5

TP OpenGL : Eclairage de scène

Nicholas Journet - TP OpenGL - IUT - [1]

Objectifs du TP

L'objectif de ce TP es d'obtenir des rendus plus vraisemblables, en mettant en oeuvre les moyens offerts par OpenGL pour simuler un éclairage réaliste.
Pour le moment, nous nous sommes contentés d'affecter à chacun des sommets de nos polygones une couleur fixe. Dans la réalité, un point de couleur rouge apparaîtra différemment suivant l'éclairage auquel il est soumis et l'angle sous lequel vous l'observez. OpenGL permet d'effectuer un rendu prenant en compte l'illumination des objets. Pour simuler de manière réaliste une scène tridimensionnelle il est nécessaire de reproduire les lois physiques qui régissent la lumière. Ces lois sont complexes, et il est impossible de les reproduire exactement. On se contente donc de choisir un modèle mathématique aussi proche que possible de la réalité. En fonction de l'application à laquelle il est destiné, le modèle d'illumination choisi permettra d'obtenir des résultats plus ou moins réalistes et rapides.
Pour étudier le modèle d'illumination utilisé par OpenGL, il nous faut aborder 3 points :

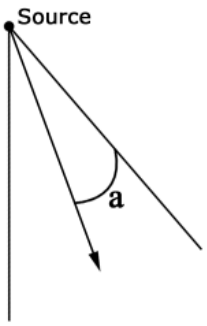
- Les sources lumineuses : quels sont les paramètres permettant de définir une source de lumière ?
- Les matériaux : Une brique ne réfléchit pas la lumière de la même manière qu'une plaque d'aluminium.
- L'algorithme d'éclairage : calculer l'éclairage en chaque pixel de l'image est trop coûteux en temps. OpenGL utilise une astuce pour accélérer le rendu

5.1 Source lunimneuse

Avec OpenGL, il est possible de créer une scène contenant jusqu'à huit sources de lumière. Ces sources peuvent être de type 'omnidirectionnel' (la source émet de la lumière uniformément dans toutes les directions, comme le soleil), ou 'spot' (la lumière n'est émise que dans un cône).
Une source de lumière est caractérisée par 10 paramètres qu'on peut classer en 4 catégories :

1. Les paramètres de lumière : la lumière émise par une source est formée de trois composantes. La plus importante est la composante diffuse. Elle est réfléchie par un objet dans toutes les directions. La composante spéculaire correspond à la lumière qui est réfléchie dans une direction privilégiée (et qui est donc à l'origine de l'effet de brillance). La composante ambiante est la plus difficile à appréhender. La lumière ambiante d'une scène est une lumière non directionnelle, que l'on peut considérer comme issue des multiples réflexions de rayons lumineux.
2. Le paramètre de position : comme son nom l'indique, il permet de définir la position de la source de lumière dans la scène.
3. Les paramètres de spot : l'angle de coupure, illustré sur la figure ci-dessous, permet de définir le type de source désiré. Si l'angle est compris entre 0 et 90˚ cela correspond à la définition d'un spot, et il est alors possible de modifier les deux autres paramètres de spot : la direction dans laquelle il pointe et l'exposant du spot. Ce dernier paramètre permet de faire varier la concentration de la lumière à l'intérieur du cône définissant le spot. Si l'exposant vaut 0, la lumière est répartie également dans toutes les directions définies par le cône. Plus la valeur de l'exposant est importante, plus la lumière sera concentrée autour de l'axe donné par le paramètre de direction.

1. Ce tp est issu des notes de cours de Xavier Michelon - linuxorg. Toute modification de ce support de cours doit y faire référence



4. Les paramètres d'atténuation : ils permettent de prendre en compte le phénomène physique suivant : plus un objet est éloigné d'une source de lumière, moins il est éclairé par cette dernière. Les paramètres d'atténuation sont au nombre de 3 : le facteur d'atténuation constante, le facteur d'atténuation linéaire et le facteur d'atténuation quadratique. Si on note respectivement A_c, A_l et A_q ces trois coefficients, l'intensité reçue par un point P situé à une distance d de la source lumineuse est divisée par A_c+A_l*d+A_q*d^2. Par défaut, A_c=1 et A_l=A_q=0, ce qui correspond à une atténuation nulle (division par 1).

5.2 Les matériaux

La spécification d'un matériau pour un objet se fait par l'intermédiaire de 5 paramètres :

- La couleur diffuse. Dans la réalité, un rayon lumineux est constitué d'un ensemble d'ondes de longueurs différentes. A chaque longueur d'onde correspond une couleur (visible ou non). Un objet frappé par un rayon lumineux va absorber certaines longueurs d'onde et réfléchir les autres. Si l'herbe est verte lorsqu'elle est éclairée par la lumière du soleil, c'est parce qu'elle ne réfléchit que les ondes dont la couleur est verte.
  Dans notre cas, une couleur est représentée par un triplet de composantes rouge, verte, et bleue. Par synthèse additive, on arrive à recréer à partir de ces trois composantes la plupart des couleurs visibles. Pour définir le comportement d'un objet vis-à-vis d'une couleur, il suffit de dire quel pourcentage de chaque composante est réfléchi. Ainsi, si on affecte à un matériau les pourcentages (R=1, V=0.5, B=0), soit (R=100%, V=50%, B=0%), et si on l'éclaire avec une lumière blanche (R=1,V=1,B=1), le matériau va réfléchir un rayon de couleur (R=1,V=0.5,B=0), et l'objet paraîtra orange. Cet exemple montre bien la double signification physique du paramètre : on peut l'interpréter comme le pourcentage réfléchi de chaque composante de couleur, ou bien comme la couleur du rayon réfléchi lorsque l'objet est éclairé par une source de lumière blanche. Vous remarquerez que si vous éclairez cet objet avec une lumière bleue (R=0,V=0,B=1), le matériau va renvoyer une rayon de couleur noire (R=0,V=0,B=0) qui correspond à une absence de lumière.
- La couleur spéculaire, la couleur ambiante et la couleur émise. Puisque il est possible de découposer la lumière émise par une source en lumière diffuse, spéculaire et ambiante, il est possible de spécifier le comportement du matériau pour chacune de ces composantes. Ceci explique les trois premiers paramètres du matériau : couleur diffuse, couleur spéculaire et couleur ambiante. Le paramètre de couleur émis correspond à une composante de lumière supplémentaire, qui permet de prendre en compte le fait que certains objets peuvent émettre eux-même de la lumière. Si on modélise une ampoule allumée, il est possible d'attribuer du blanc ou du jaune à la couleur émise par le matériau de l'objet. Pour les matériaux classiques, la couleur émise est noire. Cependant, cette couleur émise n'est pas considérée comme une source de lumière pour les autres objets de la scène.
- Le coefficient de brillance. Pour expliquer ce dernier paramètre, revenons à ce que nous avons vu concernant la lumière spéculaire : elle est à l'origine du phénomène de brillance qui crée des tâches de lumière intense sur les objets. La composante spéculaire de la lumière est réfléchie dans une direction privilégiée. Dans la réalité, cette réflexion ne se fait jamais de manière parfaite, et le coefficient de brillance permet de modéliser cette imperfection. Sa signification physique est un étalement des taches spéculaires sur les objets lorsqu'on diminue la valeur du coefficient.

5.3 Algorithme d'éclairage

Pour des raisons d'efficacité, OpenGL ne calcule pas la couleur de chaque pixel d'un polygone : soit il remplit chaque polygone avec un couleur unie (mode de remplissage `Flat`), soit il utilise un algorithme de Gouraud (mode `Smooth`), dont le principe est le suivant : pour un polygone donné, la couleur de chacun des sommets est calculée, et le polygone est rempli avec un dégradé entre ces différentes couleurs.
Pour calculer correctement la réflexion des rayon lumineux en un point, OpenGL a besoin de connaître la perpendiculaire à la surface de l'objet au point considéré. On appelle cette donnée une normale. Dans le programme exemple que vous allez coder dans ce TP, nous utiliserons une théière générée avec ses normales.

5.4 Exemple

L'objectif est d'afficher à l'écran une théière générée par glut et éclairée par 2 sources lumineuses différentes. Il faudra avoir la possibilité de tourner autour de la théière avec les touches 'a' et 'z', et de faire varier certains paramètres d'éclairage avec d'autres touches.

5.4.1 Paramètres d'éclairage

L'architecture du programme est classique et les seules nouveautés concernent l'utilisation du modèle d'illumination. La phase d'initialisation de l'éclairage commence par la spécification du mode remplissage des polygones avec

```
1 glShadeModel(GL_SMOOTH);
```

Ensuite on définit les paramètres du modèle de lumière. Ici nous utiliseronsun point de vue local afin de rendre le calcul des reflets spéculaires plus réalistes.

```
1 glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER,GL_TRUE);
```

Ensuite on indique à OpenGL qu'on souhaite utiliser le calcul d'éclairage, en activant la variable d'état `GL_LIGHTING` :

```
1 glEnable(GL_LIGHTING);
```

Nous avons vu qu'OpenGL permet d'utiliser jusqu'à huit sources de lumière. Ces huit lampes sont indexées par les constantes `GL_LIGHT0` à `GL_LIGHT7`. Il faut activer chacune des sources qu'on souhaite utiliser (2 dans notre cas) :

```
1 glEnable(GL_LIGHT0);
2 glEnable(GL_LIGHT1);
```

Vient ensuite le paramétrage des lampes. Il se fait avec une unique fonction, `glLightfv()`, dont le prototype est le suivant :

```
1 void glLightfv(GLenum lampe,GLenum nomparam, GLType param)
```

Le paramètre `lampe` désigne la lampe dont on veut modifier un propriété.
`nomparam` est le nom du paramètre à modifier. Il s'agit d'une des dix propriétés de source lumineuse évoquées précédemment : `GL_DIFFUSE, GL_AMBIENT, GL_SPECULAR, GL_POSITION, GL_SPOT_CUTOFF, GL_SPOT_DIRECTION, GL_SPOT_EXPONENT, GL_CONSTANT_ATTENUATION,`
`GL_LINEAR_ATTENUATION, GL_QUADRATIC_ATTENUATION`
`param` désigne la valeur à affecter au paramètre choisi. Les paramètres sont passés sous forme de tableaux.
La définition de la position des sources de lumières se trouve dans la fonction d'affichage. En effet, tout comme les sommets des polygones, les paramètres de position et de direction d'une source subissent les transformations contenues dans la matrice de modélisation-visualisation. Il faut donc placer judicieusement la déclaration de ces deux paramètres. Les deux spots que nous utilisons sont omnidirectionnels (car nous ne modifions pas la valeur par défaut de `GL_SPOT_CUTOFF` qui vaut 180), et donc le paramètre direction ne nous est pas utile.

5.4.2 Paramètres de matériaux

Le système d'affection des propriétés de matériau utilise le principe de machine à états. OpenGL gère un matériau courant. Lorsqu'un polygone est décrit, il se voit affecter le matériau courant. La modification du matériau courant se fait avec la fonction `GlMaterialfv()` :

```
Void glMaterialfv(GLenum face, GLenum nomparam, Gltype param) ;
```

`face` indique la face (avant ou arrière) dont on souhaite modifier les paramètres. Nous n'avons pas encore abordé les considérations de face, et nous nous satisferons de la valeur `GL_FRONT_AND_BACK`. Tout comme pour `glLightfv()`, `nomparam` désigne la propriété qu'on souhaite changer, et 'param' est un tableau contenant la nouvelle valeur à affecter à 'nomparam'. Les valeur de 'nomparam' possibles sont : `GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_EMISSION, GL_SHININESS` (i.e. coefficient de brillance)

5.5 A vous de jouer

En suivant les commentaires inclus dans le code ci-dessous, complétez ce programme permettant d'illuminer de différentes façons un objet.

```
#include <GL/glut.h>
#include <stdlib.h>
#include <math.h>

#define PI 3.14159265

void affichage();
void clavier(unsigned char touche,int x,int y);
void reshape(int x,int y);
void calcTableCosSin();

int angle=45;
float Sin[360],Cos[360];
GLfloat L0pos[]={ 0.0,2.0,-1.0};
GLfloat L0dif[]={ 0.3,0.3,0.8};
GLfloat L1pos[]={ 2.0,2.0,2.0};
GLfloat L1dif[]={ 0.5,0.5,0.5};
GLfloat Mspec[]={0.5,0.5,0.5};
GLfloat Mshiny=50;

int main(int argc,char **argv)
{
  /* initialisation de glut et creation
     de la fenetre */
  glutInit(&argc,argv);
  glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
  glutInitWindowPosition(200,200);
  glutInitWindowSize(400,400);
  glutCreateWindow("light1");

  /* precalcul de la table des sinus et cosinus */
  calcTableCosSin();

  /* Initialisation d'OpenGL */
  glClearColor(0.0,0.0,0.0,0.0);
  glColor3f(1.0,0.0,00);
  glEnable(GL_DEPTH_TEST);

  /* Parametrage des lumieres */
//Specifiez le mode d'eclairage en utilisant la constante GL_SMOOTH
```

```
41      ____________________ ;
42  //Initialisation des parametres du modele de lumiere
43      ____________________________________________ ;
44
45    //Activez les deux lampes
46      ____________________ ;
47      ____________________ ;
48      ____________________ ;
49  //Pour la lumiere  0, initialisez le parametre de diffusion
50      ____________________________________ ;
51  //Pour la lumiere  0, initialisez le parametre de la composante speculaire
52      ____________________________________ ;
53  //Effectuez les memes reglages sur la lumiere 1
54      ____________________________________ ;
55     ____________________________________ ;
56
57    /* Parametrage du materiau */
58  //Initialisez, pour la face avant et arriere, la composante speculaire du
        materiau
59      ____________________________________________ ;
60  //Initialisez, pour la face avant et arriere, le coefficient de brillance
        du materiau
61      ____________________________________________ ;
62
63    /* Mise en place de la perspective */
64    glMatrixMode(GL_PROJECTION);
65    glLoadIdentity();
66    gluPerspective(45.0,1.0,0.1,10.0);
67    glMatrixMode(GL_MODELVIEW);
68
69    /* Mise en place des fonctions de rappel */
70    glutDisplayFunc(affichage);
71    glutKeyboardFunc(clavier);
72    glutReshapeFunc(reshape);
73
74    /* Entree dans la boucle principale */
75    glutMainLoop();
76    return 0;
77  }
78
79
80
81  void affichage(){
82    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
83    glLoadIdentity();
84    gluLookAt(4.5*Cos[angle],2.0,4.5*Sin[angle],0.0,0.0,0.0,0.0,1.0,0.0);
85  //Mise a jour des deux sources lumineuses
86    glLightfv( ____________________________ );
87    glLightfv( ____________________________ );
88  //Affichage de la teiere
89    glutSolidTeapot(1.0);
90    glutSwapBuffers();
91  }
92
93
94
95  void clavier(unsigned char touche,int x,int y)
96  {
97     switch (touche)
98      {
```

```
99      case 'z' :    /* increment de l'angle de position */
100       angle+=2;
101       if (angle>=360)
102   angle-=360;
103       glutPostRedisplay();
104       break;
105     case 'a' :  /* decrement de l'angle de position */
106       angle-=2;
107       if (angle<0)
108   angle+=360;
109       glutPostRedisplay();
110       break;
111     case 'w' : /* Allumer la lampe 0 */
112                          ;
113       glutPostRedisplay();
114       break;
115     case 'x' : /* Eteindre la lampe 0 */
116                          ;
117       glutPostRedisplay();
118       break;
119     case 'c': /* Allumer la lampe 1 */
120                          ;
121       glutPostRedisplay();
122       break;
123     case 'v': /* Eteindre la lampe 0 */
124                          ;
125       glutPostRedisplay();
126       break;
127     case 'm': /* increment reflexion speculaire */
128       Mspec[0]+=0.1;
129       if (Mspec[0]>1)
130   Mspec[0]=1;
131       Mspec[1]+=0.1;
132      if (Mspec[1]>1)
133   Mspec[1]=1;
134       Mspec[2]+=0.1;
135      if (Mspec[2]>1)
136        Mspec[2]=1;
137      //Mise a jour des proprietes du materiau
138                                                    ;
139      glutPostRedisplay();
140       break;
141     case 'l': /* decrement reflexion speculaire du materiau*/
142                ;
143
144
145                ;
146
147            ;
148                ;
149
150            ;
151                                                    ;
152       glutPostRedisplay();
153       break;
154     case 'j': /* incrementation de la brillance */
155       Mshiny-=1;
156       if (Mshiny<0)
157   Mshiny=0;
158       glMaterialf(                                 );
```

```
159         glutPostRedisplay();
160         break;
161       case 'k': /* decrement de la brillance */
162           _________ ;
163           _____________
164       __________ ;
165           ____________________________________________________ ;
166         glutPostRedisplay();
167         break;
168        case 'q' :
169         exit(0);
170       }
171 }
172
173
174
175 void reshape(int x,int y)
176 {
177   if (x<y)
178     glViewport(0,(y-x)/2,x,x);
179   else
180     glViewport((x-y)/2,0,y,y);
181 }
182
183 void calcTableCosSin()
184 {
185   /* calcul du tableau des sinus et cosinus */
186   int i;
187   for (i=0;i<360;i++) {
188     Cos[i]=cos(((float)i)/180.0*PI);
189     Sin[i]=sin(((float)i)/180.0*PI);
190   }
191 }
```