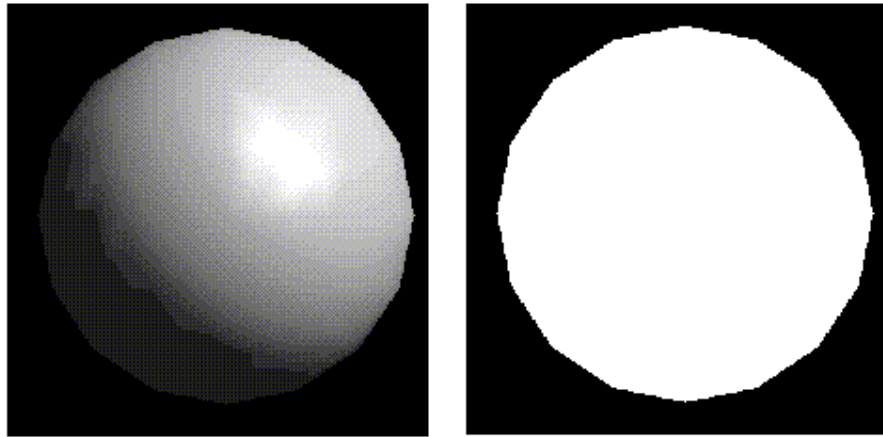


الفصل الرابع

الإضاءة

مقدمة:

تكتسب الصور جمالها عند تلوينها بألوان جميلة وزاهية ولكن التلوين وحده لا يكفي لإعطاء الصور مظهراً واقعياً فمثلاً لو تخيلنا أماناً صورة شاطئ البحر وقمنا بتلوين البحر باللون الأزرق فإن الصورة تبقى ناقصة ، لأن اللون الأزرق لوحده لا يكفي لإعطاء البحر لونه الحقيقي فالبحر يبدو باللون الأزرق البراق في يوم صيفي مشمس حيث تنعكس عليه أشعة الشمس وكذلك يظهر باللون الأزرق الكامد في يوم بارد غائم و يختلف لونه بحسب موقع الشمس في السماء ففي الصباح أزرق فاتح براق ووقت الغياب يأخذ لون برتقالي كما يختلف بحسب الامتداد فعند الأفق يبدو بلون وقرب الشاطئ يبدو بلون آخر ... وهكذا . إذاً موضوع تلوين الصورة لا ينفصل عن موضوع الإضاءة لما لها من تأثير كبير وواضح على مظهر الأجسام في الصورة . كما أن معظم الأشكال لا تبدو ثلاثية البعد ، حتى يتم تسليط الضوء عليها انظر الشكل .



وبما أن الـ OpenGL مسؤولة عن محاكاة الواقع وإظهار الصور بمظهرها الحقيقي إذاً يجب أن تكون قادرة على تحقيق مفهوم الإضاءة وقبل أن نبدأ بعرض ما تقدمه الـ OpenGL لتحقيق هذا الموضوع سنذكر المفاهيم الأساسية المرتبطة بالضوء والإضاءة وما يرتبط بهما من مكونات أساسية .

الضوء: شكل من أشكال الطاقة مركب من الفوتونات يُمكننا من رؤية الأجسام لأن العين حساسة له . يدخل الضوء إلى العين من البؤبؤ فتركزه العدسة في بؤرة على الشبكية حيث توجد نوعان من الخلايا العصبية الحساسة للضوء في شبكية العين وهما الخلايا العصبية والخلايا المخروطية ، حيث أن الخلايا العصبية تعمل في العتمة أو النور الخافت وهي عاجزة عن تمييز الألوان لذلك تنحصر رؤيتنا في الأماكن المعتمة على إحصار الأسود والأبيض فقط ، والخلايا المخروطية هي التي تمكننا من إدراك الألوان وهناك ثلاث مجموعات من الخلايا المخروطية في شبكية العين إحداها حساسة للضوء الأحمر والآخر الأزرق والثالثة للضوء الأخضر ومن المعروف أنه يمكن تركيب أي لون بمزج هذه الألوان الأولية فعند سقوط الضوء على الشبكية تتأثر به مجموعة أو أكثر من الخلايا المخروطية تبعاً للونه .

يعمل جهاز العرض البياني في الحاسوب بطريقة مماثلة إلى حد كبير لإحصار الألوان حيث تتألف شاشة جهاز العرض من شبكة مكونة من نقاط صغيرة جداً وتدعى النقطة الواحدة بالبكسل . لجعل الألوان مرئية نضيء كل نقطة من نقاط الشبكة (البكسلات) بضوء مركب من الألوان الأساسية (الأحمر ، الأخضر ، الأزرق) . وإضاءة البكسلات بنسب مختلفة من الأحمر والأخضر والأزرق تعطي كل بكسل لون مختلف وبالتالي نحصل على صورة ملونة .

يسمى النظام اللوني في جهاز العرض البياني الباعث للأشعة الضوئية الممثلة بالألوان الأحمر والأخضر والأزرق بالنظام اللوني RGB وأحياناً تضاف إلى هذا النظام قيمة رابعة هي A والتي تدعى ألفا . وبالتالي لتخزين لون بكسل يكفي أن نخزن نسب كل من R , G , B أو يمكن تخزين الألوان باستخدام ما يسمى بجدول الألوان color-indexing وهو جدول مسبق التعريف حيث كل لون له نسب خلط لـ R مع G مع B مثبتة مسبقاً ومخزنة في هذا الجدول ، فعند الحاجة لترميز لون في صورة ما نقوم بتخزين مؤشر على اللون الأقرب له في جدول الألوان المسبق التعريف .

الإضاءة: عندما ننظر إلى سطح فيزيائي فإن العين تستطيع أن تدرك لون هذا السطح بسبب انتشار الفوتونات وتأثيرها على شبكية العين ، وهذه الفوتونات ناتجة عن منبع ضوئي واحد أو أكثر . بعض هذه الفوتونات يمتصه الجسم وبعضه الآخر يعكسه ، وتختلف نسب الامتصاص وعكس الفوتونات من سطح لآخر باختلاف الخصائص الفيزيائية لكل من هذه السطوح ، فبعض هذه السطوح مشعة فتعكس الضوء بشكل كبير في اتجاه واحد في حين تعكس سطوح أخرى الضوء في كل الاتجاهات وبالتالي تقل شدة الإضاءة.

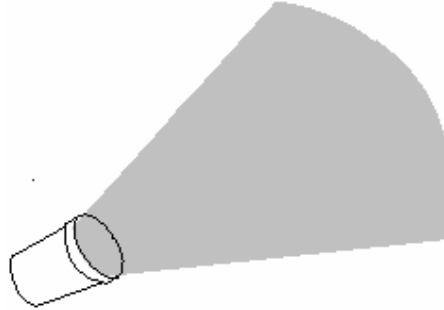
لدراسة الإضاءة يجب التركيز على المفهومين التاليين :

١. المصدر الضوئي .
٢. الجسم المضاء .

المصدر الضوئي:

هناك ثلاث أنواع لمصادر الإضاءة :

١. مصدر الضوء الاتجاهي Directional source: حيث يكون الضوء قادم من مصدر موجود في مكان بعيد جداً يمكن اعتباره في اللانهاية ولهذا تعتبر الأشعة الضوئية الصادرة عنه متوازية ونأخذ كمثال على هذا النوع الشمس .
٢. مصدر الضوء الموضعي Positional light source: حيث يكون الضوء قادم من مصدر موجود في موقع محدد و الأشعة الضوئية الصادرة عنه تكون موجه باتجاه واحد ونأخذ كمثال على هذا النوع المصباح المكتبي .
٣. مصدر الضوء الموضعي المخروطي Spotlight: حيث يكون الضوء قادم من مصدر ضوئي موجود في موقع محدد والأشعة الضوئية الصادرة عنه تكون موجه بشكل مخروطي ونأخذ كمثال مصباح الجيب.



الأجسام المضاءة:

تتحدد ألوان الأجسام بنسبة الضوء الذي تمتصه وتعكسه هذه الأجسام ، مع العلم أن تأثير المصادر الضوئية لا يظهر إذا لم تكن سطوح الأجسام تمتص وتعكس الضوء. وكذلك تختلف الاتجاهات التي ينعكس فيه الضوء بحسب الطبيعة الفيزيائية لهذه السطوح، وهذا الاختلاف بالاتجاهات ينتج عنه أنواع مختلفة من الإضاءة سنذكرها لاحقاً.

الإضاءة في الـ OpenGL:

تستطيع الـ OpenGL معالجة موضوع الإضاءة على اعتبار أن الضوء مركب من الألوان الأساسية وهي الأحمر والأخضر والأزرق ، وبالتالي تتحدد خصائص أي مصدر ضوئي بكمية الألوان الأحمر والأخضر والأزرق التي يرسلها هذا المصدر . وكذلك يعتمد لون الأجسام على اللون أو الألوان التي يعكسها ذلك الجسم . وعند تعرض الأجسام للضوء تمتص بعضه وتعكس الباقي . فإذا سقط الضوء الأبيض على جسم وامتنص هذا الجسم للونين الأخضر والأزرق وعكس الضوء الأحمر فإنه جسم أحمر اللون . وكل الأجسام الملونة تمتص كل ألوان الضوء ما عدا لونها فتعكسه، باختلاف نسب الامتصاص وعكس الألوان الأساسية للضوء يختلف لون الجسم . وإذا أضأنا جسم لونه أخضر بضوء أحمر صافي أو أزرق صافي فإن هذا الجسم سيبدو أسود اللون لأن كل من الضوئين الأزرق والأحمر لا يحويان على مركبة اللون الأخضر الذي من المفروض أن يعكسه الجسم الأخضر ليظهر لونه الحقيقي انظر الشكل:



لاحظ كيف ظهرت أوراق الورد بلون أسود عند إضاءتها باللون الأحمر ، وكيف امتصت الورد الحمراء لونها .

إن خصائص المادة مكونة من المركبات التالية : Specular، Diffuse، Ambient.

تعرف المركبتين ambient، Diffuse لون المادة، أما المركبة Specular فيعبر عن شدة اللون أو لمعانه. وتعتبر الإضاءة في الـ OpenGL مركبة من أربع مكونات أساسية هي :

Specular، Diffuse، Ambient and وهي أنواع الإضاءة بالإضافة إلى المكون

Emitted : حيث تنشأ الإضاءة منه ولا يتأثر بالمصادر الضوئية الأخرى .

أنواع الإضاءة:

١. الإضاءة المحيطية Ambient : حيث يكون الضوء قادم من جميع الاتجاهات وعند اصطدامه بسطح الجسم ينعكس في جميع الاتجاهات ، ولنأخذ كمثال على هذا النوع الإضاءة في الغرفة حيث يصطدم الضوء الصادر عن النيون بجدران الغرفة فيعكس كل جدار الضوء في جميع الاتجاهات فيصبح الجدار وكأنه مصدر للضوء وعندها يصبح من الصعب تحديد المصدر الأساسي للضوء فتدعى هذه الإضاءة بالإضاءة المحيطية Ambient .

٢. الإضاءة المنتشرة Diffuse : حيث يكون الضوء قادم من اتجاه واحد وعند اصطدامه بسطح الجسم ينعكس في جميع الاتجاهات بشكل متساوي لذلك فإن سطح الجسم يبدو متساوي اللامعان في جميع نقاط سطحه ويبدو بلون باهت ولا تتأثر الإضاءة بتغير موقع العين النازرة لهذا الجسم . تدعى هذه الإضاءة بالإضاءة المنتشرة Diffuse .

٣. الإضاءة البراقة : حيث يكون الضوء قادم من اتجاه واحد وعند اصطدامه بسطح الجسم ينعكس باتجاه واحد ومحدد حيث تظهر البقعة من السطح الذي اصطدم بها الضوء وانعكس منها البقعة الأكثر لمعاناً من كل النقاط الأخرى. مثال شعاع الليزر وتدعى هذه الإضاءة بـ specular light .

خطوات إضافة الإضاءة إلى المشهد في الـ OpenGL :

١. تعريف الأشعة النازمة لكل عقدة من عقد أغراض المشهد.
٢. إنشاء واختيار المنابع الضوئية و توضعها في المشهد.
٣. إنشاء واختيار نموذج الإضاءة، الذي يعرف مستوى الضوء المحيط العام Global Ambient Light، وتأثير موضع نقطة النظر (فيما إذا كانت في اللانهاية أو قريبة من المشهد)، مم له تأثير على الحسابات التي تقوم بها OpenGL لتحديد تصرف الضوء في المشهد.
٤. تعريف خواص المواد للأغراض في المشهد.

مثال ١ :

رسم حلقة مضبوطة

```
#include <GL/gl.h>
#include <GL/glu.h>
#include "aux.h"

void myinit(void)
{
    GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat mat_shininess[] = { 50.0 };
    GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };

    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);
    glDepthFunc(GL_LEQUAL);
    glEnable(GL_DEPTH_TEST);
}

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glutSolidSphere (1.0, 20, 16);
    glFlush();
}
```

```
void myReshape(GLsizei w, GLsizei h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h)
        glOrtho (-1.5, 1.5, -1.5*(GLfloat)h/(GLfloat)w,
                1.5*(GLfloat)h/(GLfloat)w, -10.0, 10.0);
    else
        glOrtho (-1.5*(GLfloat)w/(GLfloat)h,
                1.5*(GLfloat)w/(GLfloat)h, -10.0, 10.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH );
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (100, 100);
    glutCreateWindow (argv[0]);
    myinit();
    glutDisplayFunc(display);
    glutReshapeFunc(myReshape);
    glutMainLoop();
}
```

تعريف الأشعة النازمة NormalVector: هي أشعة تشير إلى اتجاه عمودي على سطح ما، فإذا كان السطح مستوياً كان الشعاع وحيداً من أجل كل نقط المستوى، أما إذا كان منحنيّاً فالناظم يختلف في كل نقطة. ولذلك يمكن في OpenGL القيام بإسناد ناظم إلى كل عقدة مكونة للشكل. يعبر الناظم عن توجيه السطح في الفراغ بالنسبة إلى المنبع الضوئي، لذلك فهو يحدد كم من الضوء سيصل إلى السطح في عقده.

مثال ٢:

تعريف الناظم

```
glBegin(GL_POLYGON);
glNormal3fv(n0);
glVertex3fv(v0);
glNormal3fv(n1);
glVertex3fv(v1);
glNormal3fv(n2);
glVertex3fv(v2);
glNormal3fv(n3);
glVertex3fv(v3);
glEnd();
```

إنشاء المصدر الضوئي:

لإنشاء المصدر الضوئي يجب تحديد خصائصه والتي تتمثل باللون الذي يصدره ، المكان الموجود فيه ، الاتجاه الذي تنطلق نحوه أشعة الضوء . وكل هذا يتحدد بتعليمة واحدة تقدمها الـ OpenGL وهي :

Void *glLight{if}[v](GLenum *light*, GLenum *pname*, TYPEparam);*

يتحدد الضوء الذي يصدر عن المنبع الضوئي بالبارمتر *light* الذي يأخذ ثمانية قيم مختلفة ينتج عنها ثمانية أضواء مختلفة وهي :

GL_LIGHT0, GL_LIGHT1, ... , or GL_LIGHT7

وتحدد خصائص هذا الضوء بحسب البارمتر *pname* أما قيمة هذه الخاصية فتتحدد بالبارمتر *TYPEparam*، وسنعرض جميع الخصائص التي تتحدد بالبارمتر *pname* ضمن الجدول التالي :

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
GL_SPOT_DIRECTION	(0.0, 0.0, -1.0)	(x, y, z) direction of spotlight
GL_SPOT_EXPONENT	0.0	spotlight exponent
GL_SPOT_CUTOFF	180.0	spotlight cutoff angle
GL_CONSTANT_ATTENUATION	1.0	constant attenuation factor
GL_LINEAR_ATTENUATION	0.0	linear attenuation factor
GL_QUADRATIC_ATTENUATION	0.0	quadratic attenuation factor

إن القيمة الافتراضية لكل من GL_DIFFUSE and GL_SPECULAR في هذا الجدول تخص الضوء GL_LIGHT0 ، أما من أجل باقي الألوان فإن القيمة الافتراضية لهما تساوي (0.0, 0.0, 0.0, 1.0) .

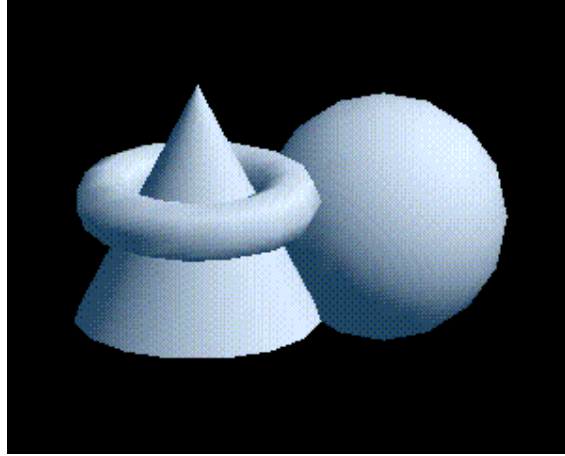
مثال ٣:

تحديد الضوء ونوع الإضاءة الصادرة عن المنبع الضوئي

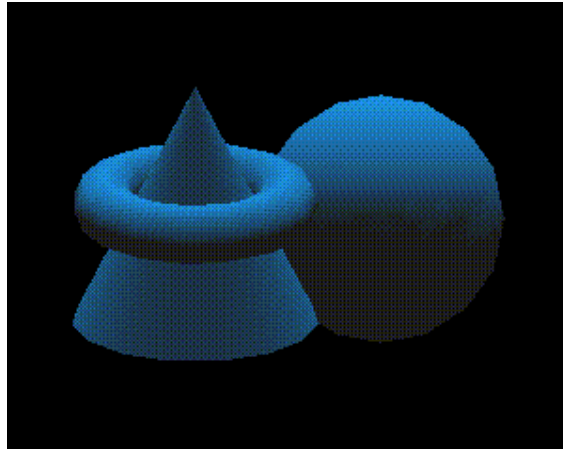
لإعطاء المشهد إضاءة محيطية زرقاء :

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

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



في هذا المشهد لدينا إضاءة محيطية زرقاء باهتة مع مصدر ضوئي يعطي إضاءة بيضاء منتشرة في كل الاتجاهات . إن البارمتر `GL_DIFFUSE` يجعل الضوء المختار قريب جداً من لون الضوء الحقيقي ويأخذ هذا البارمتر القيمة $(1.0, 1.0, 1.0, 1.0)$ من أجل الضوء `GL_LIGHT0` والقيمة $(0.0, 0.0, 0.0, 0.0)$ من أجل باقي الأضواء .



في هذا المشهد لدينا إضاءة زرقاء منتشرة (diffuse light source) مع عدم وجود أي تأثير لإضاءة محيطية .

أما بالنسبة للبارمتر GL_SPECULAR فإنه يظهر بقعة مشعة ولامعة في مكان سقوط الضوء على سطح الجسم، وغالباً ما تكون هذه البقعة المشعة بلون أبيض. ولإعطاء تأثير حقيقي يجب جعل قيمة المتحول GL_SPECULAR ، GL_DIFFUSE دائماً متماثلة، حيث يأخذ البارمتر GL_SPECULAR القيمة (1.0, 1.0, 1.0, 1.0) من أجل الضوء GL_LIGHT0 ويأخذ أيضاً القيمة (0.0, 0.0, 0.0) من أجل باقي الأضواء .

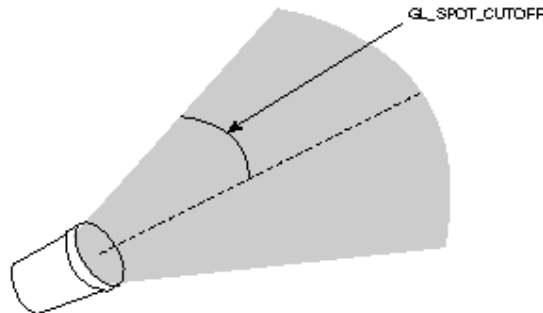
موقع المصدر الضوئي :

- يتحدد موقع المصدر الضوئي بالتعليمة التالية :

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

حيث تمثل عناصر المصفوفة قيم الإحداثيات x, y, z أما البارمتر الرابع يسمى w والذي يحدد نوع المصدر الضوئي الذي تصدر عنه الإضاءة :

1. مصدر الضوء الموضعي W=1 Positional : يعني أن المصدر الضوئي موقعه معروف ومحدد بالقيم x, y, z مثل الضوء الصادر عن المصباح المكتبي .
2. مصدر الضوء الاتجاهي W=0 Directional : يعني أن المصدر الضوئي يقع في اللانهاية وتمثل القيم x, y, z عكس اتجاه الأشعة الضوئية الصادرة عن المنبع مثل الضوء الصادر عن الشمس ولالتي تتجه نحو المحور z السالب.
3. مصدر ضوئي على شكل بقعة ضوئية : وهو يعتبر أحد أشكال الإضاءة الموضعية ، حيث يتم إرسال الضوء على شكل مخروطي ، ولإنشاء مصدر ضوئي يعطي بقعة ضوئية يجب تحديد اتساع الشعاع المخروطي الصادر ، ويتم ذلك بتحديد الزاوية بين محور المخروط والحافة لهذا المخروط (تذكر أن هذا المصدر الضوئي ما هو إلا حالة خاصة من المصدر الضوئي الموضعي فبالإضافة إلى تحديد موقعه أيضاً) . مع العلم أنه ليس هناك أي مانع من إنشاء مصدر ضوئي كهذا من المصدر الضوئي الاتجاهي ولكن لن يكون هناك أي فائدة من ذلك ولن تحصل على الإضاءة المطلوبة . ولتحديد الزاوية في الـ OpenGL نستخدم البارمتر GL_SPOT_CUTOFF ، بحيث تكون الزاوية الكلية للمخروط تساوي $2 * GL_SPOT_CUTOFF$.



لا حظ أنه لا توجد أي إضاءة خارج حدود المخروط . وبشكل افتراضي فإن المصدر الضوئي المخروطي يكون غير فعّال حيث يكون البارمتر GL_SPOT_CUTOFF يساوي ١٨٠ درجة وهذا يعني أن الضوء يرسل في كل الاتجاهات (الزاوية في قمة المخروط تساوي ٣٦٠ درجة) . ولتحكم بقيمة الزاوية في الـ OpenGL نستخدم التعليمة التالية : glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0); . ولتحديد الاتجاه الذي ترسل فيه أشعة المخروط الضوئي نكتب :

```
GLfloat spot_direction[] = { -1.0, -1.0, 0.0 };
glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spot_direction);
```

شدة الإضاءة و تخامدها:

تختلف شدة الإضاءة بحسب قرب المصدر الضوئي وبعده فكلما كان المصدر الضوئي قريب تكون شدة الإضاءة قوية و كلما بعد المصدر الضوئي تضعف الإضاءة وتصبح خافتة ، ولكن هذا الأمر لا ينطبق على المصدر الضوئي الذي يقع في اللانهاية (كالشمس) ، حيث أن بعد الشمس ووجودها في اللانهاية لا يعني أن شدة الإضاءة ضعيفة وقليلة ، إذاً موضوع اختلاف شدة الإضاءة بحسب الموقع ينطبق على الإضاءة الموضعية فقط ، وبحسب التخامد باستخدام المعامل التالي:

$$\text{attenuation factor} = \frac{1}{k_c + k_l d + k_q d^2}$$

حيث :

D = المسافة بين موضع الضوء النقطي والعقدة
Kc = GL_CONSTANT_ATTENUATION ثابت
Kl = GL_LINEAR_ATTENUATION خطي
Kq = GL_QUADRATIC_ATTENUATION مربع

افتراضياً تكون قيم هذه الثوابت كما يلي:

Kl and Kq = 0.0 , Kc= 1.0 .

استخدام عدة مصادر ضوئية في نفس الوقت :

سبق وذكرنا أن الـ OpenGL توفر لنا على الأقل ثمانية أضواء مختلفة . و بما أن الـ OpenGL قادرة على إجراء عمليات حسابية لمعرفة الضوء الساقط على كل نقطة من نقاط الشكل حيث أن هذا الضوء هو خليط من الأضواء الصادرة عن المصادر الضوئية المختلفة ، فإن تأثيرات الضوء بشكل عام تكون متزايدة . لتحديد الأضواء المختلفة يوجد ثمانية ثوابت يمكنك أن تستخدمها وهي :

GL_LIGHT0, GL_LIGHT1, GL_LIGHT2, GL_LIGHT3 ... GL_LIGHT8 .

مع ملاحظة الاختلاف في الخصائص الافتراضية لكل ضوء كما ذكرنا سابقاً .

مثال ٤ :

إن الأسطر التالية تعطينا إضاءة على شكل مخروط ضوئي أبيض ضعيف الإنارة

```
GLfloat light1_ambient[] = { 0.2, 0.2, 0.2, 1.0 };
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, 45.0);
glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spot_direction);
glLightf(GL_LIGHT1, GL_SPOT_EXPONENT, 2.0);

glEnable(GL_LIGHT1);
```

ملاحظة ١ :

تذكر أنه لجعل المصدر الضوئي يقوم بعمله ويرسل إضاءة ما، يجب تفعيل كل مصدر من المصادر الضوئية باستخدام التعليمة **glEnable**.

التحكم بموقع واتجاه المصادر الضوئية:

يمكن التحكم بموقع واتجاه المصادر الضوئية تماماً كما تحكمنا بمواقع الأشكال الهندسية وذلك بإجراء عمليات الانسحاب والدوران المختلفة ، حيث يتأثر بمكس مصفوفات **ModelView** فقط . بعد أن نحدد الموقع والاتجاه الأساسيان لمصدر الضوئي باستخدام التعليمة **glLight*()** يمكن أن نغير هذا الاتجاه حسب ما يتطلب المشهد باستخدام تعليمات الانسحاب والدوران التي كنا قد تعرفنا عليهم سابقاً .

لنفترض أننا أردنا تدوير أو انسحاب موضع منبع ضوئي، أحد الطرق هو إعداد الموضع بعد التحويل.

```
void display(GLint spin)
{
    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();
```

إذا أردت أن تحرك الضوء بحيث يبقى موضعه ثابتاً بالنسبة للمشاهد، (يتحرك مع تحرك نقطة النظر)، يجب تحديد موضع الضوء قبل التحويل كما في المثال التالي:

```
Void reshape (int w, int h)
{
    GLfloat light_position() = { 0.0, 0.0, 1.0, 1.0 };

    glViewport(0, 0, w-1, h-1);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(40.0, (GLfloat) w/(GLfloat) h, 1.0, 100.0);
    glMatrixMode(GL_MODELVIEW);

    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
}

void display(GLint spin)
{
    glClear(GL_COLOR_BUFFER_MASK | GL_DEPTH_BUFFER_MASK);
    glPushMatrix();
        glTranslatef (0.0, 0.0, -5.0);
        glRotatef ((GLfloat) spin, 1.0, 0.0, 0.0);
        auxSolidTorus();
    glPopMatrix();
    glFlush();
}
```

مثال ٥:

تحريك المصدر الضوئي

ليكن لدينا المثال التالي، لتدوير مصدر ضوئي مع مربع حول حلقة Tours:

كيف يمكن أن نثبت فيه المنبع الضوئي مع نقطة النظر؟

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

static int spin = 0;

/*Initialize material property, light source, lighting model.
 * and depth buffer.
 */
void init(void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glShadeModel (GL_SMOOTH);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);
}

/*Here is where the light position is reset after the modeling
 * transformation (glRotated) is called. This places the
 * light at a new position in world coordinates. The cube
 * represents the position of the light.
 */
void display(void)
{
    GLfloat position[] = { 0.0, 0.0, 1.5, 1.0 };

    glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPushMatrix() ;
    gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

    glPushMatrix() ;
    glRotated ((GLdouble) spin, 1.0, 0.0, 0.0);
    glLightfv (GL_LIGHT0, GL_POSITION, position);

    glTranslated (0.0, 0.0, 1.5);
    glDisable (GL_LIGHTING);
    glColor3f (0.0, 1.0, 1.0);
    glutWireCube (0.1);
    glEnable (GL_LIGHTING);
    glPopMatrix() ;

    glutSolidTorus (0.275, 0.85, 8, 15);
    glPopMatrix() ;
    glFlush() ;
}
```

```

void reshape (int w, int h)
{
    glViewport (0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(40.0, (GLfloat) w/(GLfloat) h, 1.0, 20.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void mouse(int button, int state, int x, int y)
{
    switch (button)
    {
        case GLUT_LEFT_BUTTON:
            if (state == GLUT_DOWN)
            {
                spin = (spin + 30) % 360;
                glutPostRedisplay();
            }
            break;
        default:
            break;
    }
}

void keyboard(unsigned char key, int x, int y)
{
    switch (key){
        case 27:
            exit(0);
            break;
    }
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (100, 100);
    glutCreateWindow (argv[0]);
    init ();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMouseFunc(mouse);
    glutKeyboardFunc(keyboard);
    glutMainLoop();
    return 0;
}

```

اختيار نموذج الإضاءة:

تحدد الـ OpenGL النموذج الضوئي بواسطة ثلاث مركبات أساسية ، هي :

١ . الإضاءة المحيطية العامة .

٢ . نقطة النظر .

٣ . الإضاءة للوجهين .

الإضاءة المحيطية العامة:

كل مصدر ضوئي يمكن أن يرسل إضاءة محيطية ، وعندما نقول أن المصدر الضوئي ذو إضاءة محيطية عامة فهذا يعني أن كمية الضوء المرسله تتساوى في كل نقطة من نقاط المشهد بالإضافة إلى ذلك توجد إضاءة محيطية لا يمكن معرفة مصدرها الأصلي . لتحديد شدة الإضاءة المحيطية العامة في المشهد نستخدم التعليمة التالية :

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

حيث أن بارمترات المصفوفة **lmodel_ambient[]** تمثل قيم الـ RGBA ، وهذه القيم تعتبر قيم افتراضية من أجل GL_LIGHT_MODEL_AMBIENT .

نقطة النظر:

إن موقع العين النازرة تؤثر في العمليات الحسابية المتعلقة بالإضاءة البراقة المنعكسة . وبصورة خاصة فإن لمعان أي عقدة من الشكل يتعلّق بالشعاع الناظم المرتبط بهذه العقدة ، الاتجاه بين العقدة و المصدر الضوئي،

والإتجاه بين العقدة وعين الناظر. وفي حال كانت العين الناظرة تقع في اللانهاية فإن الإتجاه بينها وبين أي عقدة يبقى ثابت. إن استخدام موقع معروف للعين الناظرة يعطي نتائج أقرب للواقع، ولكن بما إن اتجاه العين الناظرة سيتم حسابه من أجل كل عقدة من عقد الشكل فإن ذلك سيقال من فعالية الأداء. إن موقع العين الافتراضي يعتبر في اللانهاية ويمكن التحكم بذلك باستخدام التعليمة التالية :

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

ولإرجاع موقع العين لاللانهاية نمرر الثابت GL_FALSE .

الإضاءة للوجهين:

غالباً ما نأخذ بعين الاعتبار الإضاءة الأمامية للشكل فقط ونهمل أمر إضاءة الوجه الخلفي للأشكال التي لدينا على اعتبار أن الوجه الخلفي عادةً يكون غير ظاهر ، ولكن أحياناً قد نضطر لإنارة الجزء الخلفي وذلك عندما نريد إظهار الجزء الداخلي من الشكل الذي لدينا أو عندما نريد إظهار الفرق بين خصائص الوجه الخلفي (المادة التي يتكون منها) له و خصائص الوجه الأمامي عندها نستخدم إضاءة بجانبين :

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

وعندما تريد إيقاف هذه الإضاءة يمكن استخدام الثابت GL_FALSE .

خصائص الأجسام المضاءة :

رأينا سابقاً كيف استطعنا إنشاء مصادر ضوئية والتحكم بخصائصها ونماذج الإضاءة المستخدمة ، وسنتحدث في هذا القسم عن طريقة التحكم بخصائص الأجسام المضاءة مع العلم أن خصائص الأجسام التي نريد تحديدها تشبه إلى حد كبير تلك التي عرفناها من أجل المصادر الضوئية و سنذكر الألوان واللمعان والإضاءة المحيطة وانتشار الضوء ، كما أن آلية التحكم بها تشابه آلية التحكم بخصائص المصدر الضوئي مع فارق في التعليمة المستخدمة ، حيث نستخدم مع الأجسام التعليمة التالية : ***glMaterial** .

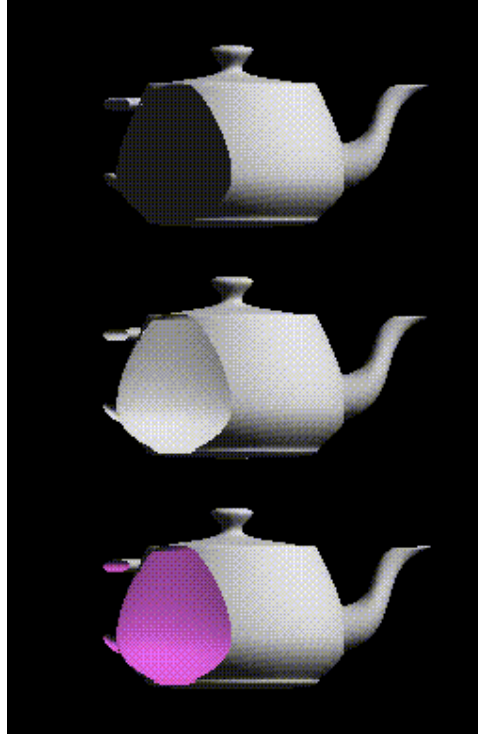
```
void glMaterial{if}[v](GLenum face, GLenum pname, TYPEparam);
```

وسنعرض جدول يوضح القيم التي يمكن أن يأخذها المتحول *pname* :

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

كما سبق وذكرنا فإنه يمكن التحكم بإضاءة الوجه الأمامي والخلفي للجسم ، فعندما يكون الوجه الخلفي للجسم في الواقع مرئي يمكن التحكم بخصائص الوجهين الأمامي والخلفي للجسم باستخدام البارمتر `face` في التعليمة `glMaterial*()` . كما يمكن إظهار الاختلاف في المادة التي يتألف منها الوجه الأمامي عن تلك في الوجه الخلفي بحيث يظهر ذلك جلياً باستخدام الإضاءة ذات الجانبين .



يظهر في الشكل أن إبريق الشاي الأول مضاء بإضاءة ذات جانب واحد ، أما الإبريق الذي في الوسط مضاء بإضاءة ذات جانبيين و خصائص الوجه الأمامي والخلفي متشابهة والإبريق الأخير مضاء أيضاً بإضاءة ذات جانبيين مع اختلاف في خصائص الوجه الأمامي والخلفي .

الإضاءة المحيطة وانتشار الضوء المنعكس عن سطوح الأجسام :

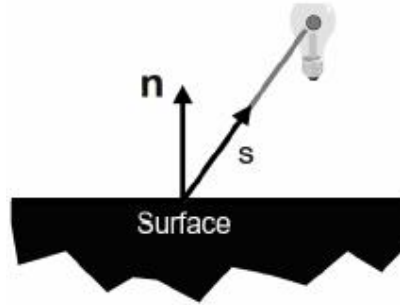
تحدثنا عن الإضاءة المحيطة وانتشار الضوء الصادر عن المنبع الضوئي ، أما الآن فسنحدث عنها من وجهة نظر الأجسام ، حيث تتصرف الأجسام تصرف المنابع الضوئية عندما تسقط عليها الإضاءة وتنعكس من سطوحها فتختلف الإضاءة المنعكسة من سطوح هذه الأجسام بحسب طبيعة المواد وخصائص هذه الأجسام ولتحديد هاتين الخاصيتين نستخدم الثابتين `GL_AMBIENT`، `GL_DIFFUSE` مع التعليمة `. glMaterial*( )`

الإضاءة المحيطية المنعكسة : Ambient Reflection

تنتشر الإضاءة المحيطية في كل الاتجاهات بكمية متساوية ، وتعتبر الإضاءة المحيطية المنعكسة مستقلة تماماً عن موقع المصدر الضوئي و مستقلة عن موقع عين الناظر .

الإضاءة المنتشرة المنعكسة :Diffuse Reflection

ينتشر الضوء المنعكس في كل الاتجاهات ، وتعتبر الإضاءة المنتشرة أهم عنصر في إظهار لون الجسم الحقيقي ، وتعتمد الإضاءة المنتشرة على موقع المصدر الضوئي في حين أنها مستقلة عن موقع العين الناظرة . وتتسبب الإضاءة المنتشرة بإعطاء الجسم سطوع أكثر .

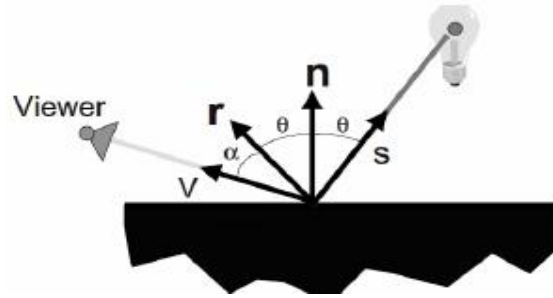


وفي العالم الحقيقي تعطي الإضاءة المحيطية والمنتشرة نفس اللون ولذلك فإن الـ OpenGL تسمح لك بإعطاء كل من هاتين الخاصيتين نفس القيمة باستخدام تعليمة واحدة :

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

الإضاءة المنعكسة المسببة للمعان : Specular Reflection

حيث تتعكس الإضاءة في اتجاه واحد فقط وتعتمد شدة هذه الإضاءة على موقع المصدر الضوئي ، وموقع العين الناظرة ، وتتسبب هذه الإضاءة بإظهار بقعة أكثر لمعاناً من كل باقي أجزاء الجسم المضاء .



ولملاحظة اختلاف اللامعان يمكنك وضع جسم معدني في ضوء الشمس والنظر إليه من اتجاهات مختلفة عندها ستلاحظ اختلاف مكان البقعة اللامعة من الجسم باختلاف موقع العين الناظرة .
يمكننا التحكم بلون البقعة المضيئة باستخدام الثابت GL_SPECULAR بالإضافة إلى القدرة على التحكم بحجم هذه البقعة باستخدام الثابت GL_SHININESS .

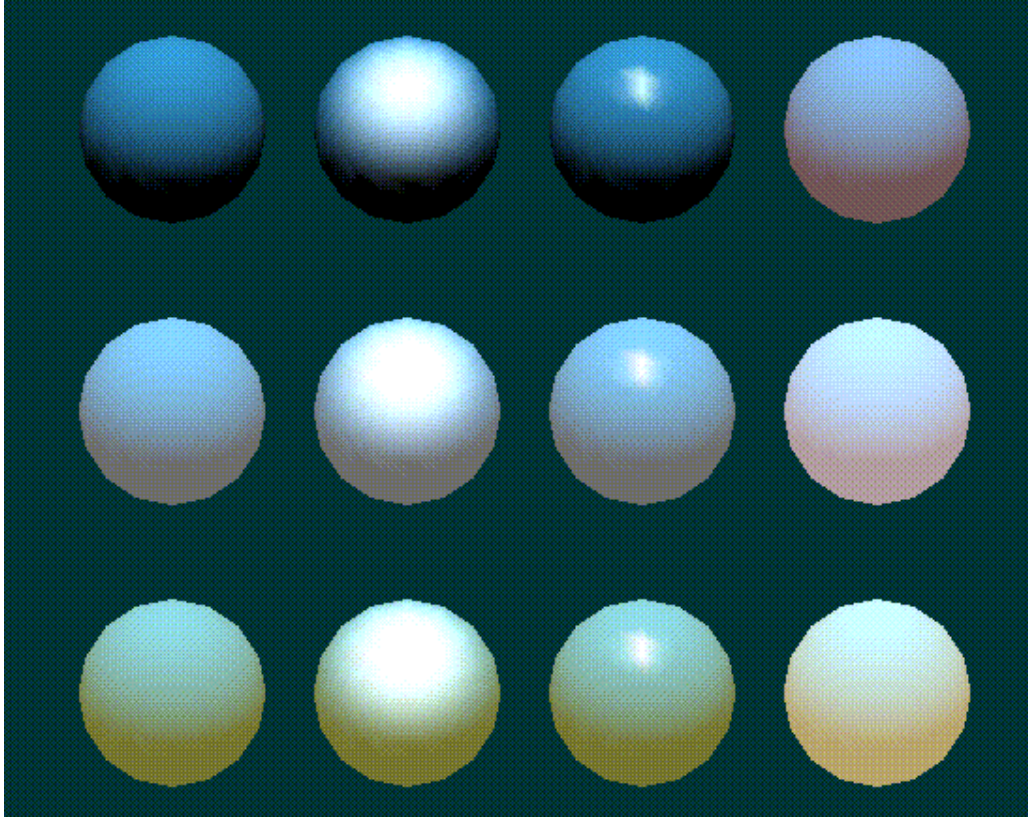
```
GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat low_shininess[] = { 5.0 };
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, low_shininess);
```

مثال عن تحديد خصائص الأجسام المضيئة :

```
GLfloat ambient[] = { 0.2, 0.2, 0.2, 1.0 };
GLfloat diffuse[] = { 1.0, 0.8, 0.0, 1.0 };
GLfloat specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat shine = 100.0;
glMaterialfv (GL_FRONT, GL_AMBIENT, ambient);
glMaterialfv (GL_FRONT, GL_DIFFUSE, diffuse);
glMaterialfv (GL_FRONT, GL_SPECULAR, specular);
glMaterialfv (GL_FRONT, GL_SHININESS, &shine);
```

تغيير خصائص الأجسام المضيئة :

قد يحتوي المشهد الواحد على عدة أجسام ذات خصائص مختلفة بحيث تظهر الإضاءة الساقطة على كل جسم انطباعاً ضوئياً ولونياً مختلفاً وسنوضح في المثال التالي طريقة تحديد خصائص الأجسام ، حيث سنرسم ١٢ كرة لكل منها خصائص .



لدينا في هذا الشكل ١٢ كرة، تختلف عن بعضها بالمادة:

- § السطر الأول : لا توجد في إضاءة محيطية منعكسة. السطر الثاني : إضاءة محيطية رمادية منعكسة.
- السطر الثالث : إضاءة محيطية زرقاء منعكسة.
- § العمود الأول يستخدم اللون الأزرق للإضاءة المنتشرة من المادة blue diffuse material وبدون أي خاصية specular . العمود الثاني: تم إضافة إضاءة بيضاء منعكسة white specular reflection مع معامل لمعان صغير، في العمود الثالث تم استخدام معامل لمعان أكبر من سابقه، و أخيراً في العمود الرابع استبدل المعامل specular بالمعامل Emission .

: Emission

يمكن جعل الغرض يبدو وكأنه يرسل ضوءاً باستخدام GL_EMISSION ويتم تحديد لون الإضاءة التي يرسلها الجسم بتعريف قيم الـ RGBA المتعلقة بالثابت GL_EMISSION .

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

ولكن ذلك لا يجعل منها مصدراً ضوئياً.

مثال ٦:

سنعرض النص البرمجي المتعلق بالاثني عشر كرة التي رأيناها في الشكل السابق

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

/*Initialize z-buffer, projection matrix, light source ,
 * and lighting model. Do not specify a material property here.
 */
void init(void)
{
    GLfloat ambient[] = { 0.0, 0.0, 0.0, 1.0 };
    GLfloat diffuse[] = { 1.0, 1.0, 1.0, 1.0};
    GLfloat specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat position[] = { 0.0, 3.0, 2.0, 0.0 };
    GLfloat lmodel_ambient[] = { 0.4, 0.4, 0.4, 1.0 };
    GLfloat local_view[] = { 0.0 };
    glClearColor(0.0, 0.1, 0.1, 0.0);
    glEnable(GL_DEPTH_TEST);
    glShadeModel(GL_SMOOTH);

    glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
    glLightfv(GL_LIGHT0, GL_POSITION, position);
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lmodel_ambient);
    glLightModelfv(GL_LIGHT_MODEL_LOCAL_VIEWER, local_view);

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
}

/*Draw twelve spheres in 3 rows with 4 columns .
 * The spheres in the first row have materials with no ambient
reflection.
 * The second row has materials with significant ambient reflection.
 * The third row has materials with colored ambient reflection.
 * The first column has materials with blue, diffuse reflection only
 * The second column has blue diffuse reflection, as well as specular
 * reflection with a low shininess exponent.
```

```

*The third column has blue diffuse reflection, as well as specular
* reflection with a high shininess exponent (a more concentrated
highlight.)
* The fourth column has materials which also include an emissive
component.
*
* glTranslatef() is used to move spheres to their appropriate
locations.
/*

void display(void)
{
    GLfloat no_mat[] = { 0.0, 0.0, 0.0, 1.0 };
    GLfloat mat_ambient[] = { 0.7, 0.7, 0.7, 1.0 };
    GLfloat mat_ambient_color[] = { 0.8, 0.8, 0.2, 1.0 };
    GLfloat mat_diffuse[] = { 0.1, 0.5, 0.8, 1.0 };
    GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat no_shininess[] = { 0.0 };
    GLfloat low_shininess[] = { 5.0 };
    GLfloat high_shininess[] = { 100.0 };
    GLfloat mat_emission[] = {0.3, 0.2, 0.2, 0.0};

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    /*draw sphere in first row, first column
    * diffuse reflection only; no ambient or specular
    */
    glPushMatrix();
    glTranslatef (-3.75, 3.0, 0.0);
    glMaterialfv(GL_FRONT, GL_AMBIENT, no_mat);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
    glMaterialfv(GL_FRONT, GL_SPECULAR, no_mat);
    glMaterialfv(GL_FRONT, GL_SHININESS, no_shininess);
    glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
    glutSolidSphere(1.0, 16, 16);

```

```

*/draw sphere in first row, fourth column
* diffuse reflection; emission; no ambient or specular reflection
/*
    glPushMatrix();
    glTranslatef (3.75, 3.0, 0.0);
    glMaterialfv(GL_FRONT, GL_AMBIENT, no_mat);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
    glMaterialfv(GL_FRONT, GL_SPECULAR, no_mat);
    glMaterialfv(GL_FRONT, GL_SHININESS, no_shininess);
    glMaterialfv(GL_FRONT, GL_EMISSION, mat_emission);
    glutSolidSphere(1.0, 16, 16);
    glPopMatrix();
*/draw sphere in second row, first column
* ambient and diffuse reflection; no specular
/*
    glPushMatrix();

```



```

*/draw sphere in third row, first column
* colored ambient and diffuse reflection; no specular
/*
glPushMatrix();
glTranslatef (-3.75, -3.0, 0.0);
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient_color);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, no_mat);
glMaterialfv(GL_FRONT, GL_SHININESS, no_shininess);
glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
glutSolidSphere(1.0, 16, 16);
glPopMatrix();

*/draw sphere in third row, second column
* colored ambient, diffuse and specular reflection; low shininess
/*
glPushMatrix();
glTranslatef (-1.25, -3.0, 0.0);
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient_color);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, low_shininess);
glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
glutSolidSphere(1.0, 16, 16);
glPopMatrix();

*/draw sphere in third row, third column
* colored ambient, diffuse and specular reflection; high
shininess
/*
glPushMatrix();
glTranslatef (1.25, -3.0, 0.0);
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient_color);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, high_shininess);
glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
glutSolidSphere(1.0, 16, 16);
glPopMatrix();

*/draw sphere in third row, fourth column
* colored ambient and diffuse reflection; emission; no specular
/*
glPushMatrix();
glTranslatef (3.75, -3.0, 0.0);
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient_color);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, no_mat);

```

```

void reshape(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity()
    if (w <= (h * 2))
        glOrtho (-6.0, 6.0, -3.0*((GLfloat)h*2)/(GLfloat)w,
                 3.0*((GLfloat)h*2)/(GLfloat)w , -10.0,10.0)
    else
        glOrtho (-6.0*(GLfloat)w/(GLfloat)h,
                 6.0*(GLfloat)w/(GLfloat)h, -6.0,6.0, -10.0, 10.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void keyboard(unsigned char key, int x, int y)
{
    switch (key){
        case 27:
            exit(0);
            break;
    }
}

void main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize (600, 450);
    glutCreateWindow(argv[0]);
    init();
    glutReshapeFunc(reshape);
    glutDisplayFunc(display);
    glutKeyboardFunc (keyboard);
    glutMainLoop();
}

```

لاحظ أنه لتحديد خصائص كل جسم على حدا تم استخدام تعليمتي `glPushMatrix()` ، `glPopMatrix()` . إن تغيير خصائص الأجسام يسبب كلفة عالية في التنفيذ لذلك يجب التقليل قدر الإمكان من تغيير خصائص الأجسام لتقليل الكلفة ، كما يمكن استخدام التعليمة التالية:

void glColorMaterial(GLenum face, GLenum mode);

لتقليل من كلفة التغيير الناتجة عن تغيير خصائص الأجسام بحيث يأخذ البارمتر `face` إحدى القيمتين الثابنتين التاليتين `GL_FRONT` ، `GL_BACK` ، ويأخذ البارمتر `mode` القيم التالية : `GL_AMBIENT` ، `GL_DIFFUSE` ، `GL_AMBIENT_AND_DIFFUSE` (the default) ، `GL_SPECULAR` ، or `GL_EMISSION`.

وبعد استدعاء التعليمة `glColorMaterial` نكتب التعليمة `glEnable()` مع الثابت `GL_COLOR_MATERIAL` ، ومن ثم يمكنك التحكم باللون باستخدام التعليمة `glColor*()` .

```
glColorMaterial(GL_FRONT, GL_DIFFUSE);
glEnable(GL_COLOR_MATERIAL);
glColor3f(0.2, 0.5, 0.8);
/* draw some objects here */
glColor3f(0.9, 0.0, 0.2);
/* draw other objects here */
glDisable(GL_COLOR_MATERIAL);
```

ولتغيير أي خاصية متعلقة بالأجسام يجب استدعاء التعليمة `glColorMaterial` ، وبما أن الـ `OpenGL` غير قادرة على التحكم بالبارمتر `mode` لكل وجه على حدا ، وإذا كنت تريد إجراء أكثر من تغيير بنفس الوقت عندها استخدم التعليمة `*glMaterial()` مع الانتباه إلى إيقاف فعالية التعليمة `glColorMaterial` .

مثال ٧:

سنستخدم في المثال التالي التعليمة `glColorMaterial()` للتحكم بخصائص الأجسام وتغييرها :

```
#include <GL/gl.h>
#include <GL/glu.h>
#include "aux.h"

GLfloat diffuseMaterial[4] = { 0.5, 0.5, 0.5, 1.0 };

void myinit(void)
{
    GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };

    glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuseMaterial);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialf(GL_FRONT, GL_SHININESS, 25.0);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glDepthFunc(GL_LEQUAL);
```

```
void changeGreenDiffuse (AUX_EVENTREC *event)
{
    diffuseMaterial[1] += 0.1;
    if (diffuseMaterial[1] > 1.0)
        diffuseMaterial[1] = 0.0;
    glColor4fv(diffuseMaterial);
}

void changeBlueDiffuse (AUX_EVENTREC *event)
{
    diffuseMaterial[2] += 0.1;
    if (diffuseMaterial[2] > 1.0)
        diffuseMaterial[2] = 0.0;
    glColor4fv(diffuseMaterial);
}

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    auxSolidSphere(1.0);
    glFlush();
}

void myReshape(GLsizei w, GLsizei h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    _
```


تمرين :

رأينا سابقا المثال التالي كرة موضوعة داخل مكعب وبإعطاء هذه الكرة سرعة ابتدائية تبدأ بالحركة داخل المكعب، وعند اصطدامها بحواف المكعب تتخامد سرعة هذه الكرة حتى تتوقف. سنضيف على هذا المثال مصدر ضوئي متحرك يشبه الشمس بحيث تبدو وكأنها تشرق وعندها تبدأ الكرة بالتحرك ومع تحرك الشمس نحو الغرب تبدأ الكرة بالتخامد إلى أن تتوقف عند غياب الشمس ، ويجب التحكم بالإضاءة التي ترسلها الشمس حيث تختلف من وقت الشروق وحتى وقت المغيب .

