الجلسة السابعة

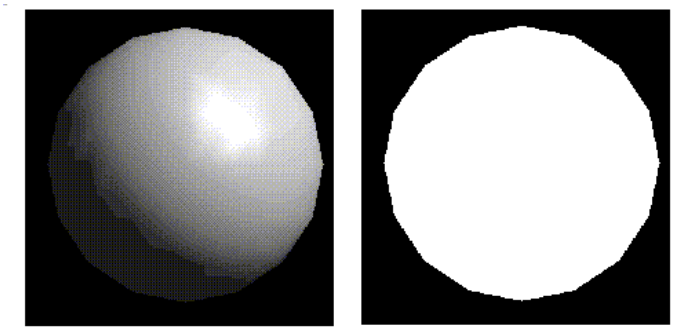
العناصر الهندسية ثلاثية الأبعاد:

- هناك العديد من العناصر الهندسية 3D الممكن رسمها ضمن OpenGL باستخدام المكتبة glut. يعرض الجدول التالي هذا العناصر من الوسائط الخاصة بها:

الأمر	الوظيفة	الوسائط
glutWireSphere(radius,slices,stacks) glutSolidSphere(radius,slices,stacks)	رسم كرة سلكية أو مصمتة	نصف القطر (radius)،خطوط الطول ، دوائر العرض
glutWireCube(size) glutSolidCube(size)	رسم مكعب سلكي أو مصمت	طول الضلع (size)
glutSolidTorus(innerRadius, outerRadius, nsides, rings);	رسم طارة سلكية أو مصمتة	نصف قطر السماكة inner radius ونصف القطر الخارجي outer radius
glutSolidCone(base,height, slices,stacks); glutWireCone(base,height, slices,stacks);	رسم مخروط سلكي أو مصمت	نصف قطر قاعدة المخروط base والارتفاع height
glutWireTeapot(size) glutSolidTeapot(size)	رسم إبريق شاي سلكي أو مصمت	نصف قطر القاعدة size

الإضاءة

- معظم العناصر في الحقيقة لا يظهر شكلها ثلاثي الأبعاد إلا بتطبيق إضاءة عليها.



- تصدر الفوتونات من مصادر الضوء لتسقط على العناصر المضاءة. يُمتص بعض هذه الفوتونات ويعكس بعضها من قبل السطح المضاء. حسب خصائص السطح يتم عكس الفوتونات (في اتجاهات معينة أو في جميع الاتجاهات).

- تعامل OpenGL الإضاءة بتقسيمها إلى مكوناتها اللونية RGB. وبالتالي يتحدد لون مصادر الضوء حسب كمية RGB التي تصدرها. وتتحدد مادة السطح العاكس حسب نسبة المكونات RGB القادمة إلى السطح والتي تنعكس في اتجاهات معينة.
- تأخذ OpenGl بعين الاعتبار تقسيم الإضاءة إلى أربعة مكونات مستقلة:

1. Emitted: أبسط أنواع الإضاءة. يصدر من العنصر ولا يتأثر بأي مصدر إضاءة.
2. Ambient: يأتي من جميع الاتجاهات. الإضاءة الخلفية في الغرفة هي مكون Ambient، يصل الضوء Ambient إلى العين بعد مروره على عدة أسطح.
3. Diffuse: يأتي من مصدر وحيد وعندما يصطدم بالسطح ، ينتشر بشكل متساو في جميع الاتجاهات لذلك يظهر براقاً بشكل متساوٍ.
4. Specular: يأتي من اتجاه معين ويرتطم بالسطح في اتجاه مفضل. يؤدي سقوط شعاع ليزري على مرآة عالية الجودة إلى إنتاج انعكاس Specular 100%.

ألوان المواد

- تعكس كرة حمراء جميع ألوان الإضاءة الحمراء القادمة إليها وتمتص ألوان الإضاءة الأخرى الخضراء والزرقاء. إذا وُضعت كرة حمراء أمام مصدر ضوء أخضر فستظهر سوداء (سيمتص الضوء الأخضر ولا يوجد مصدر ضوء أحمر فلن يكون هناك إضاءة) .
- تمتلك المواد ألوان ambient وdiffuse وspecular مختلفة.
- يدمج انعكاس ambient للمادة مع المكون ambient لكل مصدر ضوء قادم وهكذا بالنسبة لـ diffuse وspecular.
- الضوء الواصل للعين يتجاهل جميع المؤثرات الأخرى وعلى افتراض أن مكون الضوء (LR,LG,LB) والمادة تمتلك المكونات (MR,MG,MB) فسيكون الضوء الواصل للعين (LR*MR, LG*MG, LB*MB) .
- إذا كان لدينا مصدري ضوء (R1,G1,B1) و (R2,G2,B2) فالضوء الناتج (R1+R2, G1+G2, B1+B2).

خطوات إضافة إضاءة إلى المشهد

1. تعريف أشعة الناظم لكل نقطة من العناصر.
2. إنشاء وتعيين موقع مصدر ضوء أو أكثر.
3. إنشاء نموذج الإضاءة الذي يستخدم لتحديد مستوى الإضاءة ambient العامة والمنطقة المتأثرة بها من المسقط.
4. تعريف خصائص مواد العناصر في المشهد. ولنتناول هذه الخطوات بالتفصيل:

1. تعريف أشعة الناظم لكل نقطة من العناصر

- تستخدم لتحديد اتجاه العنصر بالنسبة لمصدر الإضاءة. وهذا يحدد كمية الإضاءة الواصلة لكل نقطة من كل مصدر ضوء.

2. إنشاء وتعيين موقع مصدر ضوء أو أكثر

- تمتلك مصادر الإضاءة عدة خصائص مثل اللون والموقع والاتجاه.
- الأمر المستخدم لتحديد جميع خصائص الإضاءة:

```
void glLight{if}[v](GLenum light, GLenum pname, TYPEparam);
```

- Light: يمكن أن تكون GL_LIGHT0, GL_LIGHT1, ... , or GL_LIGHT7 وتحدد الضوء المنشئ.
- Pname: تحدد خصائص الضوء كما في الجدول التالي. Param يحدد قيمة لكل خاصية في Pname. ويمكن أن يكون مجموعة قيم.

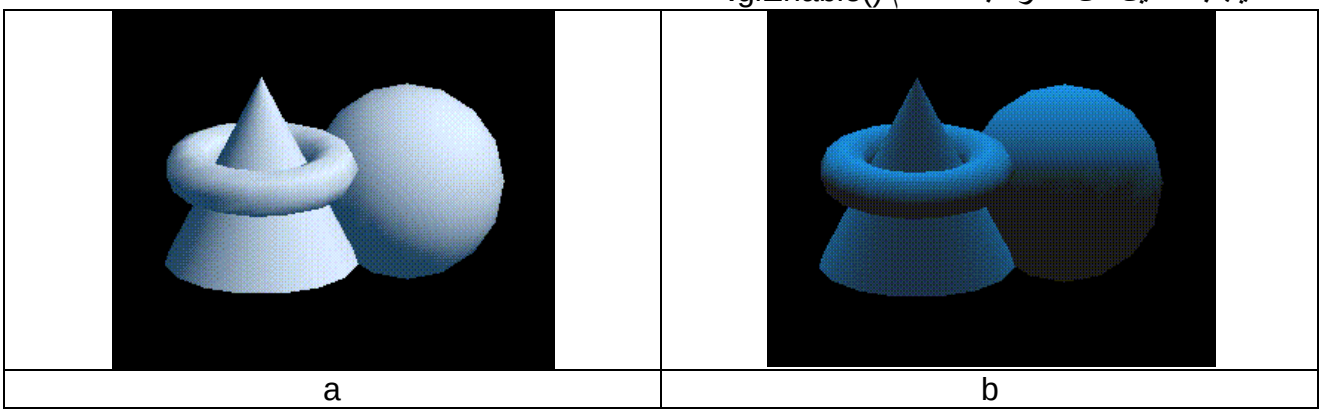
Parameter Name	Default Value	Meaning
GL_AMBIENT	(0.0, 0.0, 0.0, 1.0)	ambient RGBA intensity of light
GL_DIFFUSE	(1.0, 1.0, 1.0, 1.0)	diffuse RGBA intensity of light
GL_SPECULAR	(1.0, 1.0, 1.0, 1.0)	specular RGBA intensity of light
GL_POSITION	(0.0, 0.0, 1.0, 0.0)	(x, y, z, w) position of light

أمثلة:

```
GLfloat light_ambient[] = { 0.0, 0.0, 0.0, 1.0 };
GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };

glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```

- يجب تفعيل كل ضوء باستخدام glEnable().



a b

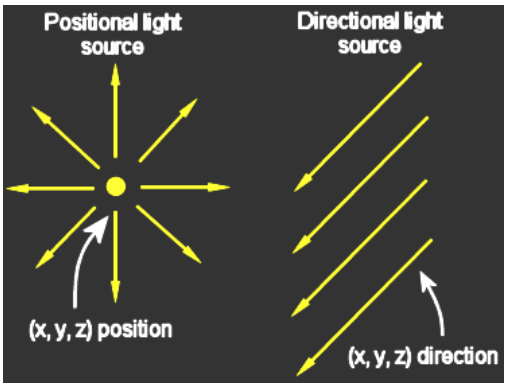
(a) This scene has pale blue ambient light and a white diffuse light source.
(b) This scene has a pale blue diffuse light source and almost no ambient light.

- هناك نوعان لتحديد مكان توضع الضوء بالنسبة للعناصر: النوع الأول directional(w=0) وله تأثير حزمة أشعة تصل أشعتها بشكل متتالي عبر الزمن مثل الشمس.

- النوع الثاني positional(w<>0) تصدر الحزم الضوئية من مكان معين ويحدد لها مكان تأثير على عناصر المشهد كما في المصباح المكتبي.

مثال:

```
GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```



التحكم بموقع واتجاه مصادر الضوء

- يعامل الضوء لتحديد موقعه واتجاهه تماماً كما يعامل أي عنصر هندسي. وبالتالي يتأثر الضوء بمصفوفة ModeView (Projection لا يؤثر في اللإضاءة).

أمثلة:

1. يبقى موقع الضوء ثابتاً

```
glViewport(0, 0, w, h);
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
if (w <= h)
   glOrtho (-1.5, 1.5, -1.5*h/w, 1.5*h/w, -10.0, 10.0);
else
   glOrtho (-1.5*w/h, 1.5*w/h, -1.5, 1.5, -10.0, 10.0);
glMatrixMode (GL_MODELVIEW);
glLoadIdentity();
GLfloat light_position[] = { 1.0, 1.0, 1.0, 1.0 };
glLightfv(GL_LIGHT0, GL_POSITION, position);
```

2. يتحرك الضوء حول عنصر معين

```
   GLfloat light_position[] = { 0.0, 0.0, 1.5, 1.0 };
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glPushMatrix();
      glTranslatef(0.0, 0.0, -5.0);
      glPushMatrix();
         glRotated((GLdouble) spin, 1.0, 0.0, 0.0);
         glLightfv(GL_LIGHT0, GL_POSITION, light_position);
      glPopMatrix();
      auxSolidTorus();
   glPopMatrix();
}
```

3. تحديد نموذج إضاءة

- لتحديد نموذج إضاءة لدينا الأمر glLightModel*() مع الخيارات:

أ- الضوء المحيط العام Global Ambient Light:

يشارك كل مصدر ضوء بضوء ambient إلى المشهد. هناك أيضاً ضوء ambient آخر لا ينبع من أي مصدر ضوء (ضوء ambient عام)

مثال:

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

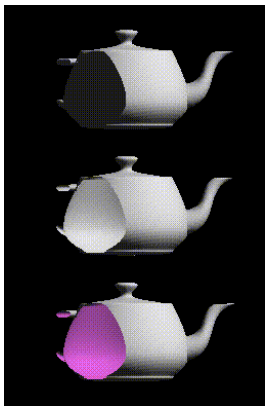
Each of the three teapots is drawn with increasing ambient light.

ب - إضاءة الوجهين

- حسابات الإضاءة تنفذ لجميع المضلعات سواء كانت وجه أمامي أو خلفي. يفضل عدم إضاءة الوجه غير المرئي وأيضاً قد يضاء بشكل مختلف.

مثال

```
glLightModeli(LIGHT_MODEL_TWO_SIDE, GL_TRUE);
```



The teapots are clipped to expose their interiors. The top teapot uses one-sided lighting, the middle one uses two-sided lighting with the same material for both front and back faces, and the bottom teapot uses two-sided lighting and different materials for the front and back faces

4. تعريف خصائص مواد العناصر في المشهد

- سنعرف هنا خصائص المواد: ألوان ambient وdiffuse وspecular واللمعان واللون الناتج عن أي ضوء منبعث.
- الأمر المستخدم لتحديد خصائص مواد العنصر هو:

void glMaterial{if}[v](GLenum face, GLenum pname, TYPEparam);

- face: يدل على الوجه من العنصر الذي سنطبق المادة ،وقيمه GL_FRONT أو GL_BACK أو GL_FRONT_AND_BACK.
- Pname: تعرف خاصية مادة محددة وتعين القيم لها بواسطة param .

Parameter Name	Default Value	Meaning
GL_AMBIENT	(0.2, 0.2, 0.2, 1.0)	ambient color of material
GL_DIFFUSE	(0.8, 0.8, 0.8, 1.0)	diffuse color of material
GL_AMBIENT_AND_DIFFUSE		ambient and diffuse color of material
GL_SPECULAR	(0.0, 0.0, 0.0, 1.0)	specular color of material
GL_SHININESS	0.0	specular exponent
GL_EMISSION	(0.0, 0.0, 0.0, 1.0)	emissive color of material
GL_COLOR_INDEXES	(0,1,1)	ambient, diffuse, and specular color indices

أمثلة:

.1

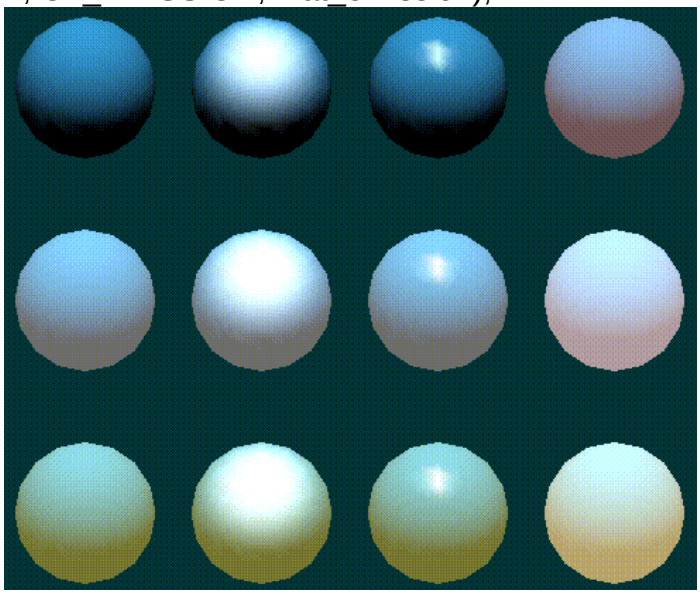
```
GLfloat mat_amb_diff[] = { 0.1, 0.5, 0.8, 1.0 };
glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE,  mat_amb_diff);
```

.2

```
GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat low_shininess[] = { 5.0 };
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, low_shininess);
```

.3

```
GLfloat mat_emission[] = {0.3, 0.2, 0.2, 0.0};
glMaterialfv(GL_FRONT, GL_EMISSION, mat_emission);
```



Twelve spheres, each with different material parameters. The row properties are as follows:
row 1 - No ambient reflection;
row 2 - Grey ambient reflection;
row 3 - Blue ambient reflection.
The first column uses a blue diffuse material color with no specular properties.
The second column adds white specular reflection with a low shininess exponent.
The third column uses a high shininess exponent and thus has a more concentrated highlight.
The fourth column uses the blue diffuse color and, instead of specular reflection, adds an emissive component.

القسم العملي

تطبيق1: رسم كرة مضاءة

```
#include <GL/glut.h>
#include <stdlib.h>
#include<math.h>
static void redraw(void)
{
   GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
   GLfloat mat_shininess[] = { 50.0 };
   GLfloat light_position[] = { -20.0, 0.0, 0.0, 0.0 };
```

```
    glClearColor(1.0,1.0,1.0,0.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glTranslatef(0.0f,0.0f,-100.0);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glutSolidSphere(20.0,15,15);

     glutSwapBuffers();
}

 int main(int argc, char**argv)
{
     glutInit(&argc,argv);
     glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
     glutInitWindowPosition(100,100);
     glutInitWindowSize(400,400);
     glutCreateWindow("lighting");
     glutDisplayFunc(redraw);
     glMatrixMode(GL_PROJECTION);
     gluPerspective(45,1.0,10.0,200.0);
     glMatrixMode(GL_MODELVIEW);
     glutMainLoop();
     return 0;
}
```

تطبيق2: تحريك ضوء باستخدام تحويلات Modeling

سنعمل على رسم طارة Torus وتحريك وتدوير مصدر ضوء حولها

```
#include <GL/glut.h>
#include <stdlib.h>
#include<math.h>
static int spin = 0;
static void redraw(void)
{
    GLfloat position[] = { 0.0, 0.0, 1.5, 1.0 };
     glClearColor(1.0,1.0,1.0,0.0);
     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
     glPushMatrix();

     glTranslatef(0.0f,0.0f,-50.0);
     glPushMatrix();
```

```
      glRotated ((GLdouble) spin, 0.0, 1.0, 0.0);
      glTranslated (0.0, 0.0, 15.0);
    glDisable (GL_LIGHTING);
    glColor3f (0.0, 1.0, 1.0);
    glutWireCube (1.0);
    glLightfv (GL_LIGHT0, GL_POSITION, position);
    glEnable (GL_LIGHTING);

    glPopMatrix();
    glutSolidTorus (2.75, 8.5,20,20);
    glPopMatrix();
    spin = (spin + 1) % 360;

     glutPostRedisplay();
     glutSwapBuffers();

}

 int main(int argc, char**argv)
{
    glutInit(&argc,argv);
     glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
     glutInitWindowPosition(100,100);
     glutInitWindowSize(400,400);
     glutCreateWindow("lighting");
     glutDisplayFunc(redraw);
     glMatrixMode(GL_PROJECTION);
     gluPerspective(45,1.0,10.0,200.0);
     glMatrixMode(GL_MODELVIEW);
     glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glEnable(GL_COLOR_MATERIAL);
     glutMainLoop();
     return 0;
}
```

تمرين: يطلب رسم تجمعات هندسية من العناصر الهندسية المدروسة:

- طاولة.
- كرسي.
- شاشة حاسوب.

تطبيق3: رسم مكعب مضاء بعدة مصادر إضاءة

سنعمل على رسم مكعب وتطبيق إضاءة عليه بواسطة لوحة المفاتيح ، فعندما نضغط المفتاح L تطبق الإضاءة ، وعندما نضغط المفتاح F يلغى تطبيق الإضاءة

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

GLfloat LightAmbient[]=     { 0.0f, 0.5f, 0.5f, 1.0f };
GLfloat LightDiffuse[]=     { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat LightPosition[]=    { 0.0f, 0.0f, 20.0f, 1.0f };
static void redraw(void);
int main(int argc, char **argv);
void keyboard ( unsigned char key, int x, int y );//لتعريف مفاتيح لوحة المفاتيح

int main(int argc, char **argv)
{
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
        glutInitWindowPosition(100,100);
        glutInitWindowSize(400,400);
        glutCreateWindow("draw Lighting rectangle");
        glutDisplayFunc(redraw);
        glMatrixMode(GL_PROJECTION);
        gluPerspective(45,1.0,10.0,200.0);
        glMatrixMode(GL_MODELVIEW);
        glLightfv(GL_LIGHT1, GL_AMBIENT, LightAmbient);          // Ambient إعداد الضوء
        glLightfv(GL_LIGHT1, GL_DIFFUSE, LightDiffuse);          //  Diffuse إعداد الضوء
        glLightfv(GL_LIGHT1, GL_POSITION,LightPosition);         // موقع الضوء
        glEnable(GL_LIGHT1);                                     // LIGHT1 تفعيل الضوء
        glEnable(GL_COLOR_MATERIAL);
        glutMainLoop();
        return 0;
}

static void redraw(void)
{
        glClearColor(1.0,1.0,1.0,0.0);
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();
        glTranslatef(0.0f,0.0f,-100.0);
        glRotatef(45,1.0f,0.0f,0.0f);
        glRotatef(45,0.0f,1.0f,0.0f);
        glBegin(GL_QUADS);
```

```
        // الوجه الأمامي
        glColor3f(1.0,0.0,0.0);
        glNormal3f( 0.0f, 0.0f, 1.0f);
        glVertex3f(-10.0f, -10.0f,  10.0f);
        glVertex3f( 10.0f, -10.0f,  10.0f);
        glVertex3f( 10.0f,  10.0f,  10.0f);
        glVertex3f(-10.0f,  10.0f,  10.0f);
        // الوجه الخلفي
        glColor3f(1.0,1.0,0.0);
        glNormal3f( 0.0f, 0.0f,-1.0f);
        glVertex3f(-10.0f, -10.0f, -10.0f);
        glVertex3f(-10.0f,  10.0f, -10.0f);
        glVertex3f( 10.0f,  10.0f, -10.0f);
        glVertex3f( 10.0f, -10.0f, -10.0f);
        // الوجه العلوي
        glColor3f(0.0,1.0,0.0);
        glNormal3f( 0.0f, 1.0f, 0.0f);
        glVertex3f(-10.0f,  10.0f, -10.0f);
        glVertex3f(-10.0f,  10.0f,  10.0f);
        glVertex3f( 10.0f,  10.0f,  10.0f);
        glVertex3f( 10.0f,  10.0f, -10.0f);
        // الوجه السفلي
        glColor3f(1.0,0.0,1.0);
        glNormal3f( 0.0f,-1.0f, 0.0f);
        glVertex3f(-10.0f, -10.0f, -10.0f);
        glVertex3f( 10.0f, -10.0f, -10.0f);
        glVertex3f( 10.0f, -10.0f,  10.0f);
        glVertex3f(-10.0f, -10.0f,  10.0f);
        // الوجه الأيمن
        glColor3f(0.0,0.0,1.0);
        glNormal3f( 1.0f, 0.0f, 0.0f);
        glVertex3f( 10.0f, -10.0f, -10.0f);
        glVertex3f( 10.0f,  10.0f, -10.0f);
        glVertex3f( 10.0f,  10.0f,  10.0f);
        glVertex3f( 10.0f, -10.0f,  10.0f);
        // الوجه الأيسر
        glColor3f(0.0,1.0,1.0);
        glNormal3f(-1.0f, 0.0f, 0.0f);
        glVertex3f(-10.0f, -10.0f, -10.0f);
        glVertex3f(-10.0f, -10.0f,  10.0f);
        glVertex3f(-10.0f,  10.0f,  10.0f);
        glVertex3f(-10.0f,  10.0f, -10.0f);
    glEnd();
    glutKeyboardFunc    ( keyboard );
```

```
        glutPostRedisplay();
        glutSwapBuffers();
}

void keyboard ( unsigned char key, int x, int y )
{
  switch ( key ) {
    case 27:  /*  Escape key  */
      exit ( 0 );
      break;
    case 'L':
      glEnable(GL_LIGHTING);
      break;
    case 'F':
      glDisable(GL_LIGHTING);
      break;
    default:
      break;
              }

}
```

****