

# **OpenGL 3.0**

## **Chapitre 4**

Texte

Matériaux et éclairage

C.Turrier

15 décembre 2020

# Sommaire

|                         |   |
|-------------------------|---|
| 1) Écrire du texte..... | 3 |
| 2) Éclairage.....       |   |
| 3) Matériaux.....       |   |
| 4) Exemples.....        |   |

# 1) Écrire du texte

Le programme suivant permet d'écrire du texte dans la fenêtre de programme OpenGL qui contient une scène 3D.

```
/*
 * gl-test04.cpp  OpenGL - écriture de texte
 * compilation :
 * g++ gl-test04.cpp -o test -lglut -lGLU -lGL
 * gcc gl-test04.cpp -o test -lglut -lGLU -lGL
 */
#include <GL/glut.h>
#include <string.h>

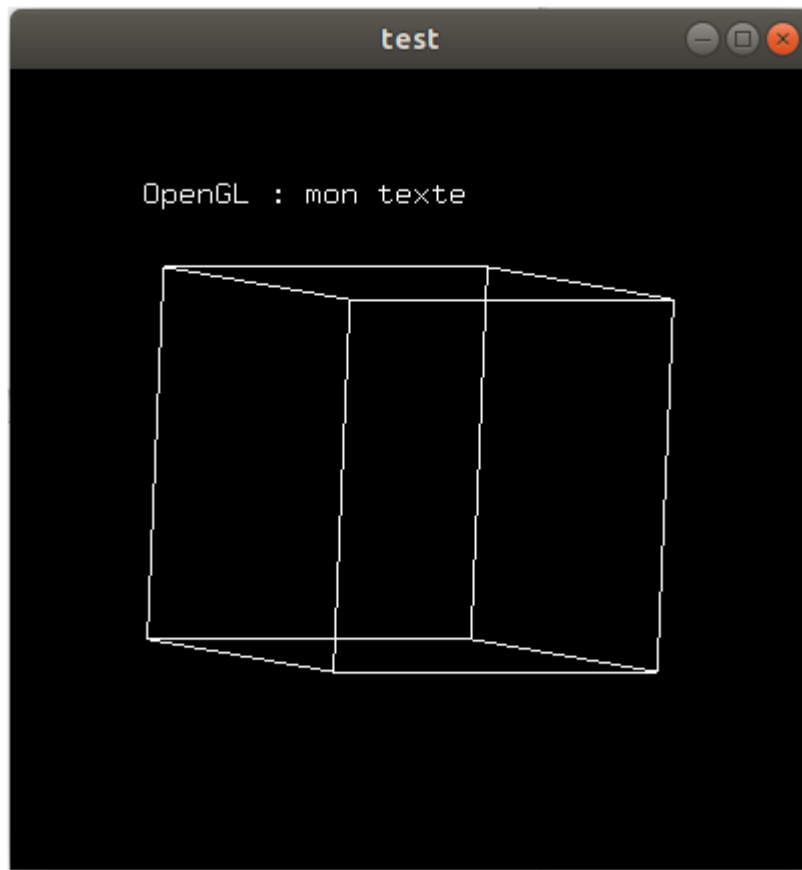
char texte[40+1];
void ecrire(char *texte, int x, int y, int z )
{
    int longueur, i;
    glRasterPos3f(x, y,z);
    longueur = (int) strlen(texte);
    for (i = 0; i < longueur; i++)
    {glutBitmapCharacter(GLUT_BITMAP_9_BY_15, texte[i]);}
}

void affiche(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity();
    glColor3f(1,1,1);
    strcpy(texte, "OpenGL->écriture de texte");
    ecrire(texte, -1.3,1.3,0);
    glRotatef(30, 0.0, 1.0, 0.0);
    glRotatef(5, 1.0, 0.0, 0.0);
    glutWireCube(1.4);
    glutSwapBuffers();
}

void dimensionne(int w,int h)
{
    glViewport(0,0,w,h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-1.5, 1.5, -1.5, 1.5, -20.0, 1.5);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
    glutInitWindowSize(400,400);
    glutInitWindowPosition(100,100);
```

```
glutCreateWindow("test");  
glutDisplayFunc(affiche);  
glutReshapeFunc(dimensionne);  
glClearColor(0.0,0.0,0.0,0.0);  
glutMainLoop();  
return 0;  
}
```



## 2) Éclairage

Pour éclairer une scène 3D on utilise une ou plusieurs sources lumineuses. Ces source lumineuses pouvant être de différentes sortes (ponctuelles, spot, directionnelles,...) et sont susceptibles de produire un éclairage de lumière incidente ambiante, diffuse ou spéculaire.

### 2.1) Éclairage de lumière ambiante

Un éclairage constitué d'une lumière incidente ambiante est un éclairage dont la lumière est tellement dispersée par l'environnement que sa direction est impossible à déterminer. Elle semble provenir de toutes les directions.

### 2.2) Éclairage de lumière diffuse

Un éclairage constitué d'une lumière incidente diffuse est un éclairage dont la lumière vient d'une direction. Lorsqu'elle atteint la surface d'un objet mat, elle est dispersé de manière égale dans toutes les directions

### 2.3) Éclairage de lumière spéculaire

Un éclairage constitué d'une lumière incidente spéculaire est un éclairage dont la lumière provient d'une direction particulière. Lorsqu'elle atteint la surface d'un objet brillant, elle se réfléchit sur sa surface.

Lorsqu'il est éclairé, un objet 3D réfléchit la lumière qui l'éclaire. L'apparence que prend cet objet dépend de la réflectance du matériau qui le constitue.

La réflectance (ou facteur de réflexion)  $\rho$  d'un matériau est la proportion entre la lumière  $L_r$ , réfléchié par la surface de ce matériau, et la lumière  $L_i$  incidente qui éclaire ce matériau.

### 3) Matériaux

Soit un objet, constitué d'un certain matériau (bois, fer, verre ...), éclairé par une lumière incidente d'intensité  $I$

La lumière réfléchie par cet objet est composée des trois lumières suivantes (**modèle de Phong**) :

- la lumière réfléchie ambiante d'intensité  $I_{ra}$
- la lumière réfléchie diffuse d'intensité  $I_{rd}$
- la lumière réfléchie spéculaire d'intensité  $I_{rs}$

#### 3.1) Lumière réfléchie ambiante

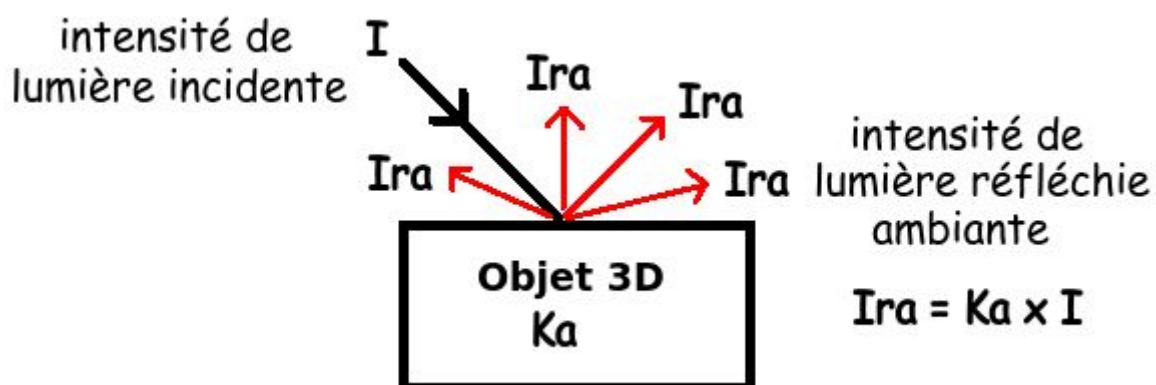
L'intensité de lumière réfléchie ambiante  $I_{ra}$  d'un objet est la même dans toutes les directions. Son intensité dépend uniquement de l'intensité  $I$  de la lumière incidente éclairant l'objet et du coefficient  $K_a$  de réflexion de lumière ambiante de cet objet.

$I$  : intensité de lumière incidente

$K_a$  : coefficient de réflexion de lumière ambiante

$I_{ra}$  : intensité de lumière réfléchie ambiante

$$I_{ra} = K_a \times I$$



### 3.2) Lumière réfléchie diffuse

L'intensité de lumière réfléchie diffuse **Ird**, d'un objet 3D constitué d'un matériau mat, est la même dans toutes les directions.

Son intensité dépend uniquement de l'intensité **I** de la lumière incidente éclairant l'objet, de l'angle  **$\theta$**  de la lumière incidente (par rapport à la normale à la surface du matériau) et du coefficient **Kd** de réflexion de lumière diffuse de cet objet.

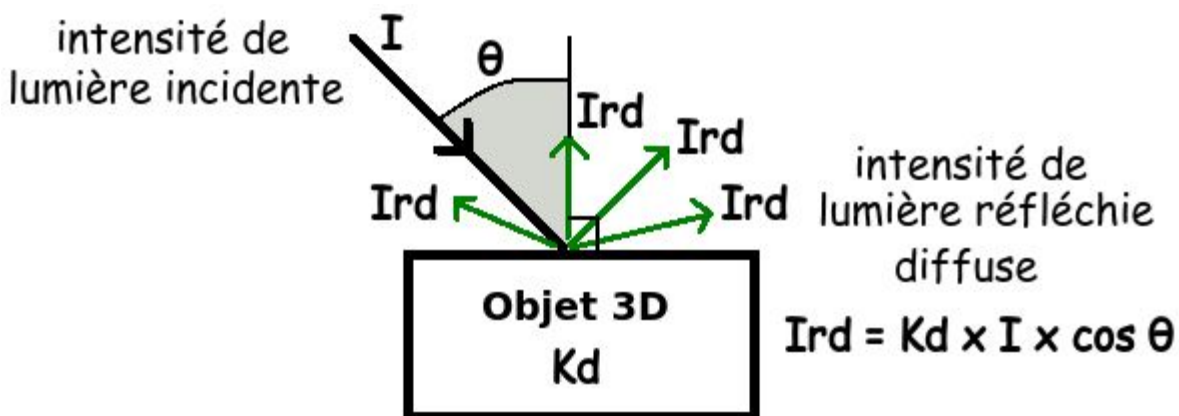
**I** : intensité de lumière incidente

**Kd** : coefficient de réflexion de lumière diffuse

**Ird** : intensité de lumière réfléchie diffuse

**$\theta$**  : angle entre la lumière incidente (**I**) et la normale au matériau

$$Ird = Kd \times I \times \cos \theta$$



### 3.3) Lumière réfléchie spéculaire

L'intensité de lumière réfléchie spéculaire  $I_{rs}$ , d'un objet 3D constitué d'un matériau réfléchissant, n'est pas la même dans toutes les directions. Son intensité dépend de l'intensité  $I$  de la lumière incidente éclairant l'objet, de l'angle  $\beta$  de l'observateur (par rapport au rayon de lumière réfléchi) et du coefficient  $K_s$  de réflexion de lumière spéculaire de cet objet.

Le modèle Phong utilise également un nombre  $n$  qui permet de faire décroître plus ou moins l'intensité de la lumière réfléchie spéculaire en fonction de l'angle  $\beta$ .

$I$  : intensité de lumière incidente

$K_s$  : coefficient de réflexion de lumière spéculaire

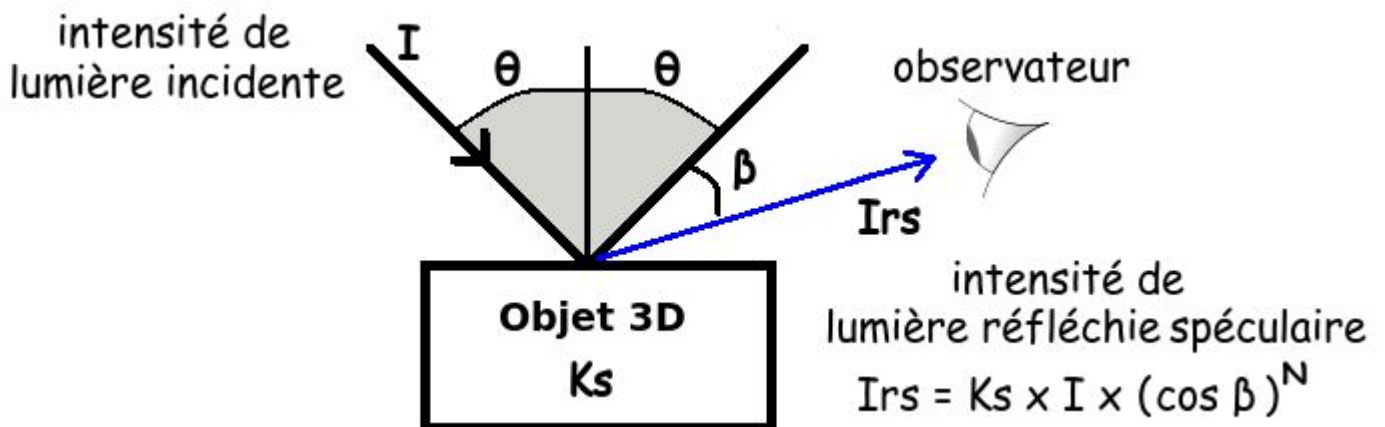
$I_{rs}$  : intensité de lumière réfléchie spéculaire

$\theta$  : angle entre la lumière incidente ( $I$ ) et la normale au matériau (permet d'identifier le rayon réfléchi)

$\beta$  : angle entre le rayon réfléchi et l'observateur matériau

$N$  : constante permettant de choisir une décroissance plus ou moins rapide de l'intensité de la lumière réfléchie spéculaire en fonction de  $\beta$

$$I_{rs} = K_s \times I \times (\cos \beta)^N$$



Dans le modèle de Phong, l'intensité totale  $I_r$  de la lumière réfléchie par un objet éclairé par une lumière incidente d'intensité  $I$  est égale à la somme des trois composantes précédentes.

$$I_r = I_{ra} + I_{rd} + I_{rs}$$

$$I_r = K_a \times I + K_d \times I \times \cos \theta + K_s \times I \times (\cos \beta)^N$$

## 4) Matériaux

### 4.1) Exemple 1

Mise en œuvre d'une lumière ambiante pour éclairer une sphère.

```
*****
* gl-test04.cpp  OpenGL - écriture de texte
* compilation :
* g++ gl-test04b.cpp -o test -lglut -lGLU -lGL
* gcc gl-test04b.cpp -o test -lglut -lGLU -lGL
*****/
#include <GL/freeglut.h>
void initLumiere(void)
{
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glEnable(GL_LIGHTING);
    float lightPos[] = { 0.0, 0.0, 3, 1.0 };
    float globAmb[] = { 0.2, 0.2, 0.2, 1.0 };
    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, globAmb);
    glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);
    glEnable(GL_LIGHT0);
}

void affiche(void)
{
    //propriété ambiante et diffuse du matériau
    float matAD[] = { 0.8, 0.0, 0.0, 1.0 };
    glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity();
    gluLookAt(0.0, 0.0, -3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, matAD);
    glEnable(GL_CULL_FACE);
    glCullFace(GL_BACK);
    glutSolidSphere(1.0, 50, 50);
    glDisable(GL_CULL_FACE);
    glutSwapBuffers();
}

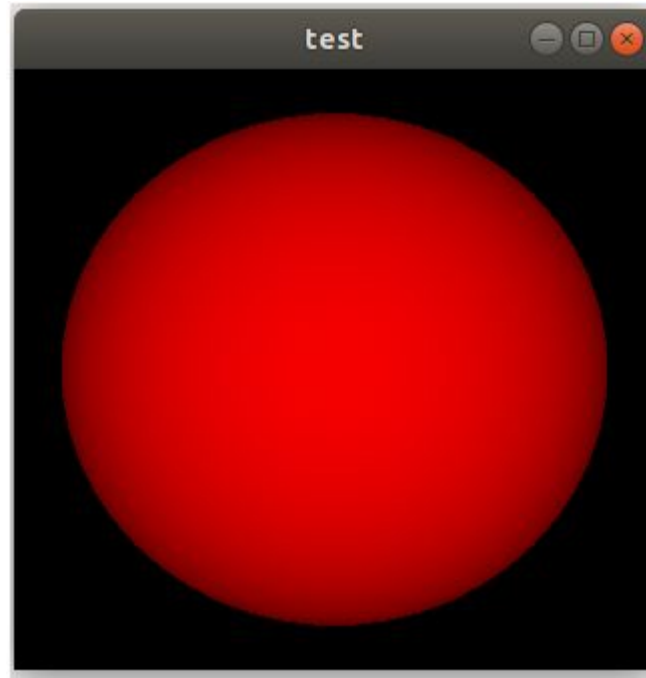
void dimensionne(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, w/h, 1.0, 20.0);
    glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
```

```

glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA );
glutInitWindowSize(500, 500);
glutInitWindowPosition(100, 100);
glutCreateWindow("test");
glutDisplayFunc(affiche);
glutReshapeFunc(dimensionne);
initLumiere();
glutMainLoop();
return 0;
}

```



- `void initLumiere(void)`  
initialise l'éclairage de la scène
- `glClearColor(0.0, 0.0, 0.0, 0.0);`  
fond d'écran noir
- `glEnable(GL_LIGHTING);`  
active l'éclairage de OpenGL  
Les fonctions `glEnable()` et `glDisable()` activent et désactivent la fonction OpenGL passée en paramètre
- `float lightPos[] = { 0.0, 0.0, 3, 1.0 };`  
position x,y,z de la source lumineuse
- `float globAmb[] = { 0.2, 0.2, 0.2, 1.0 };`  
couleur rvg de la lumière ambiante

● `glLightfv(GL_LIGHT0, GL_POSITION, lightPos);`  
initialise la lumière `GL_LIGHT0` et la position de la source de cette lumière

La fonction `glLightfv()` permet de définir une lumière et d'initialiser les valeurs de certains paramètres associés à cette lumière.

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

`light`: identifiant d'une lumière. Le nombre de lumières possibles dépend de l'implémentation, mais au moins huit lumières sont prises en charge. Ils sont identifiés par des noms symboliques de la forme `GL_LIGHTi` où `i` est une valeur: 0 à `GL_MAX_LIGHTS - 1`.

`pname`: paramètre de la lumière (`GL_AMBIENT`, `GL_DIFFUSE`, `GL_SPECULAR`, `GL_POSITION`, `GL_SPOT_DIRECTION...`)

`params`: spécifie la valeur avec laquelle le paramètre `pname` sera initialisé.

● `glLightModelfv(GL_LIGHT_MODEL_AMBIENT, globAmb);`  
initialise l'intensité des composantes RVBA pour le modèle de lumière ambiant

La fonction `glLightModelfv()` définit les valeurs d'initialisation pour le paramètre passé en argument.

```
void glLightModelfv(  
GLenum pname,  
const GLfloat *params );
```

`pname`: paramètre passé en argument (`GL_LIGHT_MODEL_AMBIENT`, `GL_LIGHT_MODEL_LOCAL_VIEWER`, `GL_LIGHT_MODEL_TWO_SIDE`)

`params`: spécifie la valeur avec laquelle le paramètre `pname` sera initialisé.

La fonction `glLightModeli` définit les paramètres du modèle d'éclairage.

● `glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);`  
initialise le mode d'éclairage double face (recto-verso) des faces des objets 3D

La fonction `glLightModeli()` définit les valeurs d'initialisation pour le paramètre passé en argument.

```
void glLightModeli(  
GLenum pname,  
GLint param );
```

`pname`: paramètre passé en argument (`GL_LIGHT_MODEL_LOCAL_VIEWER`, `GL_LIGHT_MODEL_TWO_SIDE`)

`param`: spécifie la valeur avec laquelle le paramètre `pname` sera initialisé.

● `glEnable(GL_LIGHT0);`

active la lumière `GL_LIGHT0` précédemment définie

● `void affiche(void)`

affiche la scène 3D à l'écran, dans la fenêtre de programme OpenGL.

{

`float matAD[] = { 0.8, 0.0, 0.0, 1.0 };`

définition des propriétés RVB ambiante et diffuse du matériau

● `glClear(GL_COLOR_BUFFER_BIT );`

La fonction `glClear()` met à zéro les buffers utilisés par OpenGL

Le paramètre `GL_COLOR_BUFFER_BIT` indique que les buffers courants sont activés pour le dessin des couleurs ;

● `glLoadIdentity();`

La fonction `glLoadIdentity ()` est appelée, avant d'effectuer une transformation (`glTranslate`, `glRotate...`), lorsqu'on souhaite réinitialiser la matrice de transformation à son état d'origine (matrice identité).

On fait cet appel lorsqu'on souhaite qu'une transformation soit effectuée à l'aide de la matrice de transformation dans son état d'origine (et non pas à l'aide de la matrice de transformation dans son état courant) .

● `gluLookAt(0.0, 0.0, -3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);`

La fonction `gluLookAt()` définit une transformation de vue (par un observateur). L'observateur est ici placé en -3 sur l'axe z et a son regard dirigé vers le point (0,0,0), l'axe y est considéré comme la verticale de la scène.

```
void gluLookAt(  
GLdouble eyex,  
GLdouble eyey,  
GLdouble eyez,  
GLdouble centerx,  
GLdouble centery,  
GLdouble centerz,  
GLdouble upx,  
GLdouble upy,  
GLdouble upz );
```

`eyex, eyey, eyez` : position de l'observateur

`centerx, centery, centerz` : point pointé par l'observateur

`upx, upy, upz` : axe considéré comme étant la verticale

La fonction `gluLookAt` crée une matrice de vue (par un observateur), en prenant en compte un point de référence indiquant le centre de la scène et un vecteur indiquant la direction vers le haut. Du point de vue

mathématique, gluLookAt() génère une matrice qui multiplie la matrice courante de OpenGL.

● glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, matAD);  
spécifie les paramètres de matériau float matAD[] = { 0.8, 0.0, 0.0, 1.0 };  
pour le modèle d'éclairage GL\_AMBIENT\_AND\_DIFFUSE

La fonction glMaterialfv permet de spécifier les paramètres des matériaux pour le modèle d'éclairage considéré.

```
void glMaterialfv(  
GLenum face,  
GLenum pname,  
const GLfloat *params );
```

face: définit les faces qui seront colorées par la lumière (GL\_FRONT, GL\_BACK ou GL\_FRONT\_AND\_BACK). Généralement on choisit GL\_FRONT

pname: définit le paramètre de matériau considéré (GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR, GL\_EMISSION, GL\_SHININESS, GL\_AMBIENT\_AND\_DIFFUSE, GL\_COLOR\_INDEXES) pour les faces prises en compte.

params: spécifie la valeur avec laquelle le paramètre pname sera initialisé.

● glutSolidSphere (1.0, 50, 50);  
crée une sphère solide de 1 de rayon

La fonction glutSolidSphere() permet de dessiner une sphère centrée à l'origine des coordonnées de modélisation. La sphère est subdivisée autour et le long de l'axe Z.

```
void glutSolidSphere(  
GLdouble radius,  
GLint slices,  
GLint stacks);
```

radius: rayon de la sphere

slices: nombre de subdivisions sur l'axe z (longitude)

stacks: nombre de subdivisions sur l'axe z (latitude)

● glutSwapBuffers();  
Bascule le buffer mémoire vidéo à l'écran; le double buffering ayant été initialisé par glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA );