Plan du cours

I. Introduction

II. La bibliothèque graphique GLUT

III. Gestion des évènements du clavier et de la souris

IV. Visualisation et caméra

V. Texture

VI. Eclairage et matériaux

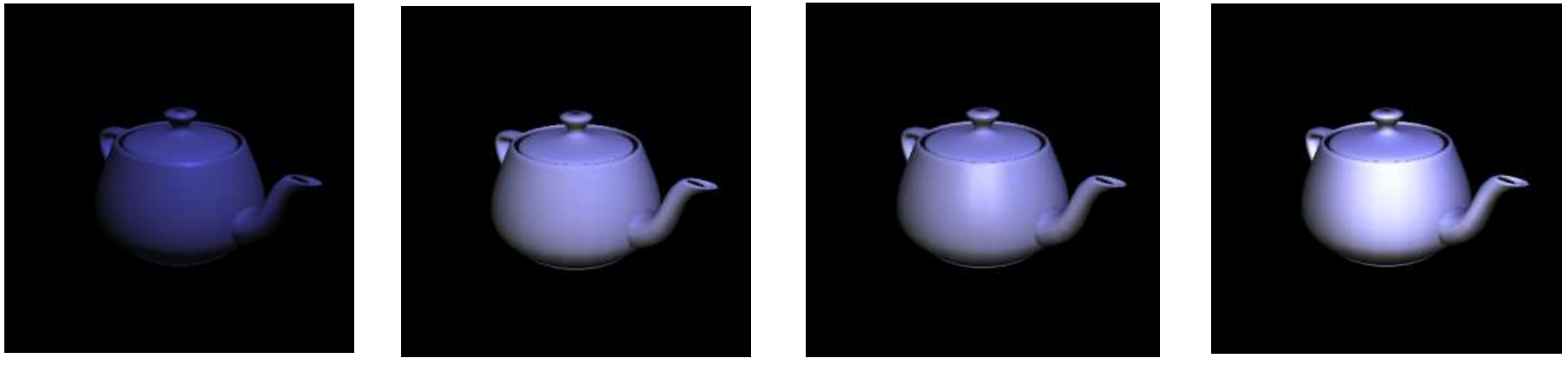
Modèle d'éclairage OpenGL

La perception de la couleur de la surface d'un objet du monde réel dépend de la distribution de l'énergie des photons qui partent de cette surface et qui arrivent aux cellules de la rétine de l'oeil. Chaque objet réagit à la lumière en fonction des propriétés matérielles de sa surface.

Le modèle d'éclairage d'OpenGL considère qu'un objet peut émettre une lumière propre, renvoyer dans toutes les directions la lumière qu'il reçoit, ou réfléchir une partie de la lumière dans une direction particulière, comme un miroir ou une surface brillante.

Les lampes, elles, vont envoyer une lumière dont les caractéristiques seront décrites par leurs trois composantes : ambiante, diffuse ou spéculaire.

OpenGL distingue quatre types de lumières : émissif, ambiant, diffus et spéculaire. Les quatre composants sont calculés indépendamment, puis additionnés.



Modèle d'éclairage OpenGL

- ***Lumière émise Ne concerne que les objets***

Les objets peuvent émettre une lumière propre, qui augmentera leur intensité, mais n'affectera pas les autres objets de la scène.

- ***Lumière ambiante Concerne les objets et les lampes***

C'est la lumière qui a tellement été dispersée et renvoyée par l'environnement qu'il est impossible de déterminer la direction d'où elle provient. Elle semble venir de toutes les directions. Quand une lumière ambiante rencontre une surface, elle est renvoyée dans toutes les directions.

- ***Lumière diffuse Concerne les objets et les lampes***

C'est la lumière qui vient d'une direction particulière, et qui va être plus brillante si elle arrive perpendiculairement à la surface que si elle est rasante. Par contre, après avoir rencontré la surface, elle est renvoyée uniformément dans toutes les directions.

- ***Lumière spéculaire Concerne les objets et les lampes***

La lumière spéculaire vient d'une direction particulière et est renvoyée par la surface dans une direction particulière. Par exemple un rayon laser réfléchi par un miroir.

- ***Brillance Ne concerne que les objets***

Cette valeur entre 0.0 et 128.0 détermine la taille et l'intensité de la tâche de réflexion spéculaire. Plus la valeur est grande, et plus la taille est petite et l'intensité importante.

Les lampes

➢ ***Nombre de lampes***

OpenGL offre d'une part une lampe qui génère uniquement une lumière ambiante (lampe d'ambiance), et d'autre part au moins 8 lampes (GL_LIGHT0, ... , GL_LIGHT7) que l'on peut placer dans la scène et dont on peut spécifier toutes les composantes.

La lampe GL_LIGHT0 a par défaut une couleur blanche, les autres sont noires par défaut.

La lampe GL_LIGHT*i* est allumée par un ***glEnable****(GL_LIGHTi)*

Il faut également placer avant l'instruction ***glEnable****(GL_LIGHTING)* pour indiquer à OpenGL qu'il devra prendre en compte l'éclairage.

➢ ***Couleur des lampes***

Dans le modèle d'OpenGL, la couleur d'une composante de lumière d'une lampe est définie par les pourcentages de couleur rouge, verte, bleue qu'elle émet.

glLightfv permet de spécifier pour chaque lampe tous les paramètres

void glLightfv(GLenum light, GLenum pname, const Glfloat *params)

Voici un exemple complet de définition de la lumière d'une lampe qui génère une lumière ambiante bleue , puis définir pour chaque lampe la couleur et l'intensité des trois composantes *ambiante*, *diffuse* et *Spéculaire*.

```
GLfloat bleu[4] = { 0.0, 0.0, 1.0, 1.0 };
glLightfv(GL_LIGHT0, GL_AMBIENT, bleu);
glLightfv(GL_LIGHT0, GL_DIFFUSE, bleu);
glLightfv(GL_LIGHT0, GL_SPECULAR, bleu);
```

Les lampes

Plus général, pour la lampe ***light*** = *GL_LIGHTi*

void glLightfv(GLenum light, GLenum pname, const Glfloat *params)
void glLightiv(GLenum light, GLenum pname, const GLint *params)

pname = GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_POSITION, GL_SPOT_DIRECTION, GL_SPOT_EXPONENT, GL_SPOT_CUTOFF, GL_CONSTANT_ATTENUATION, GL_LINEAR_ATTENUATION, ou GL_QUADRATIC_ATTENUATION
param = la nouvelle valeur

void glLightf(Glenum light, Glenum pname, GLfloat param)
void glLighti(Glenum light, Glenum pname, GLint param)

pname = GL_SPOT_EXPONENT, GL_SPOT_CUTOFF, GL_CONSTANT_ATTENUATION, GL_LINEAR_ATTENUATION, ou GL_QUADRATIC_ATTENUATION
param = la nouvelle valeur

Les lampes

Le nom du paramètre *pname*	Valeur par défaut *param*	Sens
GL_AMBIENT	(0.0, 0.0, 0.0,1.0)	intensité de la lumière ambiante RGBA
GL_DIFFUSE	(1.0, 1.0, 1.0,1.0)	intensité de lumière diffuse RGBA
GL_SPECULAR	(1.0, 1.0, 1.0,1.0)	intensité de la lumière RGBA spéculaire
GL_POSITION	(0.0, 0.0, 1.0, 0.0)	(*x, y, z, w*) position de la lumière
GL_SPOT_DIRECTION	(0.0, 0.0, -1.0)	(*x, y, z*) direction du projecteur
GL_SPOT_EXPONENT	0.0	projecteur exposant
GL_SPOT_CUTOFF	180,0	angle de coupure du projecteur
GL_CONSTANT_ATTENUATION	1.0	facteur d'atténuation constant
GL_LINEAR_ATTENUATION	0.0	facteur d'atténuation linéaire
GL_QUADRATIC_ATTENUATION	0.0	facteur d'atténuation quadratique

Les lampes

- ***Lampes directionnelles*** Il s'agit d'une lumière qui vient de l'infini avec une direction particulière.

La direction est spécifiée par un vecteur (x,y,z)

```
GLfloat direction[4];
direction[0]=x; direction[1]=y; direction[2]=z;
direction[3]=0.0; /* notez le zéro ici */
glLightfv(GL_LIGHT0, GL_POSITION, direction);
```

- ***Lampes positionnelles*** La lampe se situe dans la scène au point de coordonnées (x,y,z)

```
GLfloat position[4];
position[0]=x; position[1]=y; position[2]=z;
position[3]=1.0; /* notez le un ici */
glLightfv(GL_LIGHT0, GL_POSITION, position);
```

Les lampes

- ***Atténuation de la lumière***

Dans le monde réel, l'intensité de la lumière décroit quand la distance à la source de lumière augmente. Par défaut le facteur d'atténuation d'OpenGL vaut **1** (pas d'atténuation), mais vous pouvez le modifier pour atténuer la lumière des lampes positionnelles.

La formule est : facteur_d_atténuation = 1.0 / (kc + kl*d + kq*d*d), où

d est la distance entre la position de la lampe et le sommet

kc = GL_CONSTANT_ATTENUATION

kl = GL_LINEAR_ATTENUATION

kq = GL_QUADRATIC_ATTENUATION

Exemple de modification du coefficient d'atténuation linéaire pour la lampe 0 :

```
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, 2.0);
```

- ***Lampes omnidirectionnelles et spots***

Par défaut, une lampe illumine l'espace dans toutes les directions. L'autre type de lampe proposé par OpenGL est le spot. Un spot est caractérisé, en plus de sa position, par sa direction, le demi angle du cône de lumière et l'atténuation angulaire de la lumière.

La direction par défaut est {0.0, 0.0, -1.0}, c'est le demi-axe -z.

```
GLfloat direction[]={-1.0, -1.0, 0.0};
glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0); // ce spot éclairera jusqu'à 45° autour de son axe
glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, direction);
glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, 0.5); // coefficient d'atténuation angulaire
```

Les lampes: Modèle d'éclairage

W DRIRA

La notion OpenGL de modèle d'éclairage comporte trois composants:

- L'intensité de la lumière ambiante globale
- Si la position du point de vue est locale à la scène ou si elle doit être considérée comme une distance infinie
- Si les calculs d'éclairage doivent être effectués différemment pour les faces avant et arrière des objet

Les focntions utilisées pour spécifier toutes les propriétés du modèle d'éclairage sont:

void glLightModelf(GLenum pname, GLfloat param)
void glLightModeli(GLenum pname, GLint param)
void glLightModelfv(GLenum pname, const GLfloat *params)
void glLightModeliv(GLenum pname, const GLint *params)

pname	param
GL_LIGHT_MODEL_AMBIENT (fv ou iv seulement)	***par défaut (0.2, 0.2, 0.2, 1.0)*** **Intensité ambiante RGBA de la scène,**
GL_LIGHT_MODEL_LOCAL_VIEWER	***par défaut 0.0 ou GL_FALSE*** **0.0 ou *GL_FALSE* : les reflets des spots sont calculés sur l'axe –z** ***GL_TRUE* : par rapport à la position de l'œil**
GL_LIGHT_MODEL_TWO_SIDE	***par défaut 0.0 ou GL_FALSE*** **0.0 ou *GL_FALSE* : seule la face « front » est calculée** ***GL_TRUE* : les deux faces d'une surface sont calculées**

Les lampes: Modèle d'éclairage

Il faut indiquer avec ***glLightModel*()*** si les calculs d'éclairage se font de la même façon ou non sur l'envers et l'endroit des faces des objets. Il faut également indiquer si OpenGL doit considérer pour ses calculs d'éclairage que l'oeil est à l'infini ou dans la scène.

Si l'oeil est dans la scène, il faut calculer l'angle entre le point de vue et chacun des objets

Si l'oeil est à l'infini, cet angle est ignoré ce qui est moins réaliste, mais moins coûteux en calculs. C'est le choix par défaut. Il est modifié par l'instruction

glLightModeli*(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE)*

C'est avec cette même fonction que l'on définit la couleur de la lampe d'ambiance

glLightModelfv*(GL_LIGHT_MODEL_AMBIENT, couleur)*

Exemple :

```
GLfloat lmodel_ambient [] = {0.2, 0.2, 0.2, 1.0};
glLightModelfv (GL_LIGHT_MODEL_AMBIENT, lmodel_ambient);
```

Les matériaux

- ***Couleur d'un matériau***

Dans le modèle d'OpenGL, la couleur que l'on perçoit d'un objet dépend de la couleur propre de l'objet et de la couleur de la lumière qu'il reçoit.
Par exemple, un objet rouge renvoie toute la lumière rouge qu'il reçoit et absorbe toute la lumière verte et bleue qu'il reçoit.

- Si cet objet est éclairé par une lumière blanche (composé en quantités égales de rouge, vert et bleu), il ne renverra que la lumière rouge et apparaîtra donc rouge.
- Si cet objet est éclairé par une lumière verte, il apparaîtra noir, puisqu'il absorbe le vert et n'a pas de lumière rouge à réfléchir.

- ***Propriétés matérielles d'un objet***

Les propriétés matérielles d'un objet sont celles qui ont été évoquées dans la partie Modèle d'éclairage OpenGL : la lumière émise, la réflexion ambiante, diffuse et spéculaire du matériau dont est fait l'objet. Elles sont déterminées par des instructions :

glMaterial*(GLenum face, GLenum pname, TYPE param)*

Où ***face*** peut être GL_FRONT, GL_BACK ou GL_FRONT_AND_BACK pour indiquer à quelle face de l'objet le matériau doit être appliqué
Et ***pname*** peut être GL_AMBIENT, GL_DIFFUSE, GL_AMBIENT_AND_DIFFUSE, GL_SPECULAR, GL_SHININESS, ou GL_EMISSION.

Les matériaux

glMaterial*(GLenum face, GLenum pname, TYPE param)

Le nom du paramètre ***pname***	**Valeur par défaut** ***param***	**Sens**
GL_AMBIENT	(0,2, 0,2, 0,2, 1,0)	couleur ambiante du matériau
GL_DIFFUSE	(0,8, 0,8, 0,8, 1,0)	couleur diffuse du matériau
GL_AMBIENT_AND_DIFFUSE		couleur ambiante et diffuse du matériau
GL_SPECULAR	(0,0, 0,0, 0,0, 1,0)	couleur spéculaire du matériau
GL_SHININESS	0.0	exposant spéculaire
GL_EMISSION	(0,0, 0,0, 0,0, 1,0)	couleur émissive du matériau
GL_COLOR_INDEXES	(0,1,1)	indices de couleur ambiante, diffuse et spéculaire

Les matériaux

Une autre technique permettant de réduire les coûts de performance associés à la modification des propriétés du matériau consiste à utiliser **glColorMaterial ()** .

void glColorMaterial (GLenum face, mode GLenum);

Fait en sorte que la ou les propriétés de matériau spécifiées par le mode de la ou des faces de matériau spécifiées par face suivent la valeur de la couleur actuelle à tout moment. Une modification de la couleur actuelle (à l'aide de glColor * ()) met immédiatement à jour les propriétés de matériau spécifiées.

face peut être GL_FRONT, GL_BACK ou GL_FRONT_AND_BACK (valeur par défaut).

mode peut être GL_AMBIENT, GL_DIFFUSE, GL_AMBIENT_AND_DIFFUSE (valeur par défaut), GL_SPECULAR ou GL_EMISSION. À un moment donné, un seul mode est actif. glColorMaterial () n'a aucun effet sur l'éclairage à indice de couleur.

- ***Combinaison des coefficients***
  - Pour une lampe, les coefficients RVB correspondent à un pourcentage de l'intensité totale pour chaque couleur. Par exemple R=1.0, V=1.0, B=1.0 correspond au blanc de plus grande intensité, alors que R=0.5, V=0.5, B=0.5 correspond à un blanc d'intensité moitié moins grande, qui est donc un gris.
  - Pour un objet, les nombres correspondent à la proportion de la lumière renvoyée pour chaque couleur. Si une lampe qui a comme coefficients (LR, LV, LB) éclaire un objet (OR, OV, OB), la couleur perçue sera (LR*OR, LV*OV, LB*OB).
  - La combinaison de deux lampes de coefficients (R1, V1, B1) et (R2, V2, B2) produit une lumière (R1+R2, V1+V2, B1+B2) (et les valeurs supérieures à 1 sont ramenées à 1).
- ***Transparence:*** obtenue en indiquant pour un objet une couleur diffuse RVBA ou la valeur A sur le canal alpha est strictement plus petite que 1.

Example


```
#include <windows.h>
#include <gl\glut.h>
float left;
float right;
void Display() {
glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glPushMatrix();
        glRotated(35, 1, 1, 0);
        glBegin(GL_QUADS);
                glColor3d(1,0,0);
                glVertex3d(-50,50,50);  glVertex3d(-50,-50,50); glVertex3d(50,-50,50); glVertex3d(50,50,50);
                glVertex3d(50,50,-50); glVertex3d(50,-50,-50);glVertex3d(-50,-50,-50);glVertex3d(-50,50,-50);
                glColor3d(0,1,0);
                glVertex3d(50,50,50); glVertex3d(50,-50,50); glVertex3d(50,-50,-50); glVertex3d(50,50,-50);
                glVertex3d(-50,50,50); glVertex3d(-50,50,-50); glVertex3d(-50,-50,-50); glVertex3d(-50,-50,50);
                glColor3d(0,0,1);
                glVertex3d(-50,-50,50); glVertex3d(-50,-50,-50); glVertex3d(50,-50,-50); glVertex3d(50,-50,50);
                glVertex3d(-50,50,50); glVertex3d(50,50,50); glVertex3d(50,50,-50); glVertex3d(-50,50,-50);
        glEnd();
        glPopMatrix();
        glutSwapBuffers();
}
```

Example (suite 1)


```
void Reshape(int w, int h)
{
        if (h == 0) h = 1;
        glViewport(0,0,w,h);

         if (w<=h)
        {       left  = 300 * h/w;
                right = 300;
        }
        else
        {       left = 300;
                right = 300 * w/h;
        }
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho( -left/2, left/2, -right/2, right/2, 100, -100);
}
```

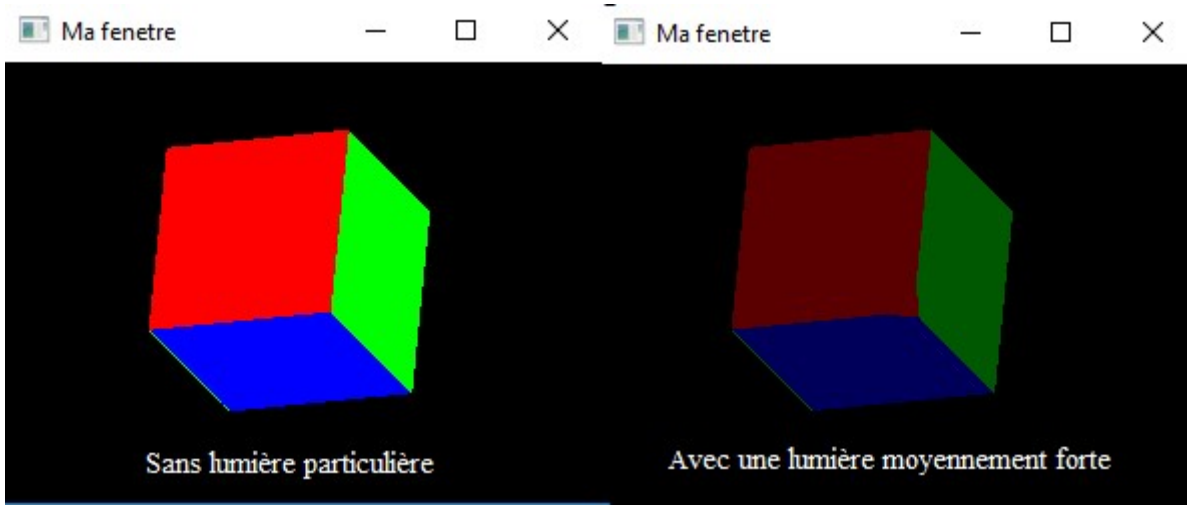
Example (suite 2)

W DRIRA

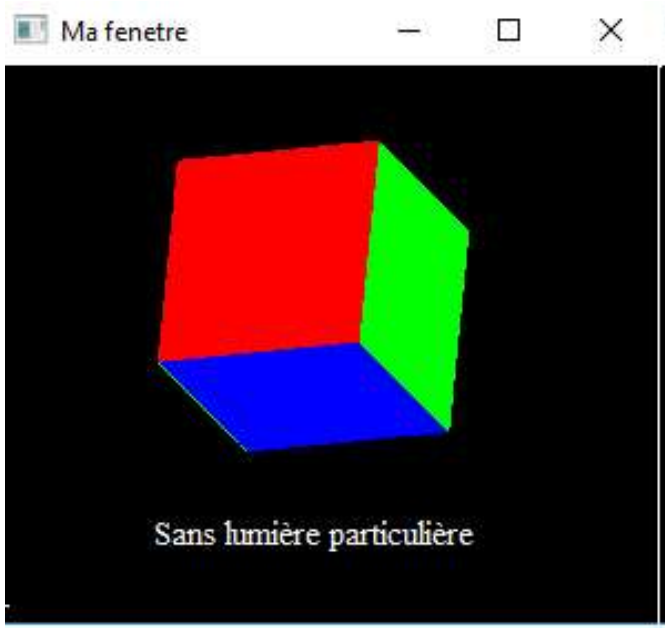
```
void init(){
    glShadeModel(GL_SMOOTH);
        glFrontFace(GL_CW);
        glEnable(GL_DEPTH_TEST);
        glMatrixMode(GL_MODELVIEW);  glLoadIdentity(); gluLookAt (0,0,-30,0,0,0,0,1,0);
        GLfloat ambient[] = {0.15,0.15,0.15,1.0};  //différents paramètres
        GLfloat diffuse[] = {0.5,0.5,0.5,1.0};
        GLfloat light0_position [] = {0.0, -10.0, 0.0, 0.0}; //w = 0 lumière placée a l'infini
        GLfloat specular_reflexion[] = {0.8,0.8,0.8,1.0};
        GLubyte shiny_obj = 128;
        glEnable(GL_LIGHTING);
        glLightfv(GL_LIGHT0,GL_AMBIENT,ambient); //les différents paramètres
        glLightfv(GL_LIGHT0,GL_DIFFUSE,diffuse);
        glLightfv(GL_LIGHT0,GL_POSITION,light0_position);
        glEnable(GL_LIGHT0);
        glEnable(GL_COLOR_MATERIAL);   //spécification de la réflexion sur les matériaux
        glColorMaterial(GL_FRONT,GL_AMBIENT_AND_DIFFUSE);
        glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT,ambient);
        glMaterialfv(GL_FRONT_AND_BACK,GL_DIFFUSE,diffuse);
        glMaterialfv(GL_FRONT_AND_BACK,GL_SPECULAR,specular_reflexion);//on peut le faire avant chaque objet
        glMateriali(GL_FRONT_AND_BACK,GL_SHININESS,shiny_obj); //si on veut qu'ils aient des caractéristiques}
```

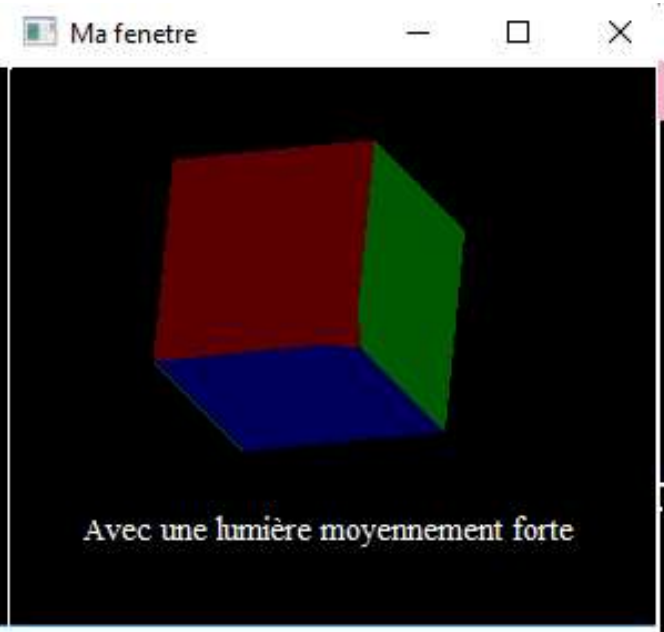
Example (suite 3)

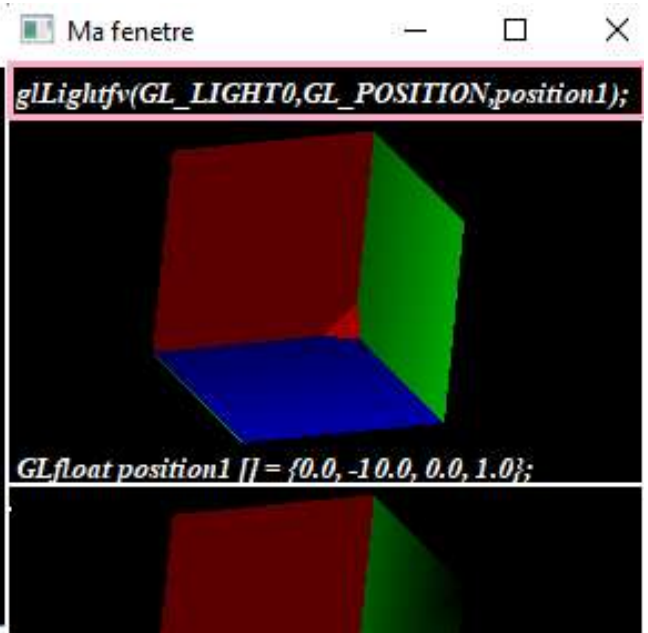

```
int main(int argc, char *argv[])
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE |GLUT_RGBA | GLUT_DEPTH);
   glutInitWindowPosition(300,200);
   glutCreateWindow("Ma fenetre");
   init ();
   glutDisplayFunc(Display);
   glutReshapeFunc(Reshape);
   glClearColor(0,0,0,0);
   glutMainLoop();
   return 0;

}
```

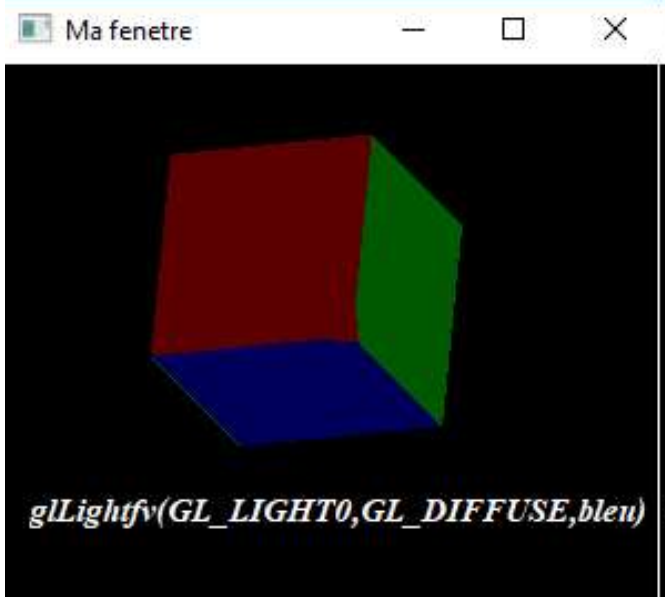


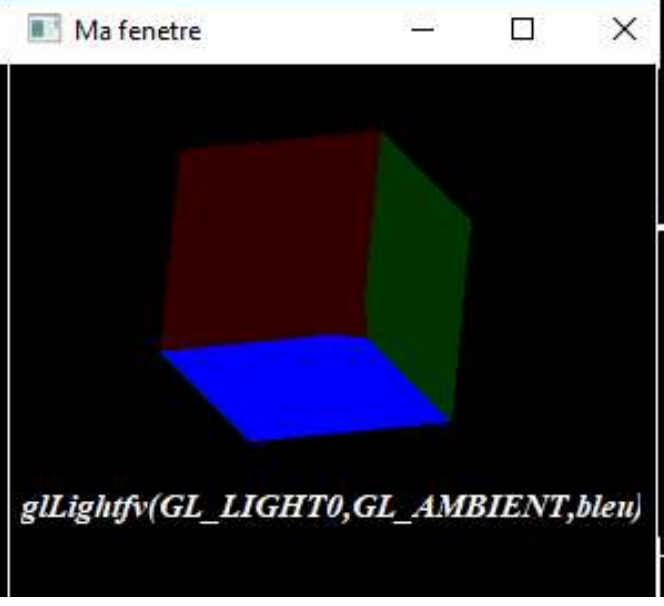
Examples

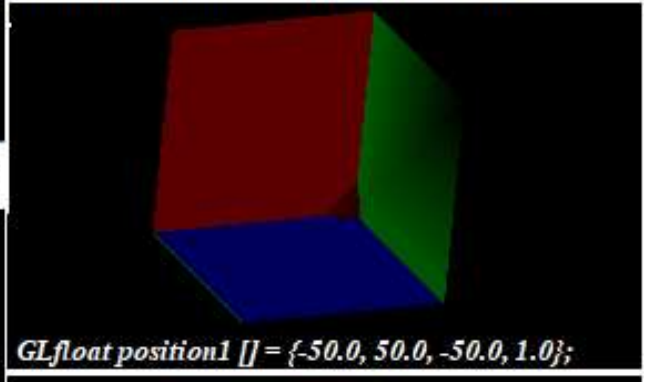

Ma fenetre
Sans lumière particulière

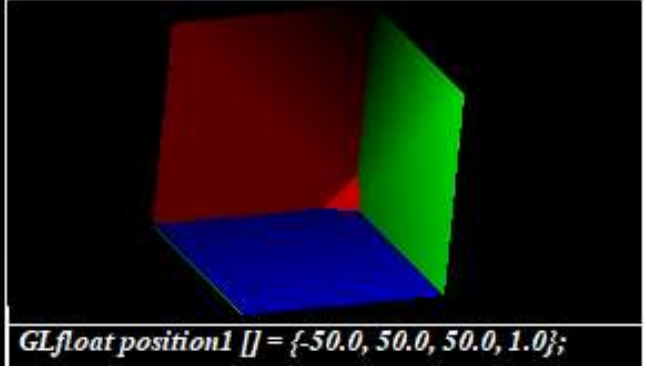

Ma fenetre
Avec une lumière moyennement forte


Ma fenetre
glLightfv(GL_LIGHT0,GL_POSITION,position1);
GLfloat position1 [] = {0.0, -10.0, 0.0, 1.0};


Ma fenetre
glLightfv(GL_LIGHT0,GL_DIFFUSE,bleu)


Ma fenetre
glLightfv(GL_LIGHT0,GL_AMBIENT,bleu)


GLfloat position1 [] = {-50.0, 50.0, -50.0, 1.0};


GLfloat position1 [] = {-50.0, 50.0, 50.0, 1.0};

Example …

Ajouter à l'exemple précédent une deuxième source de lumière de projecteur

```
GLfloat light1_ambient [] = {0.0, 2.0, 0.0, 2.0};
GLfloat light1_diffuse [] = {1.0, 1.0, 1.0, 1.0};
GLfloat light1_specular [] = {1.0, 1.0, 1.0, 1.0};
GLfloat light1_position [] = {-2.0, 2.0, 1.0, 1.0};
GLfloat spot_direction [] = {-1.0, -1.0, 0.0};

glLightfv (GL_LIGHT1, GL_AMBIENT, light1_ambient);
glLightfv (GL_LIGHT1, GL_DIFFUSE, light1_diffuse);
glLightfv (GL_LIGHT1, GL_SPECULAR, light1_specular);
glLightfv (GL_LIGHT1, GL_POSITION, light1_position);
glLightf (GL_LIGHT1, GL_CONSTANT_ATTENUATION, 1.5);
glLightf (GL_LIGHT1, GL_LINEAR_ATTENUATION, 0.5);
glLightf (GL_LIGHT1, GL_QUADRATIC_ATTENUATION, 0.2);

glLightf (GL_LIGHT1, GL_SPOT_CUTOFF, 45);
glLightfv (GL_LIGHT1, GL_SPOT_DIRECTION, spot_direction);
glLightf (GL_LIGHT1, GL_SPOT_EXPONENT, 2);

glEnable (GL_LIGHT1);
```