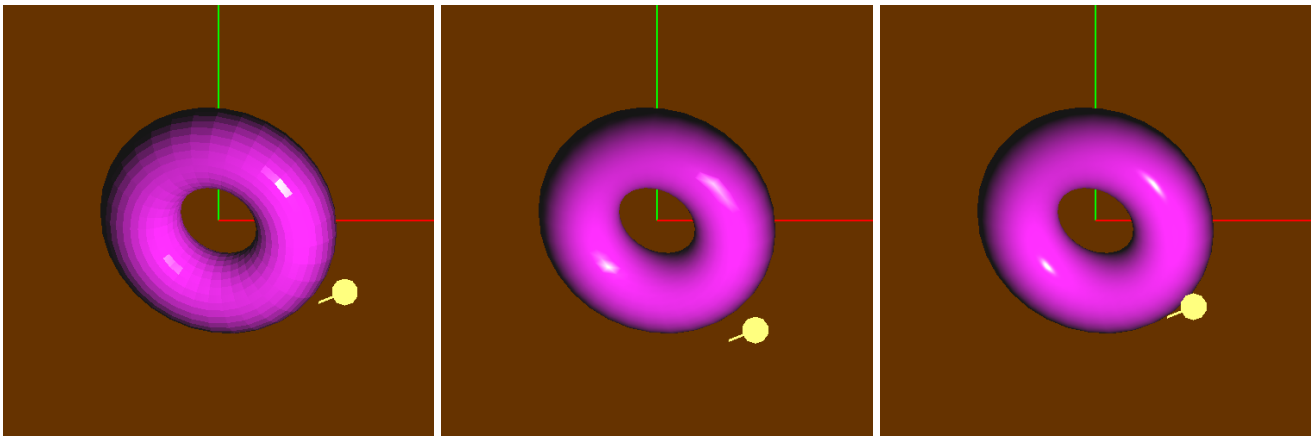


FIGURE 1 – Rendu avec illumination i) de Lambert, ii) de Gouraud, iii) de Phong. (Réflexion spéculaire de Phong dans tous les cas.)

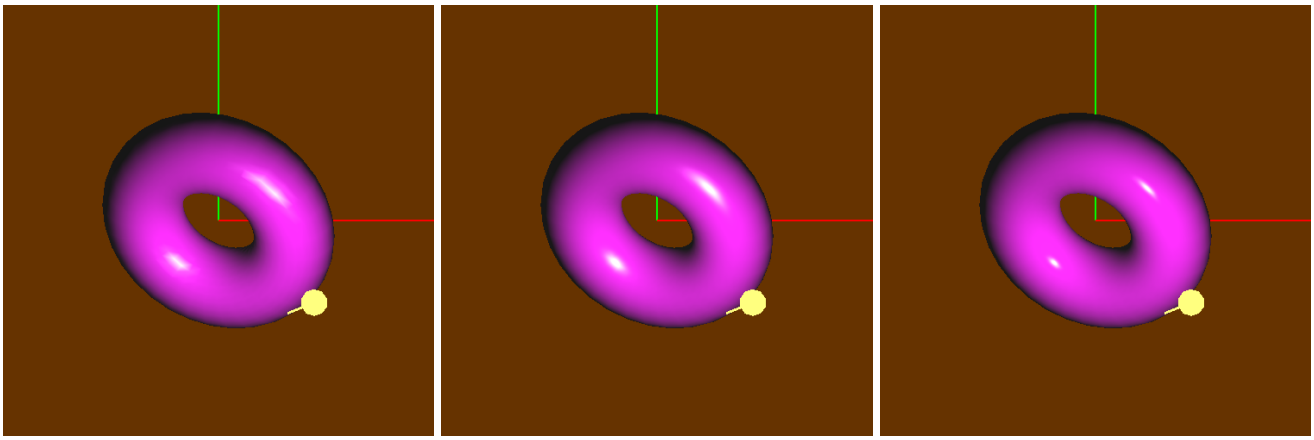


FIGURE 2 – Rendu avec i) illumination de Gouraud et réflexion spéculaire de Blinn, ii) illumination de Phong et réflexion spéculaire de Blinn, iii) illumination de Phong et réflexion spéculaire de Phong.

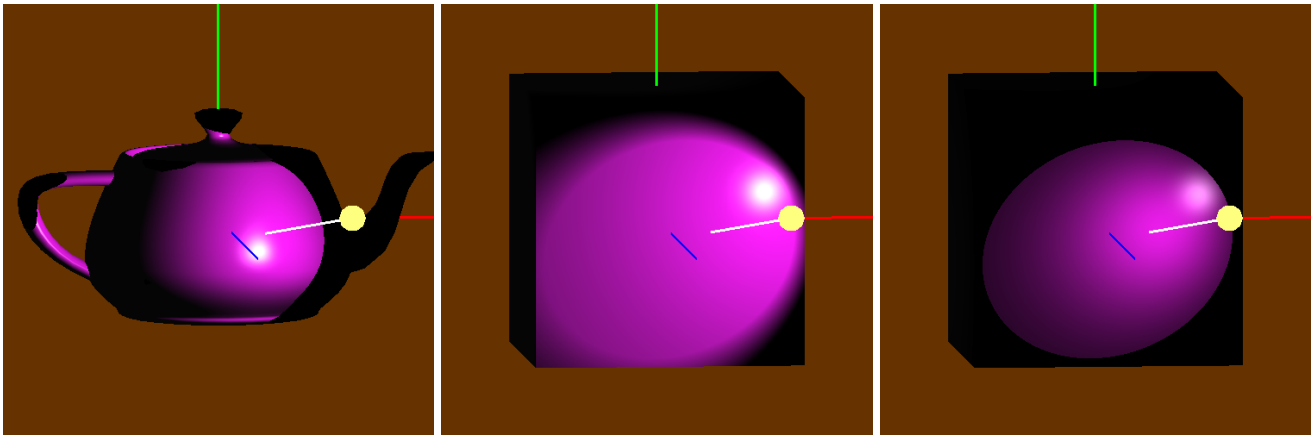


FIGURE 3 – Rendu avec le spot i) avec le modèle du style OpenGL, ii) avec le modèle du style Direct3D, iii) avec un plus grand exposant

Partie 3 : l'application de textures

Le logiciel permettra d'afficher 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. On spécifiera les coordonnées de texture afin de montrer sur le cube l'échiquier répété 3 fois dans les deux directions (Voir Figure 6.)
- Les autres objets (tore, sphère, théière, cube, cylindre, cône) définissent eux-mêmes leurs coordonnées de texture.

Enfin, on pourra choisir que les texels noirs soient affichés en noir, mi-colorés (moyenne du noir et de la couleur de l'objet) ou complètement transparents (voir les Figures 4, 5 et 7).

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.

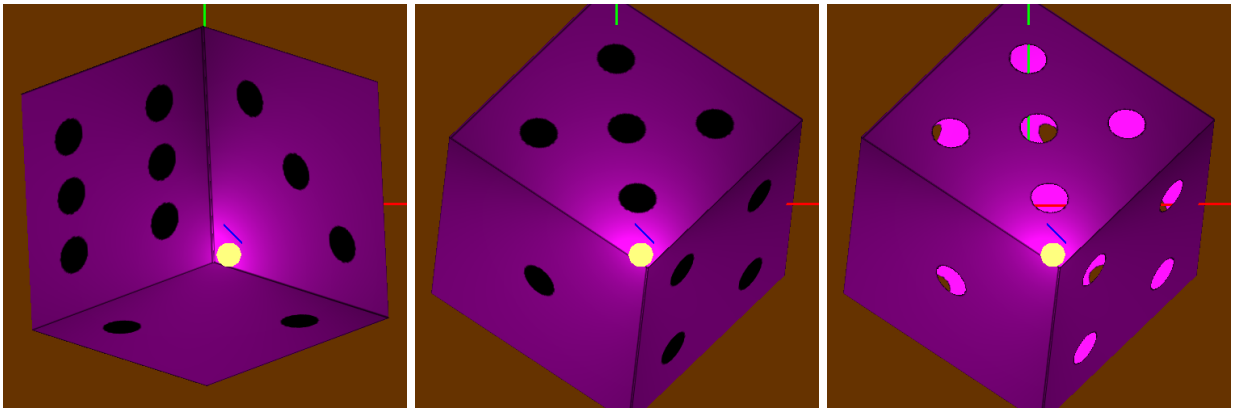


FIGURE 4 – Texture appliquée sur le dé en 3D (texels noirs opaques ou transparents)

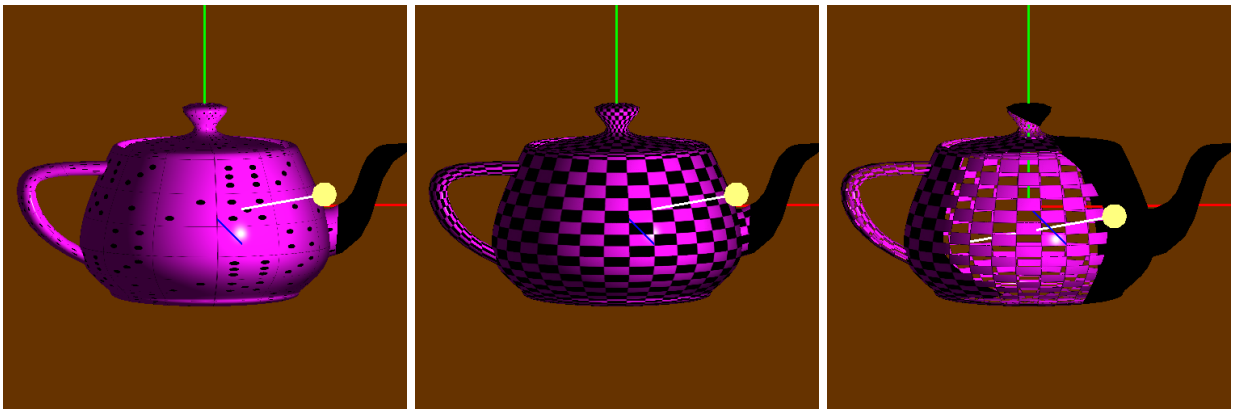


FIGURE 5 – La théière texturée (texels noirs opaques ou transparents)

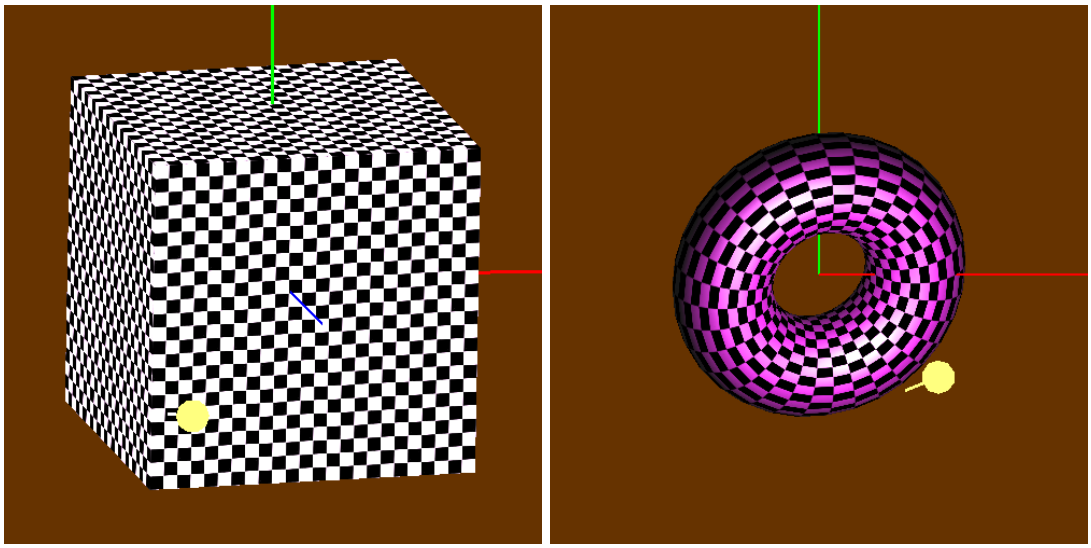


FIGURE 6 – Texture échiquier appliquée sur le cube ou sur le tore

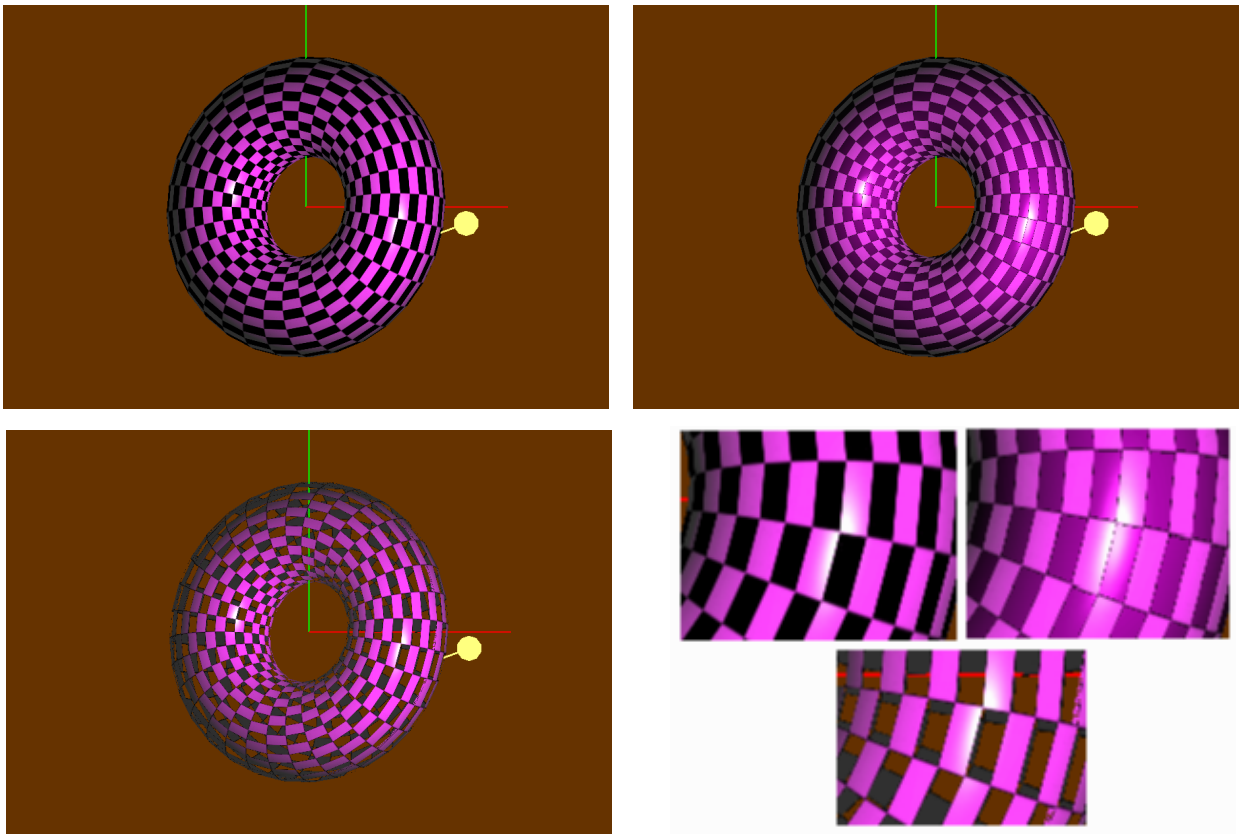


FIGURE 7 – Les texels noirs peuvent être affichés en noir, mi-colorés ou complètement transparents

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 modèle de spot inspiré d'OpenGL est correctement implémenté.

E6. Le modèle de spot inspiré 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. L'utilisateur peut changer de texture (entre dé et échiquier).

E10. Le cube est affiché correctement avec la texture du dé (voir Figure 4).

E11. Le cube et le tore sont affichés correctement avec la texture de l'échiquier (voir Figure 6).

E12. Les texels noirs sur l'objet sont affichés en noir, mi-colorés ou transparents (voir Figure 7).

3.2 Exigences non fonctionnelles

Notez bien 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) : 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 pourra donc voir des énoncés semblables à ceux-ci :

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

ou, mieux encore :

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

3.3 Rapport

Vous devez répondre aux questions dans le fichier `Rapport.txt` et l'inclure dans la remise. Vos réponses doivent être complètes et suffisamment détaillées. (Quelqu'un pourrait suivre les instructions que vous avez écrites sans avoir à ajouter quoi que ce soit.)

4 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
`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, tore, sphère, théière, cube, cylindre, cône
`t`	Choisir la texture utilisée : aucune, dé, échiquier
`o`	Changer l'affichage des texels noirs (noir, mi-coloré, transparent)
`g`	Permuter l'affichage en fil de fer ou plein
`n`	Utiliser ou non les normales calculées comme couleur (pour le débogage)
`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

5 Figures supplémentaires

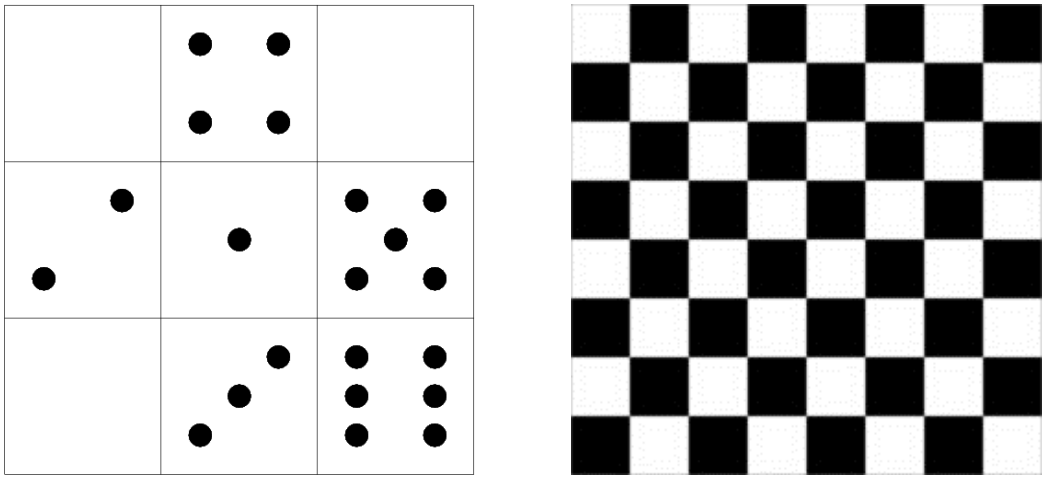


FIGURE 8 – Les textures fournies

6 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. Afficher une sphère au lieu du cube. (Vous devrez alors calculer et spécifier ses normales.)

Partie 2 :

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

Partie 3 :

6. Permettre l'affichage de l'objet texturé avec couleurs ou sans couleur selon la valeur de la variable uniforme `utiliseCouleur` (comme sur l'image de gauche de la Figure 6).
7. Modifier les coordonnées de texture pour constater l'effet sur le résultat visuel.
8. Définir les bonnes coordonnées de texture pour un objet plus complexe composé de triangles.

7 Formules utilisées

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

- 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.

7.2 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 inspirés 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	$(cos(\delta))^{1.01+c/2}$