**Chương 5. Ánh sáng**

**Mục tiêu chương**

Sau khi đọc xong chương này, bạn sẽ làm được những điều sau.:

1*.*Hiểu được các điều kiện ánh sáng trong thế giới thực được xấp xỉ như thế nào trong Open GL.

2.Kết xuất các đối tượng được chiếu sáng bằng việc định nghĩa các nguồn sáng và mô hình chiếu sáng mong muốn.

3.Định nghĩa các thuộc tính chất liệu của đối tượng đang được chiếu sáng

4.Thao tác với các ngăn xếp ma trận để điều khiển vị trí nguồn sáng.

Như bạn đã thấy trong Chương 4, OpenGL tính toán màu của từng pixel trong một cảnh hiển thị cuối, được lưu giữ trong bộ đệm. Một phần tính toán này dựa vào việc ánh sáng nào được sử dụng trong cảnh và các đối tượng trong cảnh phản chiếu hay hấp thụ ánh sang đó. Ví dụ, đại dương có độ tươi màu khác nhau khi vào những điều kiện thời tiết khác nhau, ngày nắng thì màu tươi hơn là ngày có mây xám. Sự xuất hiện của tia nắng hay đám mây quyết định bạn thấy đại dương màu xanh lam hay màu xanh xám.Thực tế hầu hết các đối tượng không được thấy dưới dạng ba chiều cho đến khi chúng được chiếu sáng. Hình 5-1 chỉ ra hai phiên bản của cùng một cảnh (một khối cầu đơn), một được chiếu sáng và một không.

****

**Hình 5-1** Một khối cầu được chiếu sáng và một không

Như bạn đã thấy, một khối cầu không được chiếu sáng trông không khác gì một cái đĩa hai chiều. Điều này cho thấy tác động qua lại chặt chẽ giữa các đối tượng và ánh sáng trong việc tạo ra một khung cảnh ba chiều.

Với OpenGL, bạn có thể tính toán ánh sáng và các đối tượng trong một khung cảnh để tạo nhiều hiệu ứng khác nhau. Chương này sẽ bắt đầu với phép khử mặt khuất. Sau đó nó giải thích làm thế nào để điều khiển ánh sáng trong một khung cảnh, thảo luận về khái niệm ánh sáng của OpenGL, và mô tả chi tiết làm thế nào để thiết lập các thông số chiếu sáng để đạt được hiệu ứng nhất định. Vào cuối chương, sẽ là các phép tính toán để xác định việc chiếu sáng ảnh hưởng như thế nào đến màu sắc được biểu diễn.

Chương học sẽ gồm những phần chính sau:

" Khử mặt khuất " Mô tả cơ bản về phép loại bỏ mặt khuất từ khung nhìn.

"Thế giới thực và chiếu sáng trong OpenGL" Giải thích tổng quát về cách ánh sáng thể hiện trên thế giới này và cách OpenGL mô tả thể hiện đó.

"Một ví dụ đơn giản: Tô vẽ một khối cầu được chiếu sáng" giới thiệu việc chiếu sáng bằng OpenGL bằng việc đưa ra một đoạn chương trình ngắn để hiển thị một hình cầu được chiếu sáng.

"Tạo các nguồn sáng" Giải thích cách định nghĩa và xác định vị trí nguồn sáng.

"Lựa chọn một mô hình chiếu sáng" thảo luận về các phần tử của một mô hình chiếu sáng và làm thế nào để xác định chúng.

"Định nghĩa các thuộc tính chất liệu" giải thích làm thế nào để mô tả các thuộc tính của đối tượng sao cho chúng tương tác với ánh sáng theo cách mong muốn.

"Phép toán của ánh sáng" đưa ra các tính toán toán học được sử dụng bởi OpenGL để quyết định hiệu ứng của ánh sáng trong một khung cảnh.

"Ánh sáng trong chế độ chỉ số màu " giải thích sự khác nhau giữa chế độ RGBA và mô hình chỉ số màu trong việc chiếu sáng.

**Khử mặt khuất**

Với mục này, bạn bắt đầu vẽ các đối tượng ba chiếu được tô bóng, trong phần đầu, với các đa giác được tô bóng, điều rất quan trọng là vẽ các đối tượng mà chúng gần vị trí quan sát và loại bỏ các đối tượng bị che khuất bởi các đối tượng khác gần mắt bạn hơn. Khi bạn vẽ một khung cảnh gồm các đối tượng ba chiều, một số trong chúng có thể che toàn bộ hoặc từng phần các đối tượng khác. Việc thay đổi điểm nhìn của bạn (viewpoint) có thể thay đối mối quan hệ che khuất. Lấy ví dụ, nếu bạn nhìn khung cảnh từ hướng đối diện, các đối tượng trước đó đứng trước các đối tượng khác bây giờ sẽ đứng sau nó. Để vẽ một khung cảnh giống thực, các mối quan hệ che khuất phải được duy trì. Giả sử đoạn mã của bạn hoạt động giống như sau:

while (1) {

get\_viewing\_point\_from\_mouse\_position();

glClear(GL\_COLOR\_BUFFER\_BIT);

draw\_3d\_object\_A();

draw\_3d\_object\_B();

}

Với một số vị trí của chuột, đối tượng A có thể che khuất đối tượng B. Với một số vị trí khác, có thể xảy ra điều ngược lại. Nếu không có gì đặc biệt được thực hiện, đoạn mã trên luôn vẽ đối tượng B thứ hai (và do đó ở phía trên đối tượng A) không quan tâm đến vị trí điểm nhìn được chọn. Trong một khung cảnh tệ nhất, nếu đối tượng A và B giao nhau sao cho một phần của đối tượng A che khuất đối tượng B và một phần của B che khuất A, việc thay đổi thứ tự vẽ không cho ta một lời giải cho trường hợp này.  
Việc loại bỏ từng phần của đối tượng rắn bị che khuất bởi các đối tượng khác được gọi là loại bỏ mặt khuất (*hidden-surface removal*). Các dễ dàng nhất để đạt được việc loại bỏ mặt khuất là hãy sử dụng bộ đệm sâu (đôi khi được gọi là bộ đêm z). (Xem thêm Chương 10.)  
 Một bộ đệm sâu hoạt động bằng việc kết hợp một độ sâu, hoặc khoảng cách, từ mặt phẳng quan sát (view plane  thường là mặt cắt gần - near clipping plane), với mỗi pixel trên cửa sổ. Ban đầu, giá trị độ sâu cho tất cả các pixel được đặt là khoảng cách lớn nhất có thể (thường làm mặt cắt xa - far clipping plane) bằng việc sử dụng câu lệnh **glClear()** với tham số GL\_DEPTH\_BUFFER\_BIT. Tiếp theo các đối tượng trong khung cảnh được vẽ theo một thứ tự tùy ý.

Các tính toán đồ họa trong phần cứng hoặc phần mềm chuyễn mỗi mặt phẳng được vẽ thành một tập các pixel trên cửa sổ trong đó mặt phẳng sẽ xuất hiện nếu nó không bị che khuất bới mặt khác. Ngoài ra, khoảng cách từ mặt quan sát được tính. Với bộ đệm sâu được kích hoạt, trước khi mỗi pixel được vẽ một phép so sánh được thực hiện với giá trị độ sâu đã được lưu tại pixel này. Nếu pixel mới gần hơn (đứng trước) pixel ở vị trí đó, màu sắc và giá trị độ sâu của pixel mới thay thế các giá trị hiện hành được ghi vào pixel. Nếu độ sâu của pixel mới lớn hơn giá trị hiện hành tại đó, pixel mới bị che khuất, và thông tin màu sắc và độ sâu của pixel mới bị bỏ qua.

Để sử dụng bộ đệm sâu, bạn cần kích hoạt bộ đệm sâu. Điều này có thể chỉ được thực hiện một lần. Trước khi vẽ, mỗi lần bạn vẽ khung cảnh, bạn cần xóa bộ đệm sâu và sau đó vẽ các đối tượng theo thứ tự tùy ý.

Để chuyển đổi đoạn mã trên sao cho nó thực hiện việc loại bỏ mặt khuất, hãy chỉnh sửa nó thành như sau:

glutInitDisplayMode (GLUT\_DEPTH | .... );

glEnable(GL\_DEPTH\_TEST);

...

while (1) {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

get\_viewing\_point\_from\_mouse\_position();

draw\_3d\_object\_A();

draw\_3d\_object\_B();

}

Đối số của **glClear()** xóa cả bộ đệm sâu và màu sắc

Việc kiểm tra bộ đêm chiều sâu có thể ảnh hướng đến thực thi ứng dụng của bạn. do thông tin bị loại bỏ hơn là sử dụng cho việc vẽ, khử mặt khuất có thể làm tăng thi hành của bạn một chút. Tuy nhiên, cài đặt bộ đệm chiều sâu của bạn có thể có ảnh hưởng lớn nhất đến việc thi hành. Một bộ đệm “phần mềm”(được cài đặt với xử lí bộ nhớ) có thể chậm hơn nhiều so với một bộ đệm phần cứng chuyên dụng.

**Thế giới thực và chiếu sáng trong OpenGL**

Khi bạn nhìn vào một mặt vật chất, mắt của bạn cảm nhận màu sắc phụ thuộc vào sự phân bố của năng lượng photons tác động vào các tế bào võng mạc. (Xem "Khái niệm" trong Chương 4.) Các photon này đến từ một nguồn sáng hoặc kết hợp của các nguồn sáng, một số trong số chúng bị hấp thụ và một số chúng bị phản xạ bởi mặt vật chất. Ngoài ra, các mặt khác có thể có rất nhiều thuộc tính khác nhau một số bóng sáng và thường phản xạ ánh sáng theo các hướng nhất định, trong khi đó một số khác phân tán ánh sáng chiếu đến một cách đều nhau cho tất cả các hướng. Hầu hết các bề mặt chất liệu đều nào ở đâu đó giữa hai mức này.

OpenGL xấp xỉ ánh sáng và chiếu sáng bằng cách tách ánh sáng thành các phần red, green và blue. Do đó, màu sắc của các nguồn sáng được mô tả bằng lượng ánh sáng red, green và blue chúng phát ra, và chất liệucủa mặt được mô tả bằng phần trăm của các ánh sáng thành phần red, green và blue chiếu tới chúng được phản xạ theo các hướng khác nhau. Các phương trình ánh sáng OpenGL chỉ là một xấp xỉ nhưng là xấp xỉ hoạt động khá tốt và có thể được tính tương đối nhanh. Nếu bạn muốn một mô hình chiếu sáng chính xác hơn (hay gì khác), bạn phải làm các tính toán của riêng bạn trong phần mềm. Phần mềm như vậy có thể rất phức tạp, khi mất hàng giờ đọc chủ đề bất kỳ có thể thuyết phục bạn.

Trong mô hình chiếu sáng OpenGL, ánh sáng trong một khung cảnh từ một số nguồn sáng có thể được bật hoặc tắt lần lượt. Một số ánh sáng xuất phát từ một hướng hoặc vị trí cụ thể, và một số nguồn sáng nhìn chung là phân tán khắp khung cảnh. Lấy ví dụ, khi bạn bật một bóng đèn trong phòng, hầu hết ánh sáng xuất phát từ bóng đèn, nhưng có một số ánh sáng xuất đến sau khi tắt gặp một, hai, ba hoặc nhiều bức tường. Những ánh sáng này (gọi là ánh sáng môi trường) được giả thiết là phân tán tới mức sao cho không biết hướng ban đầu của nó ở đâu, nhưng nó sẽ biến mất nếu một nguồn sáng cụ thể bị tắt.

Cuối cùng, có thể có một nguồn sáng bao quanh trong cảnh mà đến không từ một nguồn cụ thể, như thể nó bị phân tán nhiều lần sao cho nguồn ban đầu của nó không xác định.

Trong mô hình OpenGL, các nguồn sáng chỉ có một ảnh hưởng khi có các mặt hấp thụ và phản xạ ánh sáng. Mỗi mặt được giả thiết là được kết hợp của một chất liệuvới nhiều tính chất khác nhau. Một chất liệu có thể tự phát ánh sáng của nó (giống đèn pha ô tô), nó có thể phát tán một số ánh sáng đến theo mọi hướng, và nó có thể phản xạ một số phần ánh sáng đến theo một số hướng nào đó giống như một tấm gương hoặc mặt bóng nào đó.

Mô hình chiếu sáng OpenGL xét chiếu sáng được chia thành bốn thành phần độc lập nhau: tự phát (emissive), môi trường (ambient), khuếch tán (diffuse), và phản xạ (specular). Tất cả bốn thành phần được tính một cách độc lập và sau đó được kết hợp với nhau.

**Ambient, Diffuse, and Specular Light**

Chiếu sáng môi trường A*mbient* là ánh sáng nó được phân tán bởi môi trường nhiều tới mức hướng của nó không thể xác định nó dường như xuất phát từ mọi hướng. Ánh sáng trong một phòng có thành phần môi trường lớn, bởi hầu hết ánh sáng tới mắt bạn đã tác động đến rất nhiều bề mặt. Một ánh sáng đèn pha ngoài cửa (spotlight outdoors) có rất ít thành phần ánh sáng môi trường; hầu hết ánh sáng phát đi theo cùng hướng, và do bạn đang ở ngoài cửa, rất ít ánh sáng đến mặt bạn sau khi đã chạm các đối tượng khác, Khi ánh sáng môi trường chạm tới một măt, nó phân tán đều theo tất cả các hướng.  
Thành phần khuyếch tán là ánh sáng xuất phát từ một hướng, do đó nó sáng hơn nếu nó chiếu vuông góc trên mặt cong hơn là khi nó chiếu xiên trên mặt cong, Một khi nó chạm một mặt cong, tuy nhiên, nó phân tán đều theo mọi hướng, do đó nó xuất hiện sáng đều, không phụ thuộc và vị trí mắt nhìn. Bất kỳ ánh sáng đến từ một vị trí cụ thể hoặc hướng cụ thể có thể có một thành phần khuyếch tán.

Cuối cùng, ánh sáng phản xạ xuất phát từ một hướng cụ thể và nó có xu hướng chạm vào mặt cong theo một hướng nhất định. Một chùm sáng lazer chạm vào một gương chất lượng cao tạo ra sự phản xạ gần 100%. Kim loại và nhựa sáng bóng có thành phần phản xạ cao, và viên phấn hoặc tấm thảm hầu như không có thành phần này. Bạn có thể nghĩ tính phản xạ như là độ bóng.

Mặc dù một nguồn sáng gửi đi cùng một tần số, các thành phần môi trường, khuyếch tán và phản xạ có thể khác nhau. Lấy ví dụ, nếu bạn có một ánh sáng trắng trong một phòng có tường đỏ, ánh sáng phát tán có xu hướng màu đỏ, mặc dù ánh sáng tác động trực tiếp vào đối tượng là màu trắng. OpenGL cho phép bạn đặt các giá trị red, green, và blue cho mỗi thành phần ánh sáng một cách độc lập.

**Màu sắc chất liệu**

Mô hình chiếu sáng OpenGL tạo xấp xỉ mà màu sắc của một chất liệuphụ thuộc vào phần trăm của ánh sáng red, green và blue tới được nó phản xạ. Lấy ví dụ, một quả cầu đỏ hoàn hảo phản xạ tất cả ánh sáng đỏ tới và hấp thụ toàn bộ ánh sáng green và blue tới nó. Nếu bạn quan sát một quả cầu như vậy bằng ánh sáng trắng (tạo bởi các lượng bằng nhau của ánh sáng red, green và blue), tất cả ánh sáng red được phản xạ, và bạn nhìn thấy một quả cầu màu đỏ. Nếu quả cầu được quan sát bằng ánh sáng đỏ thuần khiết, nó cũng xuất hiện màu đỏ. Tuy nhiên, nếu quả cầu được quan sát bằng ánh sáng green thuần khiết, nó xuất hiện là màu đen (tất cả ánh sáng green bị hấp thụ, và không có ánh sáng đỏ tới nó, nên không có ánh sáng nào được phản xạ).

Giống ánh sáng, chất liệucó các màu sắc môi trường, khuyếch tán và phản xạ khác nhau, chúng quyết định sự phản xạ môi trường, khuyếch tán và phản xạ của chất liệu. Một sự phản xạ môi trường của chất liệuđược kết hợp với thành phần môi trường của mỗi nguồn sáng tới, phản xạ khuyếch tán với thành phần khuyếch tán của ánh sáng, và tương tự cho phản xạ phản chiếu. Sự phản xạ môi trường và khuyếch tán quyết định màu sắc của chất liệuvà thường tương tự nếu không trùng nhau. Phản xạ phản chiếu thường là trắng hoặc xám, do đó điểm sáng phản xạ sáng nhất trở thành màu sắc của cường độ phản xạ của nguồn sáng . Nếu bạn nghĩ về một ánh sáng trắng chiếu trên một quả cầu nhựa đỏ sáng bóng, tất cả quả cầu xuất hiện màu đỏ, nhưng điểm chiếu sáng nhất vẫn là màu trắng.  
  
  
Ngoài màu sắc môi trường, khuyếch tán và phản xạ, chất liệu có một màu phát sáng, nó như ánh sáng xuất phát từ một đối tượng. Trong mô hình chiếu sáng OpenGL, màu phát sáng của một mặt cộng với cường độ vào đối tượng, nhưng không bị ảnh hưởng bởi nguồn sáng nào. Mặc dù, màu phát sáng không sinh ra bất kỳ ánh sáng nào vào toàn bộ khung cảnh.

**Các giá trị RGBA cho ánh sáng và chất liệu**

Các thành phần màu sắc được xác định cho ánh sáng có nghĩa gì đó khác với cho chất liệu. Với một ánh sáng, các số ứng với một phần trăm của cường độ lớn nhất cho mỗi màu sắc. Nếu các giá trị R, G, và B cho ánh sáng đều là 1.0, ánh sáng là trắng nhất có thể. Nếu các giá trị đều là 0.5, màu sắc của ánh sáng vẫn là trắng, nhưng nó chỉ có cường độ một nửa, do đó nó có màu xám. Nếu R = G = 1 và B = 0 (cường độ lớn nhất của red và green không có blue), ánh sáng có màu vàng.

Đối với chất liệu, các con số ứng với phần phản xạ của các màu sắc này. Do vậy nếu R=1, G=0.5, và B=0 cho một chất liệu, chất liệuphản xạ tất cả ánh sáng đỏ chiếu tới, nửa ánh sáng green chiếu tới, và không phản xạ ánh sáng blue chiếu tới. Nói cách khác, nếu một ánh sáng OpenGL có các thành phần (LR, LG, LB), và một chất liệucó các thành phần tương ứng (MR, MG, MB), thì, bỏ qua tất cả các hiệu ứng phản xạ khác, ánh sáng đi tới mắt được cho bởi (LR\*MR, LG\*MG, LB\*MB).

Tương tự, nếu bạn có hai ánh sáng chúng gửi (R1, G1, B1) và (R2, G2, B2) đến mắt nhìn, OpenGL cộng các thành phần, cho kết quả (R1+R2, G1+G2, B1+B2). Nếu tổng nào lớn hơn 1 (ứng với một màu sáng hơn khả năng hiển thị của thiết bị), thành phần đó được cho bằng 1.

**Ví dụ đơn giản: Vẽ một khối cầu được chiếu sáng**

Đây là những bước cần thiết để bổ sung chiếu sáng vào khung cảnh của bạn.

* Định nghĩa vectơ pháp tuyến cho mỗi đỉnh của tất cả các đối tượng. Các pháp tuyến này quyết định hướng tương quan của đối tượng các các nguồn sáng.
* Tạo, chọn và đặt vị trí cho một hoặc nhiều nguồn sáng.
* Tạo và chọn một mô hình chiếu sáng (*lighting model*), nó xác định mức ánh sáng môi trường và vị trí điểm nhìn hiệu quả (cho mục đính tính toán chiếu sáng).
* Định nghĩa các thuộc tính chất liệucho các đối tượng trong màn hình.

Ví dụ 5-1 thực hiện các công việc này. Nó hiển thị một hình cầu được chiếu sáng bởi một nguồn sáng duy nhất. như đã chỉ ra trong Hình 5-1

**Ví dụ 5-1** : Vẽ một khối cầu được chiếu sáng: light.c

#include <GL/gl.h>

#include <GL/glu.h>

#include <GL/glut.h>

void init(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 };

glClearColor (0.0, 0.0, 0.0, 0.0);

glShadeModel (GL\_SMOOTH);

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);

glEnable(GL\_DEPTH\_TEST);

}

void display(void)

{

glClear (GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glutSolidSphere (1.0, 20, 16);

glFlush ();

}

void reshape (int w, int h)

{

glViewport (0, 0, (GLsizei) w, (GLsizei) 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, -1.5, 1.5, -10.0, 10.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

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);

glutMainLoop();

return 0;

}

Các lời gọi liên quan đến chiếu sáng trong câu lệnh **init()**; chúng được mô tả ngắn gọn trong các đoạn dưới đây và chi tiết hơn ở phía cuối chương. Một điều chú ý về Ví dụ 5-1 là nó sử dụng chế độ màu RGBA, chứ không phải màu chỉ số. Tính toán chiếu sáng OpenGL là khác nhau trong hai chế độ, và thực tế khả năng chiếu sáng bị hạn chế hơn trong chế độ chỉ số màu. Vì vậy, RGBA là chế độ được ưa dùng khi thực hiện chiếu sáng, và trong tất cả các ví dụ trong chương này sử dụng nó. (Xem"chiếu sáng trong chế độ chỉ số màu " để biết thêm thông tin về ánh sáng trong chế độ.)

**Định nghĩa các vectơ pháp tuyến cho mỗi đỉnh của mọi đối tượng**  
Các pháp tuyến của một đối tượng quyết định hướng tương quan đối với nguồn sáng. Đối với mỗi đỉnh, OpenGL sử dụng pháp tuyến có dấu để quyết định bao nhiêu ánh sáng mà đỉnh này nhận được từ mỗi nguồn sáng. Trong ví dụ này, các pháp tuyến cho hình cầu được định nghĩa là thành phần của thủ tục **glutSolidSphere()**. (Xem "Vecto pháp tuyến" trong Chương 2 để biết thêm chi tiết về cách xác định pháp tuyến.)

**Tạo, đặt vị trí, và kích hoạt một hoặc nhiều nguồn sáng**

Ví dụ 5-1 sử dụng duy nhất một nguồn sáng trắng; vị trí của nó được xác định bằng lời gọi **glLightfv()**. Ví dụ này sử dụng màu sắc mặc định cho ánh sáng không (GL\_LIGHT0), đó là màu trắng; nếu bạn muốn một ánh sáng màu khác, hãy sử dụng **glLight\*()** để xác định nó. Bạn có thể gộp ít nhất tám nguồn sáng khác nhau trong khung cảnh của bạn với các màu sắc khác nhau; màu sắc mặc định cho các ánh sáng khác là màu đen. (Trình biên dịch OpenGL bạn đang sử dụng có thể cho phép nhiều hơn tám.) Bạn cũng có thể đặt ánh sáng ở bất cứ chỗ nào bạn muốn bạn có thể đặt chúng gần khung cảnh, như là một đèn bàn, hoặc một khoảng cách xa vô tận, như là mặt trời. Ngoài ra, bạn có thể kiểm soát khi nào một ánh sáng sinh ra chùm sáng hẹp, hội tụ hoặc chùm sáng rộng hơn. Hãy nhớ rằng mỗi nguồn sáng làm tăng đáng kể số phép toán cần để hiển thị khung hình, do quá trình hiển thị chịu tác động bởi số nguồn sáng trong khung cảnh. (Xem "Tạo cá nguồn sáng" để biết thêm thông tin về cách tạo nguồn sáng với đặc tính mong muốn.)

Sau khi bạn xác định các đặc điểm của các ánh sáng bạn muốn, bạn phải kích hoạt chúng bằng câu lệnh **glEnable()**. Bạn cũng cần gọi **glEnable()** với GL\_LIGHTING là một tham số để chuẩn bị OpenGL thực hiện các tính toán chiếu sáng. (Xem "Kích hoạt nguồn sáng" để biết thêm thông tin)

**Chọn mô hình chiếu sáng**

Như bạn có thể mong đợi, câu lệnh **glLightModel\*()** mô tả các tham số của một mô hình chiếu sáng. Trong Ví dụ 5-1, phần tử duy nhất của mô hình chiếu sáng được xác định tường minh là ánh sáng môi trường toàn bộ. Mô hình chiếu sáng cũng xác định khi nào người quan sát khung cảnh sẽ được coi là ở khoảng cách vô tận hoặc gần khung cảnh, và khi nào các tính toán chiếu sáng sẽ được thực hiện khác cho các mặt trước và sau của các đối tượng trong khung cảnh. Ví dụ 5-1 sử dụng các cài đặt mặc định cho hai mặt này của mô hình  một người quan sát xe vô tận và chiếu sáng một mặt. Việc sử dụng một quan sát địa phương làm tăng đáng kể độ phức tạp tính toán phải thực hiện, bởi OpenGL phải tính toán góc giữa điểm nhìn và mỗi đối tượng. Với một quan sát vô tận, tất nhiên, góc bị bỏ qua, và kết quả kém thực tế hơn. Ngoài ra, do trong ví dụ này, mặt đen của hình cầu không bao giờ nhìn thấy (nó nằm trong hình cầu), chiếu sáng một mặt là đủ. (Xem"Lựa chọn một mô hình chiếu sáng" để thấy mô tả chi tiết hơn các phần tử của một mô hình chiếu sáng OpenGL.)

**Xác định các thuộc tính chất liệucho các đối tượng trong khung cảnh**

Các thuộc tính chất liệucủa một đối tượng quyết định cách nó phản xạ ánh sáng và do vậy cho thấy nó được tạo ra từ chất liệugì. Bởi vì sự tương tác giữa mặt chất liệucủa một đối tượng và ánh sáng là phức tạp, việc xác định các tính chất của chất liệusao cho một đối tượng có hình dạng mong muốn là một nghệ thuật. Bạn có thể xác định màu sắc môi trường, khuyếch tán, và phản xa và cách sáng bóng của nó. Trong ví dụ này, chỉ hai thuộc tính chất liệucuối cùng này  màu sắc chất liệuphản xạ và độ bóng  được xác định tường mình (bằng lời gọi **glMaterialfv()**). (Xem "Định nghĩa các thuộc tính chất liệu" để thấy một mô tả và các ví dụ của mọi thông số thuộc tính chất liệu.)

**Một số chú ý quan trọng**

Như khi bạn viết chương trình chiếu sáng của chính bạn, hãy nhớ rằng bạn có thể sử dụng các giá trị mặc định cho các tham số chiếu sáng; các tham số khác cần sẽ được thay đổi. Đồng thời, hãy đừng quên kích hoạt ánh sáng bạn định nghĩa và kích hoạt các tính toán chiếu sáng. Cuối cùng, hãy nhớ rằng bạn có thể có khả năng sử dụng display lists để tối ưu hoá khi bạn thay đổi các điều kiện chiếu sáng. (Xem "thiết kế khoa học danh sách hiển thị " trong Chương 7.)

**Tạo các nguồn sáng**

Nguồn ánh sáng có một số thuộc tính, chẳng hạn như màu sắc, vị trí và hướng. Các phần sau đây giải thích làm thế nào để kiểm soát các thuộc tính này và ánh sáng kết quả trông sẽ như thế nào. Các lệnh sử dụng để xác định tất cả các thuộc tính của ánh sáng là **glLight \* ()**, nó có ba đối số: Chỉ ra ánh sáng có thuộc tính đang được cần xác định, thuộc tính, và giá trị mong muốn cho thuộc tính đó.

*void glLight{if}(GLenum light, GLenum pname, TYPEparam);*

*void glLight{if}v(GLenum light, GLenum pname, TYPE \*param);*

*Tạo ánh sáng được xác định bởi ánh sáng, chúng có thể là GL\_LIGHT0, GL\_LIGHT1, ... , hay GL\_LIGHT7. Đặc tính của ánh sáng được định nghĩa từ các pname, chúng chỉ ra một tham số được đặt tên (xem Bảng 5-1). Param chỉ ra các giá trị mà thuộc tính pname được thiết lập; nó là một con trỏ đến nhóm giá trị nếu kiểu vecto được sử dụng, hay chính giá trị nếu sử dụng kiểu phi vecto. Kiểu phi vecto có thể được sử dụng để thiết lập duy nhất thuộc tính ánh sáng giá trị đơn.*

**Bảng 5-1**. Các giá trị ngầm định cho tham số pname của glLight\*()





**Chú ý:** Các giá trị ngầm định liệt kê cho GL\_DIFFUSE và GL\_SPECULAR trong Bảng 5-1 chỉ áp dụng cho GL\_LIGHT0. Đối với nguồn sáng khác, giá trị ngầm định là (0.0, 0.0, 0.0, 1.0) cho cả hai GL\_DIFFUSE và GL\_SPECULAR.

Ví dụ 5-2 chỉ ra cách sử dụng **glLight\*()**:

**Example 5-2** : Định nghĩa màu sắc và vị trí cho một nguồn sáng.

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);

Như bạn thấy, các mảng được định nghĩa cho các giá trị thông số, và hàm **glLightfv()** được gọi nhiều lần để thiết các thông số khác nhau. ví dụ, ba lời gọi đầu tiên tới hàm **glLightfv()** là không cần thiết, dochúng được sử dụng để thiết lập các giá trị mặc định cho ba thông số GL\_AMBIENT, GL\_DIFFUSE, và GL\_SPECULAR.

**Chú ý:** luôn kích hoạt **glEnable().** (Xem “Kích hoạt nguồn sáng” để biết thêm thông tin về làm thế nào để thực hiện điều này)

Tất cả các tham số cho **glLight\*()** và giá trị có thể của chúng được giải thích trong các phần sau. Các thông số này tương tác với những cái xác định toàn bộ mô hình chiếu sáng cho một cảnh cụ thể và thuộc tính chất liệu của đối tượng. (Xem "Lựa chọn một mô hình chiếu sáng"và "Xác định thuộc tính chất liệu" để biết thêm thông tin về hai chủ đề này. " Toán học của ánh sáng" giải thích làm thế nào tất cả các tham số này tương tác một cách chính xác.)

**Màu**

OpenGL cho phép bạn kết hợp ba thông số khác nhau liên quan đến màu sắc GL\_AMBIENT, GL\_DIFFUSE, và GL\_SPECULAR - với ánh sáng cụ thể tùy ý. Tham số GL\_AMBIENT dùng để chỉ cường độ RGBA của ánh sáng môi trường bao quanh là một nguồn ánh sáng cụ thể thêm vào cảnh. Như bạn có thể thấy trong Bảng 5-1, mặc định không có ánh sáng bao quanh do GL\_AMBIENT là (0.0, 0.0, 0.0, 1.0). Giá trị này được sử dụng trong Ví dụ 5-1. Nếu chương trình này đã chỉ ra ánh sáng bao quanh màu xanh là

GLfloat light\_ambient [] = {0.0, 0.0, 1.0, 1,0};

glLightfv (GL\_LIGHT0, GL\_AMBIENT, light\_ambient);

Kết quả sẽ được chỉ ra trong hình bên trái của "Bản 13" ở Phụ lục I.   
Các tham số GL\_DIFFUSE hầu như tương quan chặt chẽ nhất với những gì bạn nghĩ tự nhiên là "màu sắc của ánh sáng. " Nó định nghĩa màu RGBA của ánh sáng khuếch tán là một nguồn sáng cụ thể thêm vào cảnh. Ngầm định, GL\_DIFFUSE là (1.0, 1.0, 1.0, 1.0) cho GL\_LIGHT0, chúng tạo ra một nguồn sáng màu trắng sáng như chỉ ra ở phía bên trái của "Bản 13" tại Phụ lục I. Giá trị mặc định cho nguồn sáng bất kỳ khác (GL\_LIGHT1, ..., GL\_LIGHT7) là (0.0, 0.0, 0.0, 0,0).

Các tham số GL\_SPECULAR ảnh hưởng đến màu sắc phản chiếu nổi bật trên một đối tượng. Thông thường, một đối tượng thế giới thực, ví dụ: một chai thủy tinh có phản chiếu nổi bật là màu của ánh sáng chiếu vào nó (thường là màu trắng). Do đó, nếu bạn muốn tạo một hiệu ứng thực tế, thiết lập tham số GL\_SPECULAR có giá trị như tham số GL\_DIFFUSE. Mặc định, GL\_SPECULAR là (1.0, 1.0, 1.0, 1.0) cho GL\_LIGHT0 và (0.0, 0.0, 0.0, 0,0) cho ánh sáng bất kỳ khác.

**Lưu ý:** Các thành phần alpha của những màu sắc này không được sử dụng cho đến khi việc trộn được kích hoạt. (Xem Chương 6.) Cho đến khi, giá trị alpha có thể được bỏ qua.

**Vị trí và sự suy giảm**

Như đã đề cập trước đây, bạn có thể lựa chọn cho dù vị trí nguồn sáng xa vô hạn hay gần hơn. Kiểu đầu được gọi là nguồn sáng *định hướng*, hiệu quả của một vị trí vô hạn là các tia sáng có thể được song song với nhau đồng thời chiếu đến đối tượng. Một ví dụ về một nguồn sáng định hướng trong thế giới thực là mặt trời.

Loại thứ hai được gọi là nguồn sáng *vị trí*, do vị trí chính xác của nó trong cảnh xác định   
hiệu quả của nó trên cảnh và, một cách cụ thể, hướng mà từ đó các tia sáng đi đến. Đèn bàn là một ví dụ về một nguồn sáng vị trí. Bạn có thể thấy sự khác biệt giữa ánh sáng định hướng và vị trí trong "Bản 12" tại Phụ lục I. Ánh sáng sử dụng trong Ví dụ 5-1 là một hướng:

GLfloat light\_position [] = {1,0, 1,0, 1,0, 0,0};

glLightfv (GL\_LIGHT0, GL\_POSITION, light\_position);

Như được chỉ ra, bạn cung cấp một vector với bốn giá trị (x, y, z, w) cho tham số GL\_POSITION. Nếu giá trị cuối cùng, w, là không, nguồn ánh sáng tương ứng là một hướng, và các giá trị (x, y, z) mô tả hướng của nó. Hướng này được biến đổi bởi các ma trận mô hình quan sát. Theo mặc định, GL\_POSITION là (0, 0, 1, 0), chúng định nghĩa một ánh sáng định hướng mà hướng về chiều âm trục z. (Lưu ý rằng không có gì ngăn cản bạn tạo ra một ánh sáng định hướng với hướng (0, 0, 0), nhưng như một ánh sáng như thế sẽ không giúp bạn nhiều.)

Nếu giá trị w là khác không, ánh sáng là vị trí, và giá trị (x, y, z) xác định vị trí của ánh sáng trong tọa độ đối tượng đồng nhất. (Xem Phụ lục F.). Vị trí này được biến đổi bởi các   
ma trận mô hình quan sát và được lưu trữ trong tọa độ mắt nhìn. (Xem "Điều khiển vị trí và hướng của ánh sáng " để biết thêm thông tin về cách điều khiển biến vị trí của ánh sáng.) Ngoài ra, theo mặc định, một ánh sáng vị trí bức xạ theo mọi hướng, nhưng bạn có thể giới hạn thành chiếu sáng hình nó bằng cách định nghĩa ánh sáng là ánh sáng chùm. (Xem "Ánh sáng chùm" để giải thích cách định nghĩa ánh sáng là ánh sáng chùm.)

**Lưu ý**: Nhớ rằng màu sắc trên khuôn mặt của một đa giác tô bóng mịn được xác định bởi tính toán màu cho các đỉnh. Do điều này, bạn có thể muốn tránh việc sử dụng đa giác lớn với ánh sáng cục bộ. Nếu bạn xác định vị trí ánh sáng gần phần giữa của đa giác, các đỉnh có thể quá xa để nhận nhiều ánh sáng, và toàn bộ đa giác sẽ trông tối hơn so với bạn mong đợi. Để tránh vấn đề này, chia đa giác lớn thành những đa giác nhỏ hơn.

Đối với ánh sáng trong thế giới thực, cường độ ánh sáng giảm khi khoảng cách từ ánh sáng tăng. Do ánh sáng định hướng là xa vô hạn, nó không có ý nghĩa trong việc giảm cường độ của nó theo khoảng cách, do đó, sự suy giảm là không có tác dung cho ánh sáng định hướng. Tuy nhiên, bạn có thể muốn giảm ánh sáng từ   
ánh sáng vị trí. OpenGL giảm một nguồn sáng bằng cách nhân gộp các nguồn đó với một   
phân số suy giảm:

  
với d = khoảng cách giữa vị trí của ánh sáng và đỉnh

kc = GL\_CONSTANT\_ATTENUATION

kl = GL\_LINEAR\_ATTENUATION

kq = GL\_QUADRATIC\_ATTENUATION

Mặc định, kc là 1.0 và cả hai kl và kq là không, nhưng bạn có thể cho các thông số giá trị khác nhau:

glLightf (GL\_LIGHT0, GL\_CONSTANT\_ATTENUATION, 2,0);

glLightf (GL\_LIGHT0, GL\_LINEAR\_ATTENUATION, 1.0);

glLightf (GL\_LIGHT0, GL\_QUADRATIC\_ATTENUATION, 0,5);

Lưu ý rằng kết hợp tất cả bao quanh, khuếch tán, và phản chiếu bị suy giảm. Chỉ có các giá trị bao quanh phát ra và toàn cầu không bị suy giảm. Cũng lưu ý rằng do suy giảm yêu cầu thêm một phép chia (và có thể nhiều phép toán hơn) cho mỗi màu sắc tính toán, việc sử dụng ánh sáng suy giảm có thể làm chậm hiệu suất ứng dụng

**Ánh sáng chùm**

Như đã đề cập, bạn có thể có một ánh sáng vị trí hoạt động như một ánh sáng chùm có nghĩa là, bằng cách giới hạn hình dạng ánh sáng phát ra thành hình nón. Để tạo ra ánh sáng chùm, bạn cần phải xác định sự lan tỏa hình nón ánh sáng bạn mong muốn. (Nhớ rằng do ánh sáng chùm là các ánh sáng vị trí, bạn cũng phải xác định vị trí của chúng tại nơi bạn muốn. Một lần nữa, lưu ý rằng không có gì ngăn cản bạn tạo ra một ánh sáng chùm định hướng nhưng nó sẽ không cho bạn kết quả mong muốn). Để xác định góc giữa các trục của hình nón và một tia dọc theo các cạnh của hình nón, sử dụng tham số GL\_SPOT\_CUTOFF. Góc của hình nón tại đỉnh là gấp đôi giá trị này, như chỉ ra trong Hình 5-2.

  
**Hình 5-2**: Tham số GL\_SPOT\_CUTOFF

Lưu ý rằng không có ánh sáng được phát ra ngoài các cạnh của hình nón. Mặc định, đặc điểm ánh sáng chùm là vô hiệu hóa vì tham số GL\_SPOT\_CUTOFF là 180,0. Giá trị này có nghĩa là ánh sáng đó được phát ra theo mọi hướng (góc ở đỉnh của hình nón là 360 độ, do đó, nó không còn là một hình nón). Các giá trị cho GL\_SPOT\_CUTOFF được giới hạn được trong phạm vi [0.0,90.0] (trừ khi nó có giá trị đặc biệt 180,0). Các dòng sau thiết lập tham số cutoff là 45 độ:

glLightf (GL\_LIGHT0, GL\_SPOT\_CUTOFF, 45.0);

Bạn cũng cần chỉ ra hướng của ánh sáng chùm, chúng xác định các trục hình nón của ánh sáng:

GLfloat spot\_direction [] = {-1,0, -1,0, 0,0};

glLightfv (GL\_LIGHT0, GL\_SPOT\_DIRECTION, spot\_direction);

Hướng được xác định theo tọa độ đối tượng. Mặc định, hướng là (0.0, 0.0, -1,0), do đó, nếu bạn không thiết lập rõ giá trị của GL\_SPOT\_DIRECTION, ánh sáng hướng theo chiều âm của trục z. Ngoài ra, luôn nhớ rằng hướng của ánh sáng chùm được biến đổi bởi ma trận mô hình quan sát như thể nó là một vector pháp tuyến, và kết quả được lưu trữ trong tọa độ mắt nhìn. (Xem "Điều khiển vị trí và hướng của ánh sáng” để biết thêm thông tin về biến đổi đó.)

Ngoài góc cutoff và hướng của ánh sáng chùm, có hai cách bạn có thể điều khiển được   
phân bổ cường độ của ánh sáng hình nón. Trước tiên, bạn có thể thiết lập các phân số suy giảm mô tả trước đây, chúng được nhân với cường độ của ánh sáng. Bạn cũng có thể thiết lập các thông số GL\_SPOT\_EXPONENT, chúng được mặc định là không, để điều khiển mức độ tập trung của ánh sáng. Cường độ ánh sáng cao nhất tại tâm của hình nón. Nó suy giảm về phía các cạnh của hình nón theo hàm cos của góc giữa hướng của   
ánh sáng và hướng từ ánh sáng đến các đỉnh đang được chiếu sáng, tăng lũy thừa của số mũ spot.

Do đó, số mũ spot dẫn đến một nguồn sáng hội tụ hơn (Xem " Toán học của Ánh sáng" để biết thêm chi tiết về các phương trình được sử dụng để tính toán cường độ ánh sáng.)   
**Nhiều ánh sáng**

Như đã đề cập, bạn có thể có ít nhất tám nguồn sáng trong cảnh của bạn (có thể nhiều hơn, tùy thuộc vào cài đặt OpenGL của bạn). Do OpenGL cần thực hiện các tính toán để xác định bao nhiêu ánh sáng mỗi đỉnh nhận được từ mỗi nguồn sáng, tăng số lượng ánh sáng ảnh hưởng xấu đến hiệu suất. Các hằng số dùng để chỉ tám nguồn sáng là GL\_LIGHT0, GL\_LIGHT1, GL\_LIGHT2, GL\_LIGHT3, và tiếp tục như thế. Trong các cuộc thảo luận trước đây, các tham số liên quan đến GL\_LIGHT0 đã được thiết lập. Nếu bạn muốn một nguồn sáng bổ sung, bạn cần xác định các tham số của nó, ngoài ra, hãy nhớ rằng các giá trị mặc định là khác nhau cho các nguồn sáng khác này so với GL\_LIGHT0, như được giải thích trong Bảng 5-1. Ví dụ 5-3 định nghĩa một ánh sáng chùm trắng suy giảm.

**Ví dụ 5-3 :** nguồn sáng thứ hai

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\_LIGHT1, GL\_SPOT\_DIRECTION, spot\_direction);

glLightf(GL\_LIGHT1, GL\_SPOT\_EXPONENT, 2.0);

glEnable(GL\_LIGHT1);

Nếu những dòng này được thêm vào Ví dụ 5-1, khối cầu sẽ được chiếu sáng với hai nguồn sáng, một định hướng va một là ánh sáng chùm.

**Thử điều này**

Sửa đổi Ví dụ 5-1 theo cách sau đây

* Thay đổi ánh sáng đầu tiên là một ánh sáng màu vị trí thay cho một ánh sáng trắng định hướng
* Thêm một ánh sáng chùm màu. Gới ý: Sử dụng một số mã được chỉ ra trong phần trước.
* Đánh giá hai thay đổi này ảnh hưởng đến thực thi như thế nào

**Điều khiển một vị trí và hướng của nguồn sáng**

OpenGL xem vị trí và hướng của nguồn sáng chỉ như là vị trí của một hình học nguyên thủy. Nói cách khác, một nguồn sáng là một chủ thể cho các biến đổi ma trận giống nhau như một nguyên thủy.

Đặc biệt hơn, khi **glLight\*()** được gọi để chỉ ra vị trí hay hướng của nguồn sáng, vị trí hay hướng được biến đổi bởi ma trận mô hình quan sát hiện thời và lưu trong tọa độ mắt nhìn. Điều nà có nghĩabạn có thể thao tác một vị trí hay hướng của nguồn sáng bằng việc thay đổi nội dung của ma trận mô hình quan sát. (ma trận phép chiếu không ảnh hưởng đến vị trí hay hướng của ánh sáng). Phần này giải thích làm thế nào để đạt được ba hiệu quả khác nhau sau đây bằng việc thay đổi điểm trong chương trình tại vị trí ánh sáng được thiết lập, liên quan đến các biến đổi mô hình hay quan sát:

* Một vị trí ánh sáng mà được giữ cố định.
* Ánh sáng di chuyển quanh một đối tượng tĩnh
* Ánh sáng di chuyển dọc theo điểm nhìn.

**Giữ ánh sáng cố định**

Trong ví dụ đơn giản nhất, Ví dụ 5-1. Vị trí ánh sáng được giữ cố định. Để đạt được điều này, bạn cần thiết lập vị trí nguồn sáng sau khi mọi biến đổi quan sát và/hoặc mô hình bạn sử dụng. Trong Ví dụ 5-4, mã liên quan từ thủ tục **init()** và **reshape()** trông như sau:

**Ví dụ 5-4 :** Nguồn sáng cố định

glViewport (0, 0, (GLsizei) w, (GLsizei) 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();

/\* later in init() \*/

GLfloat light\_position[] = { 1.0, 1.0, 1.0, 1.0 };

glLightfv(GL\_LIGHT0, GL\_POSITION, position);

Như bạn có thể thấy, các ma trận cổng nhìn và phép chiếu được thiết lập đầu tiên. Sau đó, ma trận đơn vị được nạp như ma trận mô hình quan sát, sau khi vị trí nguồn sáng được thiết lập. Do ma trận đơn vị được sử dụng, vị trí nguồn sáng được chỉ ra lúc đầu (1.0, 1.0, 1.0) không được thay đổi bởi việc nhân với ma trận mô hình quan sát. Sau đó, do hoặc không phải vị trí nguồn sáng hoặc không phải ma trận mô hình quan sát được sửa đổi sau điểm này, hướng của ánh sáng vẫn giữ (1.0, 1.0, 1.0).

**Di chuyển độc lập nguồn sáng**

Bây giờ giả sử bạn muốn quay hay tịnh tiến vị trí nguồn sáng sao cho ánh sáng di chuyển liên quan đến một đối tượng cố định. Một cách để thực hiện điều này là thiết lập vị trí nguồn sáng sau khi biến đổi mô hình, chúng được thay đổi một cách cụ thể để sửa đổi vị trí nguồn sáng. Bạn có thể bắt đầu với hàng loạt lời gọi giống nhau ở **init()** của chương trình. Sau đó bạn cần thực hiện biến đổi mô hình mong muốn (trên ngăn xếp mô hình quan sát) và thiết lập vị trí nguồn sáng, chắc chắn trong **display().** Ví dụ 5-5 chỉ ra **display()** có thể là gì.

**Ví dụ 5-5 :** Di chuyển nguồn sáng độc lập

static GLdouble spin;

void display(void)

{

GLfloat light\_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(spin, 1.0, 0.0, 0.0);

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

glPopMatrix();

glutSolidTorus (0.275, 0.85, 8, 15);

glPopMatrix();

glFlush();

}

spin là một biến toàn cục và được điều khiển chắc chắn bởi một thiết bị đầu và. **display()** tạo ra cảnh để vẽ lại với ánh sáng quay các góc spin quanh một hình đế hoa cố định. Chú ý rằng hai cặp lời gọi **glPushMatrix()** và **glPopMatrix(),** chúng được sử dụng để cách ly các biến đổi quan sát và mô hình, tất cả chúng xảy ra trên ngăn xếp mô hình quan sát. Do trong Ví dụ 5-5 điểm nhìn duy trì hằng số, ma trận hiện thời được đẩy xuống ngăn xếp và sau đó biến đổi quan sát mong muốn được tải với **gluLookAt()**. Ngăn xếp ma trận được đẩy lại một lần nữa trước khi biến đổi mô hình **glRotated()** được chỉ ra. Sau đó vị trí ánh sáng được thiết lập mới, hệ tọa độ quay sao cho ánh sáng tự xuất hiện để quay từ vị trí trước đó của nó. (Nhớ rằng vị trí ánh sáng được lưu trưc trong các hệ tọa độ mắt nhìn, chúng thu được sau phép biến đổi với ma trận mô hình quan sát). Sau khi ma trận quay được loại bỏ khỏi ngăn xếp, đế hoa được vẽ.

Ví dụ 5-6 là một chương trình mà quay một nguồn sáng quanh một đối tượng. Khi nút chuột trái được nhấn, vị trí ánh sáng quay thêm 30 độ. Một hình lập phương khung, không chiếu sáng, nhỏ được vẽ để biểu diễn vị trí của nguồn sáng trong cảnh.

**Ví dụ 5-6 :** Di chuyển nguồn sáng với các biến đổi mô hình: movelight.c

#include <GL/gl.h>

#include <GL/glu.h>

#include "glut.h"

static int spin = 0;

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 ();

glTranslatef (0.0, 0.0, -5.0);

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;

}

}

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);

glutMainLoop();

return 0;

}

**Di chuyển nguồn sáng với điểm nhìn của bạn**

Để tạo một nguồn sáng di chuyển theo điểm nhìn, bạn cần thiết lập ánh sáng vị trí trước khi biến đổi quan sát. Sau đó biến đổi quan sát tác động tới cả hai nguồn sáng và điểm nhìn trong cùng hướng. Nhớ rằng ánh sáng vị trí được lưu trong tọa độ mắt nhìn, và đây là một của một số lần khi tọa độ mắt là tới hạn. Trong Ví dụ 5-7, ánh sáng vị trí được định nghĩa trong **init()**, chúng chứa ánh sáng vị trí tại (0, 0, 0) trong hệ tọa độ mắt. Nói một cách khác, ánh sáng đang chiếu sáng từ thấu kính của máy quay phim.

**Ví dụ 5-7:** Nguồn sáng di chuyển với điểm nhìn

GLfloat light\_position() = { 0.0, 0.0, 0.0, 1.0 };

glViewport(0, 0, (GLint) w, (GLint) h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(40.0, (GLfloat) w/(GLfloat) h, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

Nếu điểm nhìn được di chuyển, ánh sáng sẽ di chuyển với nó, duy trì khoảng cách (0, 0, 0) liên quan đến mắt. Tiếp tục ví dụ 5-7, các biến toàn cục (ex, ey, ez) và (upx, upy, upz) điều khiển vị trí của điểm nhìn và vecto hướng đỉnh. Thủ tục **display()** được gọi trong sự kiện lặp để vẽ lại cảnh như sau:

static GLdouble ex, ey, ez, upx, upy, upz;

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_MASK | GL\_DEPTH\_BUFFER\_MASK);

glPushMatrix();

gluLookAt (ex, ey, ez, 0.0, 0.0, 0.0, upx, upy, upz);

glutSolidTorus (0.275, 0.85, 8, 15);

glPopMatrix();

glFlush();

}

Khi hình đế hoa chiếu sáng được vẽ lại, cả hai vị trí ánh sáng và điểm nhìn được di chuyển đến cùng vị trí. Khi các giá trị truyền đến **gluLookAt()** thay đổi và mắt di chuyển, đối tượng sẽ không thể bị tối vì nó luôn được chiếu sáng từ vị trí mắt nhìn. Thậm chí bạn không thay đổi vị trí nguồn sáng, ánh sáng di chuyển vì hệ tọa độ mắt nhìn thay đổi.

Phương pháp di chuyển ánh sáng này có thể rất có ích cho việc mô phỏng chiếu sáng từ một cái mũ một người thợ mỏ.

Một ví dụ khác là một ngọn nến hay đèn lồng. vị trí ánh sáng được chỉ ra bởi lời gọi **glLightfv**(GL\_LIGHTi, GL\_POSITION, position) sẽ là khoảng cách x, y, và z từ vị trí mắt nhìn đến nguồn chiếu sáng. Sau đó vị trí mắt nhìn di chuyển, ánh sáng sẽ duy trì cùng khoảng cách liên quan.

**Thử điều này**

Sửa đổi Ví dụ 5-6 theo cách sau:

Tạo tịnh tiến ánh sáng quan đối tượng thay cho việc quay quanh nó. Gợi ý: Sử dụng **glTranslated()** thay cho **glRotated()** trong **display()**, và lựa chọn một giá trị thích hợp để sử dụng thay thế cho spin.

Thay đổi sự suy giảm sao cho ánh sáng giảm cường độ khi nó di chuyển xa đối tượng.

Gợi ý: Thêm các lời gọi **glLight\*()** để thiết lập tham số suy giảm mong muốn.

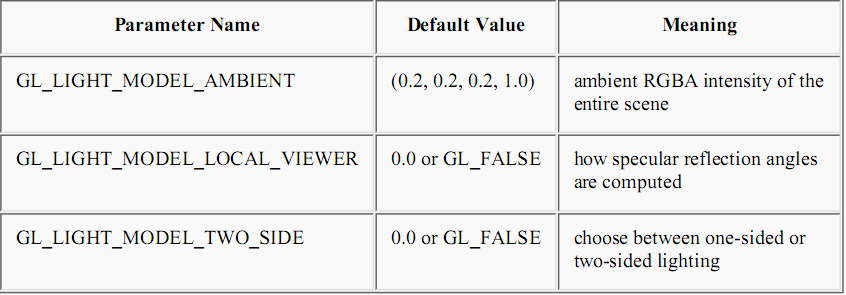
**Lựa chọn một mô hình chiếu sáng**

Khái niệm OpenGL của một mô hình chiếu sáng có ba thành phần:

* Cường độ ánh sáng bao quanh toàn cầu.
* Dù vị trí điểm nhìn là cục bộ đến cảnh hay nó nên được xét là một khoảng cách xa vô hạn.
* Cho dù các tính toán ánh sáng nên được thực hiện khác nhau cho cả hai mặt trước và sau của các đối tượng.

Phần này giải thích làm thế nào để xác định một mô hình chiếu sáng. Nó cũng trình bày cách kích hoạt ánh sáng như thế nào - có nghĩa là nói cho OpenGL bạn muốn các tính toán chiếu sáng được thực hiện như thế nào.  
Các lệnh dùng để xác định tất cả thuộc tính của mô hình chiếu sáng là **glLightModel \* (). glLightModel \* ()** có hai đối số: thuộc tính mô hình chiếu sáng và giá trị mong muốn cho thuộc tính đó.  
*glLightModel void {nếu} (GLenum pname, TYPEparam);  
glLightModel void {nếu} v (GLenum pname, TYPE \* param);  
Thiết lập thuộc tính của mô hình chiếu sáng. Đặc điểm của mô hình chiếu sáng đang thiết lập được xác định bởi pname, chúng chỉ ra một tham số được đặt tên (xem Bảng 5-2). param chỉ ra các giá trị đến các đặc tính pname được thiết lập, nó là một con trỏ trỏ tới một nhóm giá trị nếu kiểu vector được sử dụng, hoặc chính giá trị nếu kiểu phi vector được sử dụng. Các phiên bản phi vector có thể được dùng để thiết lập các đặc điểm mô hình chiếu sáng giá trị đơn duy nhất, không dùng cho GL\_LIGHT\_MODEL\_AMBIENT.*

**Bảng 5-2**: các giá trị ngầm định cho biến pname của glLightModel\*()



**Ánh sáng bao quanh toàn cầu**

Như đã trình bày trước đây, mỗi nguồn sáng có thể góp phần ánh sáng bao quanh cho một cảnh. Ngoài ra, ở đây có thể là ánh sáng bao quanh khác mà không đến từ một nguồn cụ thể. Để chỉ ra cương độ RGBA của ánh sáng bao quanh toàn cầu như thế, sử dụng tham số GL\_LIGHT\_MODEL\_AMBIENT như sau: GLfloat lmodel\_ambient[] = { 0.2, 0.2, 0.2, 1.0 };

glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, lmodel\_ambient);

Trong ví dụ này, các giá trị được sử dụng cho mô hình bao quanh là các giá trị ngầm định cho GL\_LIGHT\_MODEL\_AMBIENT. Do những số này dẫn đến một số lượng nhỏ ánh sáng bao quanh trắng, thậm chí nếu bạ không thêm một nguồn sáng cụ thể đến cảnh của bạn, bạn vẫn có thể nhìn thấy các đối tượng trong cảnh. “Bản 14” trong mục lục I chỉ ra tác động của số lượng ánh sáng bao quanh toàn cầu khác nhau.

**Điểm nhìn cục bộ hay vô hạn**

Vị trí của điểm nhìn tác động các tính toán cho điểm nổi bật được tạo ra bởi phản xạ phản chiếu. Cụ thể hơn, cường độ của điểm nổi bật tại một đỉnh cụ thể phụ thuộc vào pháp tuyến tại đỉnh đó, mỗi hướng từ đỉnh đến nguồn sáng, và hướng từ đỉnh đến điểm nhìn. Nhớ rằng điểm nhìn không thực sự di chuyển bởi lời gọi tới các lệnh chiếu sáng (bạn cần thay đổi biến đổi phép chiếu, như được chỉ ra trong “biến đổi phép chiếu” ở chương 3); thay vào đó, các giả sử khác nhau được tạo cho các tính toán ánh sáng khi điểm nhìn đã được di chuyển.

Với điểm nhìn vô hạn, hướng giữa nó và đỉnh bất kì trong cảnh vẫn là cố định. Các hướng của một điểm nhìn cục bộ dẫn đến nhiều kết quả thục hơn, như do hướng phải được tính toán cho mỗi đỉnh, toàn bộ hiệu suất bị giảm với điểm nhìn cục bộ. Ngầm định, một điểm nhìn vô hạn được sử dụng. Đây là cách để thay đổi đến điểm nhìn cục bộ:

glLightModeli(GL\_LIGHT\_MODEL\_LOCAL\_VIEWER, GL\_TRUE);

Lời gọi này đặt điểm nhìn tại (0, 0, 0) theo tọa độ mắt nhìn. Để chuyển về một điểm nhìn vô hạn, truyền đối số là GL\_FALSE.

**Ánh sáng hai phía**

Các tính toán ánh sáng được thực hiện cho mọi đa giác, cho dù chúng là mặt trước hay mặt sau. Do bạn luôn chủ ý thiết lập các điều kiện ánh sáng với các đa giác mặt trước, tuy nhiên các đa giác mặt sau thường không được chiếu sáng một cách chính xác. Trong Ví dụ 5-1, đối tượng là một khối cầu, chỉ các mặt trước là được nhìn thấy, do chúng là các đa giác bên ngoài của khối cầu. Do đó, trong trường hợp này, nó không thành vấn đề khi các đa giác mặt sau trông như thế nào. Nếu khối cầu được cắt sao cho bề mặt bên trong được nhìn thấy, tuy nhiên, bạn có thể có bề mặt trong được chiếu sáng đầy đủ theo các điều kiện ánh sáng bạn đã định nghĩa; bạn cũng có thể muốn cung cấp một mô tả chất liệu khác nhau đối với các mặt sau. Khi bạn bật ánh sáng hai phía với :

glLightModeli(GL\_LIGHT\_MODEL\_TWO\_SIDE, GL\_TRUE);

OpenGL đảo ngược các pháp tuyến bề mặt cho các đa giác mặt sau; thông thường, điều này có nghĩa là các bề mặt pháp tuyến của các đa giác mặt trước và mặt sau được nhìn thấy, hơn là việc trỏ xa. Kết quả, tất cả các đa giác được chiếu sáng chính xác. Tuy nhiên, các thao tác bổ sung này thường làm cho chiếu sáng hai mặt thực hiện chậm hơn chiếu sáng một mặt ngầm định.

Để tắt chiếu sáng hai mặt, truyền đối số là GL\_FALSE trong lời gọi trước (xem “”) (Xem phần "định nghĩa các thuộc tính chất liệu " để biết thêm thông tin về làm thế nào để cung cấp các thuộc tính chất liệu cho cả hai mặt.) Bạn cũng có thể điều khiển những mặt OpenGL coi là mặt trước với lệnh **glFrontFace()**. (Xem "đảo ngược và lọc các mặt đa giác " trong Chương 2 để biết thêm thông tin.)

**Kích hoạt chiếu sáng**

Với OpenGL, bạn cần phải kích hoạt tường minh (hay hủy) ánh sáng. Nếu ánh sáng không được kích hoạt, màu hiện tạo được ánh xạ đơn giản lên đỉnh hiện tại, và không có tính toán liên quan đến các pháp tuyến, các nguồn sáng, mô hình chiếu sáng, và thuộc tính chất liệu được thực hiện. Đây là lệnh để kích hoạt ánh sáng:

glEnable (GL\_LIGHTING);

Để hủy chiếu sáng, gọi **glDisable ()** với GL\_LIGHTING là đối số.

Bạn cũng cần phải kích hoạt tường minh mỗi nguồn sáng mà bạn định nghĩa, sau khi bạn đã chỉ ra các thông số cho nguồn đó. Ví dụ 5-1 chỉ sử dụng một ánh sáng, GL\_LIGHT0:

glEnable (GL\_LIGHT0);

**Định nghĩa thuộc tính chất liệu**

Bạn đã thấy làm thế nào để tạo ra nguồn ánh sáng với các đặc điểm nhất định và cách xác định ánh sáng mong muốn như thế nào. Phần này mô tả làm thế nào để định nghĩa thuộc tính chất liệu của các đối tượng trong cảnh: bao quanh, khuếch tán, và màu phản chiếu, chiếu sáng, và màu của ánh sáng phát ra (Xem “Toán học của ánh sáng” về các phương trình được sử dụng trong tính toán ánh sáng và thuộc tính chất liệu). Hầu hết các thuộc tính chất liệu có khái niệm tương tự như những thứ bạn đã sử dụng để tạo ra nguồn sáng. Cơ chế để thiết lập chúng là tương tự, ngoại trừ các lệnh sử dụng được gọi là **glMaterial\*().**

*glMaterial void {if} (GLenum mặt, GLenum pname, TYPEparam);  
glMaterial void {if} v (GLenum mặt, pname GLenum, TYPE \* param);*

*Chỉ ra một thuộc tính chất liệu hiện tại để sử dụng trong tính toán chiếu sáng. Mặt có thể là GL\_FRONT, GL\_BACK, hoặc GL\_FRONT\_AND\_BACK để chỉ ra mặt nào của đối tượng, chất liệu nên áp dụng. Các thuộc tính chất liệu cụ thể đang thiết lậpđược xác định bởi pname và giá trị mong muốn cho thuộc tính đã cho đó bởi param, chúng hoặc là một con trỏ tới một nhóm giá trị (nếu kiểu vector được sử dụng) hoặc giá trị thực tế (nếu kiểu phi vector được sử dụng). Các kiểu phi vector chỉ hoạt động cho việc thiết lập GL\_SHININESS. Các giá trị có thể cho pname được chỉ ra trong Bảng 5-3.*

*Lưu ý rằng GL\_AMBIENT\_AND\_DIFFUSE cho phép bạn thiết lập cả hai màu chất liệu bao quanh và khuếch tán một cách đồng thời đến giá trị RGBA.*

**Bảng 5-3** Các giá trị ngầm định cho tham số pname của glMaterial\*()





Như đã nói trong "Lựa chọn một mô hình chiếu sáng", bạn có thể thực hiện các tính toán chiếu sáng khác nhau cho các đa giác mặt trước và mặt sau của đối tượng. Nếu mặt sau có thể được nhìn thấy thực sự, bạn có thể cung cấp đặc tính chất liệu khác nhau cho các bề mặt trước và mặt sau bằng cách sử dụng các tham số mặt của **glMaterial \* ()**. Xem "Bản 14" tại Phụ lục I với một ví dụ của đối tượng được vẽ với thuộc tính chất liệu bên trong và bên ngoài khác nhau.

Cho bạn một ý tưởng về những tác động có thể, bạn có thể đạt được bằng việc thao tác các thuộc tính chất liệu, xem "Bản 16" tại Phụ lục I. Hình vẽ này chỉ ra với cùng một đối tượng được vẽ với một số thiết lập thuộc tính chất liệu khác nhau. Sử dụng nguồn sáng và mô hình chiếu sáng như nhau cho toàn bộ cảnh. Các phần sau đây thảo luận về thiết lập các thuộc tính cụ thể được sử dụng để vẽ mỗi khối cầu này.

Lưu ý rằng hầu hết các thuộc tính chất liệu được thiết lập với **glMaterial\* ()** là các màu (R, G, B, A). Bất kể những giá trị alpha được cung cấp cho các tham số khác, giá trị alpha tại đỉnh cụ thể bất kỳ là giá trị alpha chất liệu khuếch tán (có nghĩa là, giá trị alpha cho GL\_DIFFUSE với lệnh **glMaterial\*** (), trong mô tả phần tiếp theo). (Xem "trộn" trong Chương 6 về một trình bày hoàn chỉnh về các giá trị alpha.) Ngoài ra, không có thuộc tính chất liệu RGBA áp dụng trong chế độ chỉ số màu. (Xem "Ánh sáng trong   
chế độ chỉ số màu "để biết thêm thông tin về những thông số có liên quan trong chế độ chỉ số màu)

**Phản xạ khuếch tán và biên**

Các thông số GL\_DIFFUSE và GL\_AMBIENT thiết lập với **glMaterial \*()** ảnh hưởng đến màu của ánh sáng khuếch tán và biên được đối tượng phản xạ. Phản xạ khuếch tán đóng vai trò quan trọng nhất trong việc xác định bạn cảm nhận như thế nào về màu của đối tượng. Nó bị ảnh hưởng bởi màu của các ánh sáng khuếch tán đi đến và các góc của ánh sáng tới liên quan đến hướng pháp tuyến. (Đây là nơi ánh sáng tới vuông góc với bề mặt). Vị trí của điểm nhìn không ảnh hưởng đến phản xạ khuếch tán ở mọi nơi.

Phản xạ bao quanh ảnh hưởng đến màu sắc tổng thể của đối tượng. Bởi vì phản xạ khuếch tán sáng nhất tại nơi một đối tượng được chiếu sáng trực tiếp, phản xạ bao quanh là dễ thấy nhất một đối tượng không nhận được chiếu sáng trực tiếp. Tổng phản xạ bao quanh của đối tượng bị ảnh hưởng bởi ánh sáng bao quanh toàn cầu và ánh sáng bao quanh từ nguồn sáng đơn. Giống như phản xạ khuếch tán, phản xạ bao quanh không bị ảnh hưởng bởi vị trí của điểm nhìn.

Với các đối tượng trong thế giới thực, phản xạ khuếch tán và bao quanh thông thường có cùng màu. Cho nên, OpenGL cung cấp cho bạn một cách thuận tiện gán giá trị như nhau cho cả hai một cách đồng thời với **glMaterial \* ():**GLfloat mat\_amb\_diff [] = {0,1, 0,5, 0,8, 1,0};   
glMaterialfv (GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE,   
mat\_amb\_diff);   
Trong ví dụ này, màu RGBA (0,1, 0,5, 0,8, 1,0) - một màu xanh dương đậm – biểu diễn phản xạ bao quanh và khuếch tán hiện tại cho cả hai mặt trước và sau của đa giác.

Trong "Bản 16" tại Phụ lục I, dòng đầu tiên của khối cầu không có phản xạ bao quanh (0.0, 0.0, 0.0, 0,0), và dòng thứ hai có một số lượng đáng kể (0,7, 0,7, 0,7, 1,0).   
**Phản xạ phản chiếu**

Phản xạ phản chiếu từ một đối tượng tạo các vị trí nổi bật. Không giống như phản xạ bao quanh và khuếch tán, số phản xạ phản chiếu được xem bởi người quan sát phụ thuộc vào vị trí điểm nhìn - nó sáng nhất theo góc trực tiếp phản xạ. Để thấy điều này, tưởng tượng nhìn quả bóng kim loại ngoài trời dưới ánh nắng.Khi bạn di chuyển đầu của mình, điểm nổi bật được tạo do nắng di chuyển với bạn đến vùng nào đó. Tuy nhiên, nếu   
bạn di chuyển đầu của bạn quá nhiều, bạn sẽ mất toàn bộ vùng nổi bật.

OpenGL cho phép bạn thiết lập các hiệu ứng mà chất liệu có phản xạ ánh sáng (với GL\_SPECULAR) và điều khiển kích thước và độ sáng của vùng nổi bật (với GL\_SHININESS). Bạn có thể gán một số trong miền của [0.0, 128,0] cho GL\_SHININESS – vùng nổi bật có giá trị cao hơn, nhỏ hơn và sáng hơn (tập trung hơn) (Xem "Toán học của Ánh sáng" về các chi tiết vùng phản xạ được tính toán như thế nào.)

Trong "Bản 16" tại Phụ lục I, Các khối cầu trong cột đầu tiên không có phản xạ phản chiếu. Trong cột thứ hai, GL\_SPECULAR và GL\_SHININESS được gán các giá trị như sau:

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);   
Trong cột thứ ba, các tham số GL\_SHININESS được tăng lên 100.0.   
**Phát ra**  
Bằng cách chỉ ra một màu RGBA cho GL\_EMISSION, bạn có thể tạo một đối tượng xuất hiện cho ra ánh sáng của màu đó. Do hầu hết các đối tượng thế giới thực (trừ ánh sáng) không phát ra ánh sáng, bạn sẽ có thể sử dụng chủ yếu đặc điểm này để mô phỏng ánh sáng đèn và các nguồn sáng khác trong một cảnh. Trong "Bản 16" tại Phụ lục I, khối cầu trong cột thứ tư có giá trị màu đỏ xám cho GL\_EMISSION:

GLfloat mat\_emission [] = {0,3, 0,2, 0,2, 0,0};   
glMaterialfv (GL\_FRONT, GL\_EMISSION, mat\_emission);   
Chú ý rằng các khối cầu xuất hiện hơi sặc sỡ một chút, tuy nhiên, chúng không thực sự hoạt động như nguồn sáng. Bạn sẽ cần tạo một nguồn sáng và nó ở cùng vị trí với khối cầu tạo tác động đó.

**Thay đổi các thuộc tính chất liệu**

Ví dụ 5-1 sử dụng thuộc tính chất liệu như nhau cho tất cả các đỉnh của đối tượng duy nhất trong cảnh (hình cầu). Trong tình huống khác, bạn có thể muốn gán thuộc tính chất liệu khác cho các đỉnh khác nhau trên cùng một đối tượng. Nhiều khả năng, bạn có nhiều hơn một đối tượng trong một cảnh, và mỗi đối tượng có thuộc tính chất liệu khác nhau. Ví dụ, mã được viết trong "Bản 16" tại Phụ lục I phải vẽ mười hai đối tượng khác nhau (tất cả khối cầu), mỗi đối tượng với thuộc tính chất liệu khác nhau. Ví dụ 5-8 chỉ ra một phần của các mã trong **display()**.

**Ví dụ 5-8:** Thuộc tính chất liệu khác nhau: material.c

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);

glPopMatrix();

/\* draw sphere in first row, second column

\* diffuse and specular reflection; low shininess; no ambient

\*/

glPushMatrix();

glTranslatef (-1.25, 3.0, 0.0);

glMaterialfv(GL\_FRONT, GL\_AMBIENT, no\_mat);

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 first row, third column

\* diffuse and specular reflection; high shininess; no ambient

\*/

glPushMatrix();

glTranslatef (1.25, 3.0, 0.0);

glMaterialfv(GL\_FRONT, GL\_AMBIENT, no\_mat);

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 first row, fourth column

\* diffuse reflection; emission; no ambient or specular refl.

\*/

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();

Như bạn thấy, **glMaterialfv ()** được gọi nhiều lần để thiết lập các thuộc tính chất liệu mong muốn cho mỗi khối cầu. Lưu ý rằng chỉ cần gọi thay đổi một thuộc tính mà cần xét đến. Khối cầu thứ hai, thứ ba, thứ tư sử dụng cùng thuộc tính bao quanh và khuếch tán như trong khối cầu đầu tiên, do đó, những thuộc tính này không cần xét đến. Do **glMaterial\* ()** có một chi phí thực hiện liên quan đến sử dụng nó, Ví dụ 5-8 có thể được viết lại để giảm thiểu các thuộc tính chất liệu thay đổi. Một kỹ thuật khác để giảm thiểu chi phí thực hiện kết hợp với thuộc tính chất liệu thay đổi là sử dụng **glColorMaterial ()**.

*void* ***glColorMaterial*** *(GLenum mặt, GLenum chế độ);*

*Nguyên nhân các thuộc tính chất liệu(hay thuộc tính) được xác định theo chế độ của bề mặt chất liệu nào đó (hay bề mặt) được chỉ ra bởi bề mặt để theo dõi giá trị của màu hiện tại ở mọi thời điểm. Một thay đổi màu hiện tại (sử dụng* ***glColor\* ()****)cập nhật ngay lập tức các thuộc tính chất liệu chỉ ra. Các tham số bề mặt có thể là GL\_FRONT, GL\_BACK, hoặc GL\_FRONT\_AND\_BACK (mặc định). Các chế độ tham số có thể là GL\_AMBIENT, GL\_DIFFUSE, GL\_AMBIENT\_AND\_DIFFUSE (mặc định), GL\_SPECULAR, hay GL\_EMISSION. Tại mọi thời điểm đã cho, chỉ có một chế độ hoạt động.* ***glColorMaterial ()*** *không ảnh hưởng đến ánh sáng chỉ số màu.*

Lưu ý rằng **glColorMaterial ()** chỉ ra hai giá trị độc lập: giá trị đầu tiên chỉ ra mặt nào hay các mặt được cập nhật, và giá trị thứ hai chỉ ra thuộc tính chất liệu hay thuộc tính các mặt của nó được cập nhật.

OpenGL không duy trì chế độ biến riêng biệt cho mỗi bề mặt.

Sau khi gọi **glColorMaterial ()**, bạn cần phải gọi **glEnable ()** với tham số là GL\_COLOR\_MATERIAL. Sau đó, bạn có thể thay đổi màu sắc hiện tại bằng cách sử dụng **glColor\*()** (hoặc thuộc tính chất liệu khác, bằng cách sử dụng   
**glMaterial \* ()**) cần thiết khi bạn vẽ:

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_DIFFUSE);

/\* now glColor\* changes diffuse reflection \*/

glColor3f(0.2, 0.5, 0.8);

/\* draw some objects here \*/

glColorMaterial(GL\_FRONT, GL\_SPECULAR);

/\* glColor\* no longer changes diffuse reflection \*/

/\* now glColor\* changes specular reflection \*/

glColor3f(0.9, 0.0, 0.2);

/\* draw other objects here \*/

glDisable(GL\_COLOR\_MATERIAL);   
Bạn nên sử dụng **glColorMaterial ()** bất cứ khi nào bạn cần thay đổi một tham số chất liệu đơn đối với hầu hết các đỉnh trong cảnh của bạn. Nếu bạn cần thay đổi nhiều hơn một tham số chất liệu, như trường hợp của "Bản 16 " tại Phụ lục I, sử dụng **glMaterial\*().** Khi bạn không cần khả năng của **glColorMaterial ()** nữa, hãy chắc chắn để hủy nó để bạn không phải nhận thuộc tính chất liệu không mong muốn và không phải chịu các chi phí thực hiện liên quan đến nó. Giá trị hiệu suất sử dụng glColorMaterial () thay đổi tùy thuộc vào cài đặt OpenGL của bạn. Một số cài đặt thực hiện có thể được tối ưu hóa các thủ tục đỉnh để chúng có thể cập nhật nhanh chóng các thuộc tính chất liệu dựa trên màu hiện tại.

Ví dụ 5-9 chỉ ra một chương trình tương tác sử dụng **glColorMaterial ()** để thay đổi các tham số chất liệu.

Nhấn một trong ba nút chuột để thay đổi màu của các phản xạ khuếch tán.

**Ví dụ 5-9**: Sử dụng glColorMaterial (): colormat.c

#include <GL/gl.h>

#include <GL/glu.h>

#include "glut.h"

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

void init(void)

{

GLfloat mat\_specular[] = { 1.0, 1.0, 1.0, 1.0 };

GLfloat light\_position[] = { 1.0, 1.0, 1.0, 0.0 };

glClearColor (0.0, 0.0, 0.0, 0.0);

glShadeModel (GL\_SMOOTH);

glEnable(GL\_DEPTH\_TEST);

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);

glColorMaterial(GL\_FRONT, GL\_DIFFUSE);

glEnable(GL\_COLOR\_MATERIAL);

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glutSolidSphere(1.0, 20, 16);

glFlush ();

}

void reshape (int w, int h)

{

glViewport (0, 0, (GLsizei) w, (GLsizei) 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, -1.5, 1.5, -10.0, 10.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void mouse(int button, int state, int x, int y)

{

switch (button) {

case GLUT\_LEFT\_BUTTON:

if (state == GLUT\_DOWN) { /\* change red \*/

diffuseMaterial[0] += 0.1;

if (diffuseMaterial[0] > 1.0)

diffuseMaterial[0] = 0.0;

glColor4fv(diffuseMaterial);

glutPostRedisplay();

}

break;

case GLUT\_MIDDLE\_BUTTON:

if (state == GLUT\_DOWN) { /\* change green \*/

diffuseMaterial[1] += 0.1;

if (diffuseMaterial[1] > 1.0)

diffuseMaterial[1] = 0.0;

glColor4fv(diffuseMaterial);

glutPostRedisplay();

}

break;

case GLUT\_RIGHT\_BUTTON:

if (state == GLUT\_DOWN) { /\* change blue \*/

diffuseMaterial[2] += 0.1;

if (diffuseMaterial[2] > 1.0)

diffuseMaterial[2] = 0.0;

glColor4fv(diffuseMaterial);

glutPostRedisplay();

}

break;

default:

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);

glutMainLoop();

return 0;

}  
**Thử điều này**

Sửa đổi ví dụ 5-8 theo cách sau đây:

* Thay đổi ánh sáng bao quanh toàn cầu trong cảnh. Gợi ý: Thay đổi giá trị của tham số  
  GL\_LIGHT\_MODEL\_AMBIENT.
* Thay đổi các tham số khuếch tán, bao quanh, và phản xạ phản chiếu, số mũ tô bóng, và màu phát xạ. Gợi ý: Sử dụng lệnh **glMaterial\*()**, nhưng tránh gọi quá nhiều.
* Sử dụng chất liệu hai mặt và thêm một mặt phẳng cắt định nghĩa bởi người dùng để bạn có thể nhìn thấy hàng và cột ở bên trong và bên ngoài của khối cầu. (Xem " Bổ sung các mặt phẳng cắt " trong Chương 3, nếu bạn cần gọi lại các mặt phẳng cắt được định nghĩa bởi người dùng) Gợi ý: Bật ánh sáng hai mặt với

GL\_LIGHT\_MODEL\_TWO\_SIDE, thiết lập các thuộc tính chất liệu mong muốn, và thêm mặt phẳng cắt.

* Hủy bỏ tất cả các lời gọi **glMaterialfv ()**, và sử dụng lời gọi glColorMaterial ()hiệu quả hơn để đạt đượcánh sáng tương tự.

**Toán học của ánh sáng**

**Nâng cao**

Phần này trình bày các phương trình được sử dụng bởi OpenGL thực hiện các tính toán ánh sáng để xác định màu sắc theo chế độ RGBA. (xem “Toán học của ánh sáng chế độ chỉ số màu” để tính toán tương ứng đối với chế độ chỉ số màu). Bạn không cần phải đọc phần này nếu bạn sẵn sàng thử nghiệm để có được những điều kiện ánh sáng mà bạn muốn. Ngay cả sau khi đọc phần này, bạn có thể sẽ phải thử nghiệm, nhưng bạn sẽ có một ý tưởng tốt hơn về cách các giá trị của tham số ảnh hưởng như thế nào đến màu sắc của một đỉnh. Hãy nhớ rằng nếu ánh sáng không được kích hoạt, màu sắc của một đỉnh chỉ đơn giản là màu hiện tại, nếu nó được kích hoạt, các tính toán chiếu sáng mô tả ở đây được thực hiện theo tọa độ mắt nhìn.

Trong các phương trình sau đây, các phép toán trong toán học được thực hiện riêng biệt trên các thành phần R, G, và B. Vì vậy, ví dụ, khi ba điều kiện được chỉ ra như là cộng vào với nhau, các giá trị R, giá trị G, và các giá trị B cho mỗi số hạng được cộng một cách riêng biệt để tạo thành màu RGB cuối cùng (R1 + R2 + R3, G1 + G2 + G3, B1 + B2 + B3). Khi ba mục này được nhân, tính toán là (R1R2R3, G1G2G3, B1B2B3). (Hãy nhớ rằng phần tử cuối A hay thành phần alpha tại một đỉnh bằng khuếch tán chất liệu giá trị alpha tại đỉnh đó.) Các màu sắc được tạo bởi ánh sáng một đỉnh được tính như sau:   
đỉnh màu =

sự phát xạ chất liệu tại đỉnh đó + ánh sáng bao quanh tỉ lệ theo thuộc tính bao quanh của chất liệu tại đỉnh đó + bao quanh , khuếch tán và phản chiếu gộp lại từ mọi nguồn sáng, suy giảm hợp lí.

Sau khi tính toán chiếu sáng được thực hiện, các giá trị màu được co (ở chế độ RGBA) trong miền [0,1]. Chú ý rằng các tính toán chiếu sáng OpenGL không tính đến các khả năng một đối tượng chặn ánh sáng từ đối tượng khác; như kết quả tô bóng không tự động tạo ra. (Xem "tô bóng" trong Chương 14 về kỹ thuật để tạo tô bóng). Ngoài ra lưu ý rằng với OpenGL, các đối tượng chiếu sáng không bức xạ ánh sáng lên đối tượng khác.

**Chất liệu phát xạ**

Thuật ngữ chất liệu phát xạ là đơn giản nhất. Nó la giá trị RGB thiết kế đến các   
tham số GL\_EMISSION.

**Ánh sáng bao quanh toàn cầu theo tỉ lệ**

Thuật ngữ thứ hai được tính toán bằng cách nhân ánh sáng bao quanh toàn cầu (khi xác định tham số GL\_LIGHT\_MODEL\_AMBIENT) theo của thuộc tính bao quanh của chất liệu (giá trị GL\_AMBIENT khi gán với **glMaterial\*()**): ambientlight model \* ambientmaterial

Mỗi giá trị R, G, và B cho hai tham số này được nhân riêng biệt để tính toán giá trị RGB cuối cùng cho các mục này : (R1R2, G1G2, B1B2).

**Cộng gộp các nguồn sáng**

Mỗi nguồn sáng có thể gộp vào màu của đỉnh, và chúng được cộng vào với nhau. Các phương trình tính toán gộp của mỗi nguồn sáng như sau:

contribution = attenuation factor \* spotlight effect \*

(ambient term + diffuse term + specular term)

**Phân số suy giảm**

Các *phân số suy giảm* đã được mô tả trong "Vị trí và suy giảm":



Với d = khoảng cách giữa vị trí của ánh sáng và các đỉnh

kc = GL\_CONSTANT\_ATTENUATION

kl = GL\_LINEAR\_ATTENUATION

kq = GL\_QUADRATIC\_ATTENUATION

Nếu ánh sáng là một hướng, phân số suy giảm là 1.

**Hiệu ứng ánh sáng chùm**

*Hiệu ứng ánh sáng chùm* đánh giá một trong ba giá trị có thể, phụ thuộc vào ánh sáng có thực sự là một ánh sáng chùm hay không và cho dù đỉnh nằm bên trong hay bên ngoài hình nón chiếu sáng được tạo ra bởi ánh sáng chùm:

* 1 nếu ánh sáng không phải là một ánh sáng chùm (GL\_SPOT\_CUTOFF là 180,0).   
  0 nếu ánh sáng là một ánh sáng chùm, nhưng đỉnh nằm bên ngoài hình nón chiếu sáng được tạo ra bởi ánh sáng chùm.
* (Max {v • d, 0}) GL\_SPOT\_EXPONENT với

**v** = (vx, vy, vz) là vectơ đơn vị mà trỏ đến từ ánh sáng chùm (GL\_POSITION) đến đỉnh.

**d** = (dx, dy, dz) là hướng của ánh sáng chùm(GL\_SPOT\_DIRECTION), giả sử ánh sáng là một ánh sáng chùm và đỉnh nằm bên trong hình nón chiếu sáng được tạo ra bởi ánh sáng chùm .

Tích vô hướng của hai vectơ **v** và **d** thay đổi theo cos của góc giữa chúng, do đó,   
các đối tượng trực tiếp trên trục được chiếu sáng tối đa, và các đối tượng ngoài trục được chiếu sáng giảm theo cos của góc.

* Để xác định xem một đỉnh cụ thể nằm trong hình nón chiếu sáng hay không, OpenGL đánh giá (max {v • d, 0}) với v và d được định nghĩa trong trình bày trước. Nếu giá trị nhỏ hơn cos góc giới hạn ánh sáng chùm (GL\_SPOT\_CUTOFF), thì đỉnh nằm bên ngoài hình nón, nếu không, ngược lại nó bên trong hình nón.

**Thuật ngữ bao quanh**

Thuật ngữ bao quanh đơn giản chỉ là màu ánh sáng bao quanh được tỉ lệ theo thuộc tính chất liệu bao quanh:

ambientlight \* ambientmaterial

**Thuật ngữ Khuếch tán**

Thuật ngữ khuếch tán cần đưa vào tính toán xem liệu ánh sáng có chiếu trực tiếp lên đỉnh không, màu khuếch tán của ánh sáng, và thuộc tính chất liệu khuếch tán:

(Max {**L** • **n**, 0}) \* diffuselight \* diffusematerial

trong đó:

**L** = (**L**x, **L**y, **L**z) là vectơ đơn vị trỏ từ đỉnh đến vị trí ánh sáng (GL\_POSITION).   
**n** = (**n**x, **n**y, **n**z) là vectơ pháp tuyến đơn vị tại đỉnh.

**Thuật ngữ phản chiếu**

Thuật ngữ phản chiếu cũng phụ thuộc vào việc ánh sáng chiếu trực tiếp trên đỉnh hay không. Nếu L • n là nhỏ hơn hoặc bằng không, không có thành phần phản chiếu tại đỉnh này. (Nếu nó nhỏ hơn không, ánh sáng ở phía trái của bề mặt). Nếu có một thành phần phản chiếu, nó phụ thuộc vào những điều sau đây:

* Vector pháp tuyến đơn vị tại đỉnh (nx, ny, nz).

Tổng của hai vectơ đơn vị trỏ giữa (1) đỉnh và vị trí ánh sáng (hay hướng ánh sáng   
) và (2) đỉnh và điểm nhìn (giả sử GL\_LIGHT\_MODEL\_LOCAL\_VIEWER là đúng, nếu không, các vector (0, 0, 1) được sử dụng như vector thứ hai trong tổng). vector tổng này được pháp tuyến hóa (bằng cách chia mỗi thành phần cho độ lớn của vector) để dẫn đến s = (sx, sy, sz).

* Thành phần phản chiếu (GL\_SHININESS).
* Màu phản chiếu của ánh sáng (GL\_SPECULARlight).
* Thuộc tính phản chiếu của chất liệu (GL\_SPECULARmaterial).

Sử dụng các định nghĩa này, đây là cách OpenGL tính toán phản chiếu như thế nào:   
(Max {**s • n,** 0}) shininess \* specularlight \* specularmaterial

Tuy nhiên, nếu **L • n** = 0, phản chiếu bằng 0.

**Gộp tất cả cùng nhau**

Sử dụng định nghĩa các thuật ngữ mô tả trong các đoạn trên, để biểu diễn toàn bộ tính toán ánh sáng trong chế độ RGBA:

Vertex color = emissionmaterial +

ambientlight model \* ambientmaterial +



[Ambientlight \* ambientmaterial +

(Max {L • n, 0}) \* diffuselight \* diffusematerial +

(Max {s • n, 0}) shininess \* specularlight \* specularmaterial] i

**Ánh sáng trong chế độ chỉ số màu**

Trong chế độ chỉ số màu, các tham số bao gồm giá trị RGBA hoặc không có hiệu quả hoặc có một giải thích đặc biệt. Do nó khó khăn hơn nhiều để đạt được hiệu quả nhất định trong chế độ chỉ số màu, bạn nên sử dụng RGBA bất cứ khi nào có thể. Thực tế, chỉ có ánh sáng-nguồn, ánh sáng mô hình, hoặc các tham số chất liệu trong một dạng RGBA được sử dụng theo chế độ chỉ số màu là các tham số ánh sáng nguồn GL\_DIFFUSE và GL\_SPECULAR và các tham số chất liệu GL\_SHININESS. GL\_DIFFUSE và GL\_SPECULAR (*d*l và *s*l, một cách tương ứng) được sử dụng để tính toán chỉ số màu và cường độ ánh sáng phản chiếu (*d*ci và *s*ci) như sau:

*d*ci = 0,30 R (*d*l) + 0,59 G (*d*l) + B 0,11 (*d*l)

*s*ci = 0,30 R (*s*l) + 0,59 G (*s*l) + B 0,11 (*s*l)

trong đó R (x), G (x) và B (x) là các thành phần màu đỏ, xanh lá cây, xanh dương, một cách tương ứng, của màu x. Cácgiá trị trọng số 0,30, 0,59 và 0,11 phản ánh được "cảm nhận" trọng lượng có màu đỏ, xanh lá cây, xanh dương cho mắt bạn - mắt bạn nhạy cảm nhất với màu xanh lá cây và ít nhạy cảm với xanh dương.

Để xác định màu chất liệu trong chế độ chỉ số màu, sử dụng **glMaterial \* ()** với tham số đặc biệt GL\_COLOR\_INDEXES, như sau:

GLfloat mat\_colormap [] = {16,0, 47,0, 79,0};

glMaterialfv (GL\_FRONT, GL\_COLOR\_INDEXES, mat\_colormap);

Ba con số cung cấp cho GL\_COLOR\_INDEXES xác định các chỉ số màu cho bao quanh, khuếch tán, và màu chất liệu phản chiếu, một cách tương ứng. Nói cách khác, OpenGL quan tâm đến màu liên kết với chỉ số đầu (16,0 trong ví dụ này) như màu bao quanh tinh khiết, với chỉ số thứ hai (47,0) là màu khuếch tán tinh khiết, và với chỉ số thứ ba (79,0) như màu phản chiếu tinh khiết. (Ngầm định, chỉ số màu bao quanh là 0.0, và chỉ số màu khuếch tán và phản chiếu đều là 1.0. Lưu ý rằng **glColorMaterial ()** không có tác động trên đến ánh sáng chỉ số màu.)

Khi vẽ một cảnh, OpenGL sử dụng các màu kết hợp với các chỉ số ở giữa các số này để tô bóng các đối tượng trong cảnh. Do đó, bạn phải xây dựng một đoạn nối màu giữa các chỉ số được chỉ ra (Ví dụ, giữa chỉ số 16 và 47, và sau đó giữa 47 và 79). Thông thường, các đoạn nối màu được xây dựng mịn, nhưng bạn có thể muốn sử dụng các công thức khác để đạt được hiệu ứng khác nhau. Đây là một ví dụ về đoạn nối màu trơn mà bắt đầu với một màu bao quanh là đen và đi qua màu khuếch tán là tía đến một màu phản chiếu là trắng:

for (i = 0; i <32; i + +) {

glutSetColor (16 + i, 1.0 \* (i/32.0), 0,0, 1,0 \*(i/32.0));

glutSetColor (48 + i, 1.0, 1.0 \* (i/32.0), 1.0);

}   
Lệnh thư viện GLUT **glutSetColor ()** có bốn đối số. Nó liên kết chỉ số màu được chỉ ra bởi các đối số đầu tiên với bộ ba RGB được chỉ ra bởi ba đối số cuối cùng. Khi i = 0, chỉ số màu 16 được gán giá trị RGB (0,0, 0,0, 0,0), hay đen. Đoạn nối màu xây dựng trơn đến màu chất liệu khuếch tán tại chỉ số 47 (khi i = 31), chúng được gán giá trị RGB (1.0, 0.0, 1.0) là màu tía tinh khiết. Vòng lặp thứ hai xây dựng đoạn nối giữa màu khuếch tán tía và màu phản chiếu trắng (1.0, 1.0, 1.0) (chỉ số 79). "Bản 15" tại Phụ lục I chỉ ra kết quả của việc sử dụng đoạn nối màu với một khối cầu đơn được chiếu sáng.   
**Toán học của chiếu sáng chế độ chỉ số màu**

**Nâng cao**

Như bạn mong đợi, do các tham số được phép khác nhau đối với chế độ chỉ số màu hơn là đối với chế độ RGBA, các tính toán khác nhau là tốt. Do không có chất liệu phát xạ và không có ánh sáng bao quanh, chỉ các thuật ngữ quan tâm từ các phương trình RGBA là gộp phản chiếu và khuếch tán từ ánh sáng nguồn và sự bóng sáng. Thậm chí chúng cần được sửa đổi, tuy nhiên, được giải thích trong phần tiếp theo.

Hãy bắt đầu với các thuật ngữ khuếch tán và phản chiếu từ các phương trình RGBA. Trong thuật ngữ khuếch tán, thay vì diffuselight \* diffusematerial, sử dụng *d*ci đã được định nghĩa trong phần trước của chế độ chỉ số màu.

Tương tự như vậy, trong thuật ngữ phản chiếu, thay vì specularlight \* specularmaterial, sử dụng *s*ci đã được định nghĩa ở phần trước. (Tính toán suy giảm, hiệu ứng ánh sáng chùm, và tất cả các thành phần khác của các thuật ngữ trong phần trước.) Gọi các sửa đổi thuật ngữ khuếch tán và phản chiếu là d và s, một cách tương ứng. Bây giờ hãy s '= min {s, 1}, và sau đó tính toán

*c* = *a*m + *d* (1-*s ‘*) (*d*m-*a*m) + *s'* (*s*m-*a*m)

với *a*m, *d*m, và *s*m là các chỉ số bao quanh, khuếch tán, và liệu phản chiếu được chỉ ra bằng cách sử dụng GL\_COLOR\_INDEXES. Chỉ số màu cuối cùng là

c '= min {c, sm}

Sau khi tính toán chiếu sáng được thực hiện, các giá trị chỉ số màu được chuyển đổi thành điểm cố định (với một số lượng không xác định các bit về phía bên phải của điểm nhị phân). Sau đó, các phần nguyên là đánh dấu (bitwise ANDed) với 2n-1, với n là số bit trong một màu của bộ đệm chỉ số màu.