UNIVERSITÉ DU LITTORAL CÔTE D'OPALE

TP OpenGL 5
Eclairage en OpenGL

Licence Informatique 3ème année

Année 2021-2022

1 Introduction

Dans les Tps précédents, les objets modélisés apparaissaient d'une couleur uniforme, quelle que soit la façon dont vous les regardiez. Cette uniformité rend, d'une part, les images affichées assez peu réalistes et, d'autre part, limite leur compréhension, l'éclairage d'un objet par une ou plusieurs sources de lumières apportant énormément d'informations sur cet objet (sa géométrie, sa localisation dans l'espace etc ...).

OpenGL offre la possibilité d'ajouter des sources lumineuses dans une scène et de calculer leur effet sur les différents objets qui composent cette scène. Dans ce Tp, nous allons donc étudier, de manière progressive, la façon d'utiliser ces nouvelles fonctionnalités.

Opération préliminaire

Récupérez l'archive jointe à cet énoncé et installez-la sur votre machine. Elle contient un dossier `TP5` qui devra être rendu avec les modifications que vous aurez appliquées sur les fichiers sources qu'il contient. Dans son état initial, l'application affiche une théière de couleur uniforme bleue, centrée à l'origine. Elle peut être approchée/éloignée et tournée autour des trois axes principaux via les touches prévues dans le fichier `touches.c`.

2 Première approche

2.1 Activer l'éclairage

Par défaut, OpenGL n'utilise pas les fonctionnalités liées à l'éclairage. Deux opérations sont, au minimum, requises pour que l'on puisse éclairer les objets de la scène :

1. activer l'éclairage ; ceci se fait par l'intermédiaire de l'appel de fonction suivant :

   ```
   glEnable(GL_LIGHTING);
   ```

   A noter que l'éclairage peut ensuite être désactivé par l'appel symétrique suivant :

   ```
   glDisable(GL_LIGHTING);
   ```

   Vous pouvez connaître l'état d'activation de l'éclairage en effectuant l'appel suivant :

   ```
   glIsEnabled(GL_LIGHTING);
   ```

   qui vous renvoit les valeurs `GL_TRUE` ou `GL_FALSE`.
2. activer une ou plusieurs sources de lumière ; sous OpenGL, chaque source est identifiée par une constante symbolique de la forme `GL_LIGHTn`, avec $n \in [O, .., 7]$, ce qui autorise l'utilisation simultanée d'au plus 8 sources. Chaque source peut alors être activée par un appel de la forme `glEnable(GL_LIGHTn)` ou désactivée par un appel de la forme `glDisable(GL_LIGHTn)`.

Application :

1. Créer, dans le module `tp3d.c`, la fonction `init_eclairage` qui permet **d'activer l'éclairage et d'activer la source** 0. Vous rajouterez dans cette fonction les lignes suivantes, qui permettent de préciser la position de la source 0 :

   ```
   GLfloat light0_position[] = {10.0, 10.0, 10.0, 1.0};

   glLightfv(GL_LIGHT0, GL_POSITION, light0_position);
   ```

Ne pas oublier d'appeler la fonction `init_eclairage` dans le `main` (après l'appel à la fonction `init_screen`) ;

2. Compiler et tester l'application qui, à ce stade, doit vous afficher une théière grise illuminée ;
3. Ajouter, dans le module `touches.c`, les fonctionnalités nécessaires à l'activation/désactivation de l'éclairage : un appui sur la touche `o` doit se comporter comme un interrupteur et activer/désactiver l'éclairage selon son état au moment de l'appui ;
4. Compiler et tester le programme.

2.2 Rendu lissé ou non lissé

Par défaut le rendu est effectué en mode lissé, c'est à dire que la valeur de couleur en tout point d'une facette est interpolée depuis la valeur de couleur aux sommets de cette facette. Le mode de rendu peut être modifié par l'intermédiaire de la fonction `glShadeModel`, qui peut prendre l'une des deux valeurs suivantes comme unique paramètre :

— `GL_SMOOTH` : le rendu doit être effectué en mode lissé (valeur par défaut) ;
— `GL_FLAT` : le rendu doit être fait en mode « lambert » , c'est à dire avec une intensité constante sur toute la surface de la facette.

Il est d'autre part possible de connaître le mode de rendu courant en utilisant la fonction suivante :

```
void glGetIntegerv(GLenum nom, GLint *valeur)
```

avec

— `nom` : le type d'information à récupérer ; ici ce paramètre prendra la valeur `GL_SHADE_MODEL` ;
— `valeur` : une variable qui récupérera la valeur du mode de rendu courant, les valeurs possibles étant ici `GL_SMOOTH` et `GL_FLAT`.

Application : en tenant compte de ce qui vient d'être présenté, modifier l'application `tp3d` de telle manière que l'appui sur la lettre `'l'` permette de basculer alternativement le mode de rendu entre `GL_SMOOTH` et `GL_FLAT` et réciproquement. Vous noterez qu'en mode `GL_FLAT`, les facettes modélisant la surface de la théière apparaissent nettement, puisque l'éclairage n'est calculé qu'une seule fois pour tous les points qui appartiennent à cette facette.

3 Spécifier le matériau des objets

Dans les Tps précédents, la couleur d'un objet était précisée par l'intermédiaire d'un appel à la fonction `glColor3f`. Lorsque l'éclairage est activé, les appels à cette fonction sont ignorés, les caractéristiques liées au calcul de la couleur devant être spécifiées par l'intermédiaire d'un matériau, dont les composantes vont permettre un calcul plus réaliste de l'éclairage.

3.1 Les composantes d'un matériau

Sous OpenGL, chaque objet est issu d'un matériau, dont les différentes composantes permettent de spécifier les différents coefficients de réflexion de l'objet (réflexions ambiante, diffuse et spéculaire). OpenGL autorise d'autre part de préciser la brillance du matériau, propriété liée à la puissance s (coefficient spéculaire) qui apparaît dans la formule générale de calcul de l'éclairage [1]. Enfin, OpenGL autorise un objet à émettre de la lumière, les propriétés de cette lumière émise devant être spécifiées dans le matériau.

3.1.1 Réflexion ambiante et diffuse

Principe

Ces deux types de réflexion sont liées à la couleur de l'objet : ils spécifient le pourcentage de lumière qui est réfléchie par l'objet, pour chaque longueur d'onde. Ainsi, un objet que l'on souhaite de couleur rouge doit réfléchir une grande partie de la lumière incidente dans les longueurs d'onde associées à cette couleur et absorber une grande partie des autres longueurs d'onde.

OpenGL différencie la réflexion ambiante de la réflexion diffuse de manière à pouvoir préciser des coefficients éventuellement différents pour chacune d'entre elles. On rappelle que la réflexion ambiante est liée à la réflexion de la lumière ambiante qui, par définition n'a pas de direction d'incidence ; elle semble venir de partout et permet d'attribuer de la lumière aux parties d'objets non directement éclairées par

1. Revoir cette formule dans le cours traitant de l'éclairage.

une source. La réflexion diffuse par contre est liée à chaque source et exprime le pourcentage de lumière incidente qui est réfléchie dans toutes les directions par l'objet.

Définition

La définition des propriétés de réflexion ambiante et diffuse d'un objet se font par l'intermédiaire de la fonction suivante :

```
void glMaterialfv(GLenum face, GLenum propriete, float *valeurs)
```

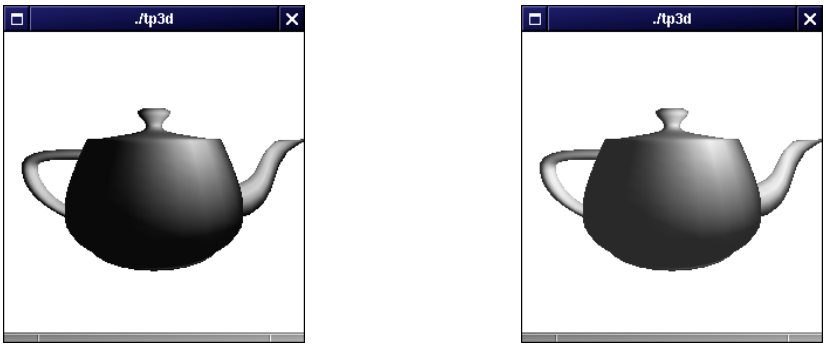
avec

- `face` : permet de préciser si le matériau s'applique sur les faces visibles de l'observateur (`GL_FRONT`), invisibles de l'observateur (`GL_BACK`) [2] ou les deux (`GL_FRONT_AND_BACK`) ;
- `propriete` : ce paramètre permet de préciser les propriétés de réflexion du matériau ; les valeurs possibles, pour cette partie, sont :
  - `GL_AMBIENT` : spécifie la valeur des coefficients ambiants ;
  - `GL_DIFFUSE` : spécifie la valeur des coefficients diffus ;
  - `GL_AMBIENT_AND_DIFFUSE` : spécifie la (même) valeur pour les coefficients ambiants et diffus. Dans la mesure où la réflexion ambiante et réflexion diffuse sont toutes deux intimement liées à la couleur d'un objet, leurs coefficients de réflexion sont généralement identiques. L'utilisation de cette constante évite donc d'effectuer deux appels à la fonction `glMaterialfv`.
- `valeurs` : ce paramètre représente un tableau de 4 valeurs réelles, comprises entre 0.0 et 1.0 : une valeur pour chacune des 3 primaires R, V et B et une pour le canal alpha (lié à la notion de transparence, mais qui ne sera pas utilisé ici).

Lorsque l'utilisateur ne crée pas de matériau, les coefficients de réflexion ambiant et diffus possèdent des valeurs par défaut :

- ambiant : $(0.2, 0.2, 0.2, 1.0)$
- diffus : $(0.8, 0.8, 0.8, 1.0)$

Ces valeurs expliquent pourquoi votre théière apparaît de couleur grisâtre aux endroits non éclairés directement et plus blanche aux endroits directement exposés à la source (voir figure 1.a).



(a) (b)

FIGURE 1 – éclairage ambiant gris (valeur 0.2) (a) éclairage ambiant et diffus de valeur 0.8 (b)

Application

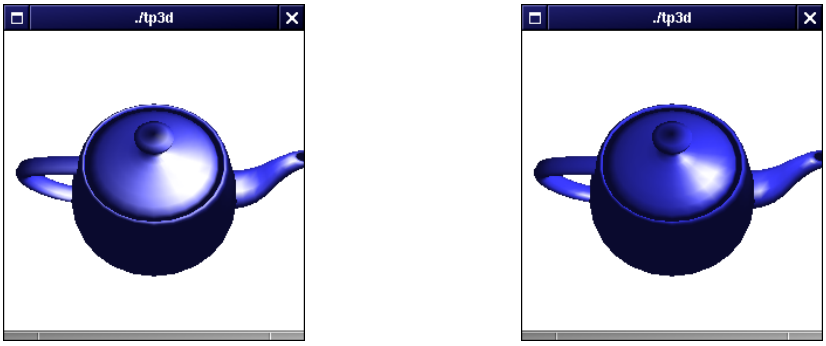
1. après avoir défini, dans la fonction `laTeapot`, un tableau de 4 `GLfloat` initialisé aux valeurs $(0.8, 0.8, 0.8, 1.0)$, modifier la valeur des coefficients ambiants de votre théière (en ne considérant que les facettes faisant face à l'observateur), recompiler votre application et vérifier le résultat (comparer les images de la figure 1) ;
2. modifier les coefficients de votre tableau de telle sorte que le coefficient de réflexion bleu soit très fort et les coefficients rouge et vert faibles (laisser le coefficient alpha à 1.0). Utiliser ce tableau pour affecter en même temps les propriétés ambiante et diffuse de la théière.

2. Certaines opérations OpenGL peuvent rendre les faces ≪ arrière ≫ d'un objet visibles. Lorsque l'utilisateur souhaite utiliser ces opérations, qui ne seront pas détaillées ici, il est donc important que ces faces arrières soient elles-aussi éclairées de manière convenable, sous peine d'obtenir un affichage peu réaliste ...

3.1.2 Réflexion spéculaire et brillance

Principe

La réflexion spéculaire est liée à la notion de brillance d'un objet : plus un objet est brillant et plus il a tendance à réfléchir la lumière incidente de manière directionnelle. Ceci permet de voir le reflet d'une source de manière plus ou moins nette selon le degré de brillance de l'objet [3](ce qui implique que, contrairement aux réflexions diffuse et ambiante, la position de l'observateur est prise en compte). Sous OpenGL il est possible de gérer la brillance d'un objet, d'une part en précisant les coefficients spéculaires utilisés pour chaque composante R, V, B et alpha et, d'autre part, en précisant un indice de brillance `s` (coefficient spéculaire), compris entre 0.0 et 128.0, qui précise la « largeur » du cône spéculaire : plus `s` est grand, et plus l'observateur doit être proche de la direction de réflexion spéculaire pour pouvoir percevoir cette réflexion. Ceci est illustré sur la figure ci-dessous : sur l'image de gauche (2.a), le coefficient `s` vaut 10.0, ce qui génère des réflexions spéculaires relativement larges ; sur l'image de droite(2.b), la valeur du coefficient a été mise à 50.0, ce qui a pour effet de réduire la tache spéculaire.



(a) (b)

FIGURE 2 – Influence de l'exposant spéculaire : `s` = 10 (a) `s` = 50 (b)

Définition

Par défaut, les valeurs des coefficients spéculaires associés à un matériau sont $(0.0, 0.0, 0.0, 1.0)$, le coefficient `s` valant 0.0.
Pour affecter des valeurs aux coefficients, il faut utiliser la même fonction que pour les coefficients diffus et ambiant : `glMaterialfv`, en utilisant la constante symbolique `GL_SPECULAR` pour le paramètre `propriete`. Pour obtenir un effet réaliste, il est conseillé de mettre la même valeur aux différents coefficients spéculaires, les réflexions réelles conservant la couleur de la source : un objet rouge illuminé par une source blanche produira un reflet spéculaire blanc, la composante diffuse conservant la couleur rouge de l'objet à l'extérieur de la zone spéculaire.
La valeur de `s` doit être fixée par une fonction légèrement différente :

```
void glMaterialf(GLenum face, GLenum propriete, float valeur)
```

avec
- `face` : *cf* la fonction `glMaterialfv` ;
- `propriete` : la constante symbolique `GL_SHININESS` ;
- `valeur` : un réel dans $[0.0, 128.0]$.

Application

Modifier `tp3d` de manière à activer l'éclairage spéculaire sur votre théière et à pouvoir incrémenter ou décrémenter la valeur de son coefficient de brillance `s` par un appui sur la touche `'S'` ou `'s'`.

3.1.3 L'émission de lumière

Le dernier paramètre d'un matériau sur lequel le programmeur peut jouer concerne la faculté de ce matériau d'émettre de la lumière : en plus de réfléchir de la lumière, le matériau peut être considéré

3. Sous OpenGL, seul le reflet direct d'une source peut être géré par la réflexion spéculaire ; la gestion des reflets entre objets doit faire appel, soit à des astuces de programmation qui sont en dehors du cadre de ce tp, soit à des algorithmes d'illumination beaucoup plus complexes que ce qu'offre OpenGL (algorithme du lancer de rayons par exemple).

comme émettant de l'énergie lumineuse en propre, énergie dont la valeur vient s'ajouter aux quantités que le matériau réfléchit. Par défaut, la valeur d'émission pour chaque primaire est nulle : $(0.0, 0.0, 0.0, 1.0)$. Ces valeurs peuvent être modifiées par un appel à la fonction `glMaterialfv` avec la constante symbolique `GL_EMISSION` comme paramètre `propriete` et un tableau de 4 nombres réels comme paramètre `valeur`.

Il faut cependant noter que la valeur d'émission associée à un objet n'est jamais prise en compte dans le calcul d'illumination des autres objets ; ce paramètre ne sert qu'à améliorer le rendu de certains objets, comme les phares de voiture par exemple. Si un objet doit effectivement émettre de la lumière, une astuce consiste à « cacher » une source lumineuse au sein de cet objet ...

4 Spécifier les caractéristiques d'une source

Comme pour les matériaux, chaque source possède des attributs permettant de paramétrer ses caractéristiques.

4.1 Les valeurs d'éclairage

Les caractéristiques concernant les valeurs d'émission d'une source sont au nombre de trois : l'émission ambiante, l'émission diffuse et l'émission spéculaire. Dans les trois cas, la fonction utilisée pour mettre à jour ces valeurs est la suivante :

```
glLightfv(GLenum light, GLenum propriete, float *valeurs)
```

avec

- `light` : précise la source dont on souhaite modifier les propriétés ; les valeurs possibles sont `GL_LIGHT0` à `GL_LIGHT7` ;
- `propriete` : constante symbolique précisant la propriété sur laquelle la fonction doit agir (`GL_AMBIENT`,`GL_DIFFUSE` ou `GL_SPECULAR`) ;
- `valeurs` : un tableau de 4 réels qui contient la valeur des 3 primaires R, V et B et du canal alpha.

4.1.1 L'intensité ambiante

Principe

L'intensité ambiante d'une source est utilisée pour les calculs d'éclairage ambiant : pour chaque objet, la valeur d'éclairage ambiant de toutes les sources actives est additionnée, puis multipliée par la valeur du coefficient de réflexion ambiant de l'objet. La valeur par défaut pour toutes les sources est $(0.0, 0.0, 0.0, 1.0)$. Compte tenu de cette valeur par défaut, les images que vous avez obtenues jusqu'à présent devraient être différentes ; en effet, toutes les parties d'objet qui ne sont pas directement illuminées par la source `GL_LIGHT0` devraient être noires (elles ne reçoivent que de l'éclairage ambiant ...). En fait, le modèle d'éclairage utilisé par OpenGL autorise l'utilisation d'un éclairage ambiant qui n'est lié à aucune source ; sa valeur par défaut est $(0.2, 0.2, 0.2, 1.0)$, ce qui explique que les parties cachées de votre théière ne soient pas entièrement noires. Il est possible de modifier la valeur de cet éclairage ambiant « global » par l'intermédiaire de l'appel de fonction suivant :

```
void glLightModelfv(GLenum propriete, GLfloat *valeurs)
```

avec

- `propriete` : la constante symbolique `GL_LIGHT_MODEL_AMBIENT` ;
- `valeurs` : un tableau de 4 réels spécifiant la valeur pour chaque composante R, V, B et alpha.

Application

1. dans la fonction `init_eclairage`, modifier la position de la source 0 de telle sorte qu'elle se trouve en $z = -3$. Ceci permettra à davantage de parties de la théière de ne pas voir directement la source et permettra de mieux illustrer l'effet de l'éclairage ambiant ;
2. modifier ensuite cette fonction de manière à avoir un éclairage ambiant global nul, puis tester cette modification ;
3. faites les modifications nécessaires pour que la source 0 ait une composante ambiante de valeur $(1.0, 0.0, 0.0, 1.0)$, le coefficient de réflexion ambiant et diffus de la théière pour le rouge devant être nul. Avant de relancer l'application, indiquer le résultat attendu pour les parties illuminées uniquement par le terme ambiant. Le résultat obtenu est-il conforme à celui que vous imaginiez ?

4. remettre un coefficient de réflexion ambiante et diffuse de la théière à une valeur faible (par exemple 0.2). Comme précédemment imaginer la façon dont l'illumination doit agir sur la couleur de la théière et vérifier par rapport au résultat obtenu ;
5. faites les modifications nécessaires pour que la source 0 ait une composante ambiante de valeur $(1.0, 1.0, 1.0, 1.0)$, imaginer le résultat obtenu et vérifier ;
6. fixer la valeur de la composante ambiante de la source 0 à $(0.1, 0.1, 0.1, 1.0)$ et celle de la composante ambiante globale à sa valeur par défaut.

4.1.2 L'intensité diffuse

Principe

Comme précédemment, il est possible de fixer l'intensité diffuse d'une source, qui sera combinée avec la réflexion diffuse de l'objet pour obtenir la valeur d'éclairage diffus. Par défaut OpenGL utilise des valeurs différentes selon la source :
— sources 0 : les valeurs par défaut sont $(1.0, 1.0, 1.0, 1.0)$;
— autres sources : les valeurs par défaut sont $(0.0, 0.0, 0.0, 1.0)$.

Application

Modifier votre application de telle sorte que l'appui sur la touche 'a' augmente la valeur de l'intensité diffuse de la source 0 (augmentation identique pour chaque composante R, V et B), tandis que l'appui sur la touche 'd' doit diminuer cette même intensité. On précise qu'il est possible de récupérer les valeurs actuelles pour une source via un appel à la fonction :

```
void glGetLightfv(GLenum light, GLenum pname, GLfloat * params);
```

où :
— `light` est l'identifiant de la source de la forme `GL_LIGHTn` ;
— `pname` est l'attribut à récupérer (par exemple `GL_DIFFUSE` ;
— `params` un pointeur vers la zone mémoire dans laquelle seront stockées les informations récupérées.

4.1.3 L'intensité spéculaire

Principe

Même principe que pour les composantes précédentes, les valeurs par défaut des intensités étant les suivantes :
— source 0 : $(1.0, 1.0, 1.0, 1.0)$;
— autres sources : $(0.0, 0.0, 0.0, 1.0)$.

Application

Modifier votre application de telle sorte que l'appui sur la touche 'A' augmente la valeur de l'intensité spéculaire de la source 0 (augmentation identique pour chaque composante R, V et B), tandis que l'appui sur la touche 'D' doit diminuer cette même intensité.

4.2 La position

La position d'une source peut être gérée par un appel de la forme :

```
glLightfv(GL_LIGHTn, GL_POSITION, lightn_position);
```

avec `lightn_position` un tableau de 4 réels qui précisent les coordonnées de la source en coordonnées homogènes. Si le quatrième paramètre (ω) est nul, cela signifie que la source est à l'infini, dans la direction donnée par les 3 premières composantes. Dans ce cas, la source ne peut pas être déplacée. Par contre, dans le cas d'une source située à distance finie ($\omega = 1.0$ en général), celle-ci peut être déplacée au même titre que les autres objets de la scène : comme tout objet, un source va subir les transformations géométriques que vous appliquez sur la scène. Il suffit, lors de la définition des objets à utiliser (ici, dans le corps de la fonction `dessiner`) d'appeler la fonction `glLightfv` pour positionner la source dans le repère globale, toute transformation appliquée ensuite à ce repère étant répercutée ensuite sur la position de la source.

Application

Faites les modifications nécessaires pour que la source `GL_LIGHT0` soit toujours positionnée au dessus de votre théière, quelles que soient les transformation géométriques modifiant la façon dont cette théière est affichée.

4.3 Les projecteurs

Les sources utilisées jusqu'à présent étaient toutes supposées être ponctuelles et émettre de l'énergie dans toutes les directions. Il est cependant possible de créer des sources qui n'éclairent que dans un cône de direction, pour obtenir un effet de projecteur. Pour cela deux paramètres illustrés sur la figure 3 doivent être précisés pour la source concernée :

— l'angle d'ouverture du projecteur : la propriété concernée est notée par la constante symbolique `GL_SPOT_CUTOFF`. Elle doit être positionnée par un appel à la fonction `glLightf`, qui prend comme paramètres l'identifiant de la source, la constante précédente et la valeur réelle concernant l'angle d'ouverture. Les valeurs autorisées pour l'angle d'ouverture sont comprises entre 0° et 90° ;

— la direction d'éclairage du spot : la propriété concernée est notée par la constante symbolique `GL_SPOT_DIRECTION`. Elle doit être positionnée par un appel à la fonction `glLightfv`, qui prend comme paramètres l'identifiant de la source, la constante précédente et un tableau de 3 réels contenant le vecteur direction précisant l'orientation de la source.

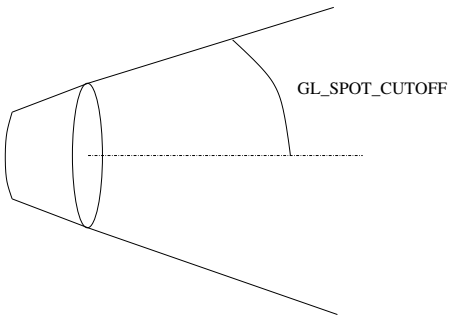


FIGURE 3 – Schématisation de l'angle d'ouverture d'un projecteur

A noter que les transformations appliquées sur la scène dans laquelle se trouve le projecteur s'appliquent aussi sur la direction du projecteur ...

Application

1. compléter la fonction `init_eclairage` de telle sorte qu'elle crée une seconde source de lumière (`GL_LIGHT1`) qui sera une source de type spot, d'intensité ambiante 0.1 pour les 3 canaux, d'intensité diffuse et spéculaire 1.0 pour les 3 canaux et d'ouverture 30°. La direction principale du spot sera le vecteur $(-1.0, -1.0, -1.0)$. Vous activerez cette source à la place de la source `GL_LIGHT0` ;
2. dans la fonction `dessiner`, ajouter la position de la source, de telle sorte qu'elle reste toujours positionnée au même endroit par rapport à la théière, quelles que soient les transformations géométriques appliquées à celle-ci. Vous prendrez comme position le point de cooordonnées $(4.0, 4.0, 4.0, 1.0)$. Compiler et tester cette première version ;
3. modifier votre application de telle sorte que l'appui sur la touche `>` augmente la valeur de l'angle d'ouverture et l'appui sur la touche `<` le diminue. Vous prendrez garde à ne pas dépasser les valeurs limites.

5 Discrétisation et normales

Comme vous l'avez vu en cours, le calcul de l'éclairage diffus et spéculaire nécessite la connaissance de la normale à la surface éclairée, en chacun des sommets utilisés pour la modéliser. Les exemples précédents utilisent tous l'objet théière fourni par Glut, le problème des normales y étant caché puisque leur définition est faite à l'intérieur de la fonction. Nous allons voir ici comment définir les normales pour les surfaces que vous seriez amenés à créer par vous même ainsi que certains problèmes sous-jacents.

5.1 Définition des normales

La définition d'une normale s'effectue par l'intermédiaire de l'une des fonctions suivantes :

```
void glNormal3f(GLfloat nx, GLfloat ny, GLfloat nz)
void glNormal3fv(GLfloat *vecteur)
```

avec

- `nx`, `ny` et `nz` les coordonnées `x`, `y` et `z` du vecteur normal ;
- `vecteur` un tableau de 3 nombres réels représentant les coordonnées `x`, `y` et `z` du vecteur normal.

Ces fonctions peuvent être utilisées lorsque vous définissez l'un des sommets de votre polygone : juste avant de définir le sommet (`glVertex`), vous définissez sa normale avec l'une des deux fonctions précédentes. A noter que lorsqu'une normale est définie entre un `glBegin` et un `glEnd`, elle reste valable jusqu'au `glEnd`, sauf si vous redéfinissez une autre normale (la normale courante est stockée dans le contexte graphique). Ceci peut être intéressant lorsque l'on définit un objet plat, pour lequel la normale est la même en chacun de ses sommets ...

Remarques

- lorsque vous fournissez un vecteur normal à OpenGL (que ce soit sous forme de ses 3 coordonnées ou par l'intermédiaire d'un tableau de 3 réels), il est important que ce vecteur soit normalisé (de longueur 1). En effet, lors des calculs d'éclairage, la norme du vecteur est utilisée (produits scalaires), ce qui implique que si les vecteurs ne sont pas normés, l'éclairage peut être plus grand ou plus petit que l'éclairage réel, ou peut même varier sans raisons ;
- il est possible de demander à OpenGL de normaliser systématiquement tous les vecteurs normaux avant de les utiliser. Ceci se fait en activant la fonction de normalisation :
  ```
  glEnable(GL_NORMALIZE)
  ```
  Ceci a cependant tendance à réduire les performances de votre application ; il est donc préférable, lorsque cela est possible, de normaliser soit même les vecteurs normaux avant de les utiliser ;
- Les opérations de changement d'échelle (`glScalef`) modifient les vecteurs normaux et, par voie de conséquence, leur norme. Lorsque ce type d'opérations est utilisé, il est préférable d'utiliser la fonction de normalisation automatique.

5.2 Le problème de la discrétisation insuffisante

5.2.1 Illustration du problème

1. dans le fichier `graphique.c` définir une fonction nommée `sol` qui construit un carré de côté 2.0, centré en $(0.0, 0.0, 0.0)$ et situé dans le plan horizontal Oxz. Penser à prévoir un matériel pour votre surface et à fournir une normale dirigée selon l'axe Oy pour chacun de ses 4 sommets ;
2. positionner un projecteur, dirigé vers le sol, sur l'axe Oy, à une hauteur de 2.0, avec un angle d'ouverture de 11° ;
3. compiler puis exécuter l'application ; le résultat obtenu est-il conforme à ce que l'on est en droit d'attendre ? Diminuer l'angle d'ouverture du spot avec la touche `<` ; que se passe-t'il ?

On rappelle ici que l'éclairage est calculé aux sommets des polygones qui définissent les objets. L'éclairage est ensuite interpolé au sein du polygone. Ceci peut poser un problème si un polygone est trop grand : les sommets du polygone peuvent être peu, voire pas, éclairés par la source, tandis que l'intérieur du polygone est directement éclairé par cette même source. L'éclairage n'étant calculé qu'aux sommets, tout le polygone apparaît globalement sombre (voir figure 4).

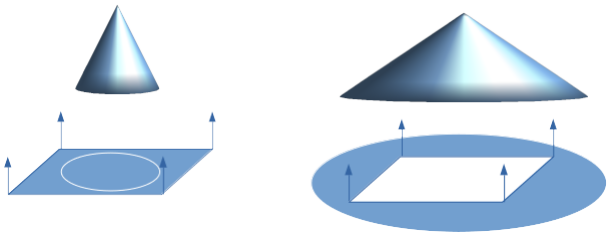


FIGURE 4 – Effet de la discrétisation sur l'éclairage par un spot. A gauche, l'ouverture du sport ne permet pas d'éclairer les sommets du carré - il reste sombre. A droite, les sommets sont éclairés de la même manière et toute la surface du carré est de couleur uniforme.

5.2.2 Accroître la discrétisation

Pour remédier à ce problème, il est nécessaire d'augmenter le nombre de points en lesquels l'éclairage sera calculé. Pour ce faire, le polygone de départ doit être découpé en de nombreux polygones plus petits.

Application

Faites les modifications nécessaires dans votre fonction `sol` pour que le polygone de départ soit découpé en $N \times N$ polygones et visualiser l'éclairage résultant en fonction de la valeur de N. Vous compléterez votre application de telle sorte qu'une variable gloable, initialisée à 1, représente le nombre de subdivisions (la valeur de N), que l'appui sur la touche `*` multiplie par 2 le nombre de subdivisions courantes tandis que l'appui sur la touche `/` divise par 2 (avec un minimum à 1). La figure 5 ci-dessous illustre les résultats qui peuvent être obtenus, avec un angle d'ouverture du spot de 6°.

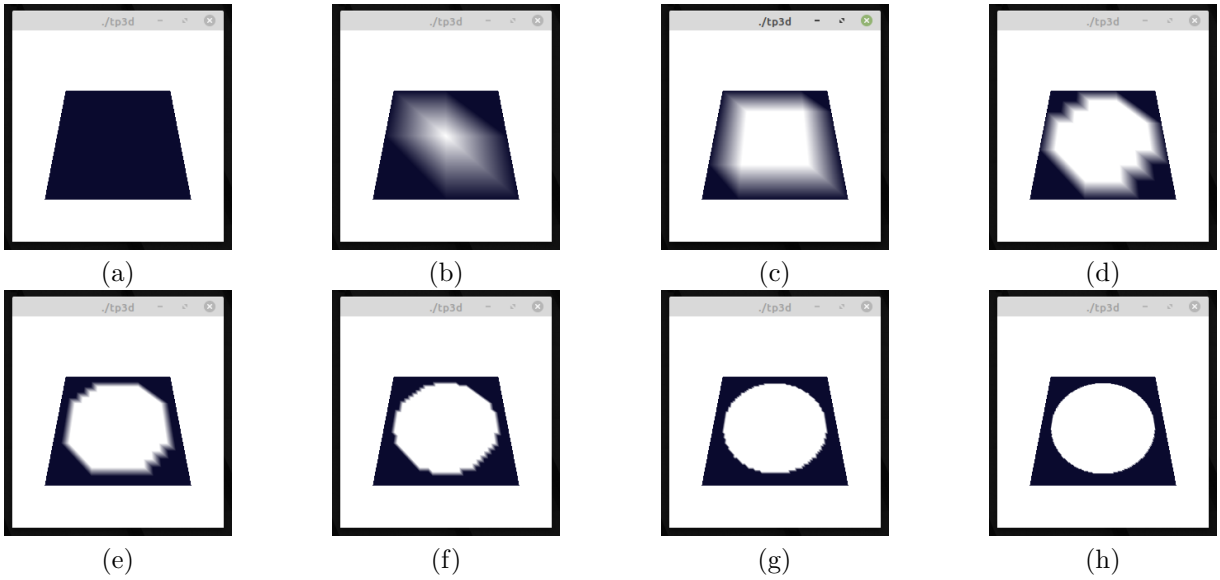


FIGURE 5 – Subdivisions successives du carré de côté 2 centré à l'origine : (a) $N = 1$, (b) $N = 2$, (c) $N = 4$, (c) $N = 8$, (d) $N = 16$, (e) $N = 32$, (f) $N = 64$, (g) $N = 128$ et (h) $N = 256$. Le spot est placé sur l'axe Oy à deux unités de distance du carré, a un angle d'ouverture de 6° et éclaire le carré de manière perpendiculaire.