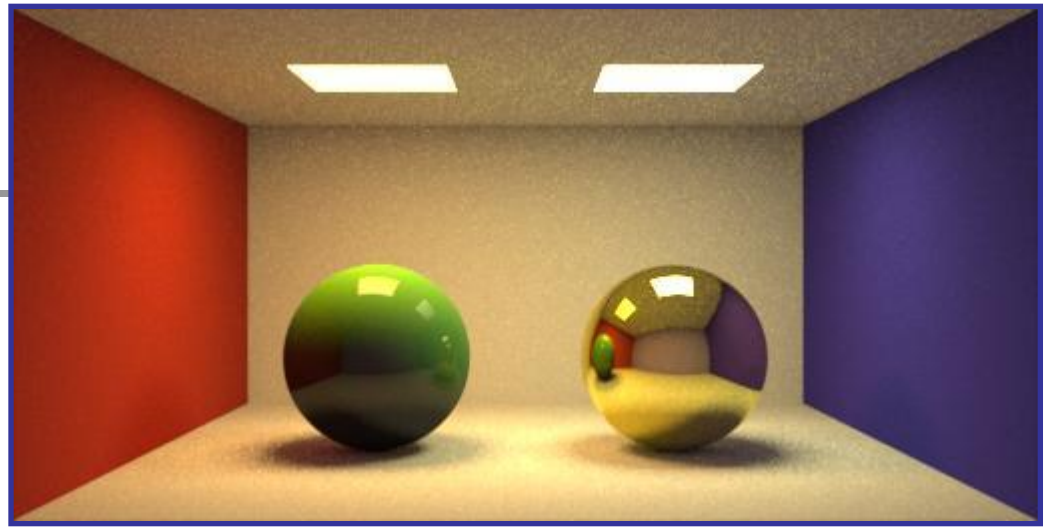
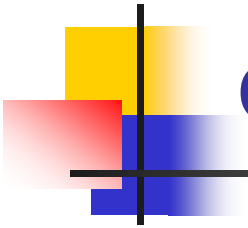


Wykład 4:

Oświetlenie w grafice 3D



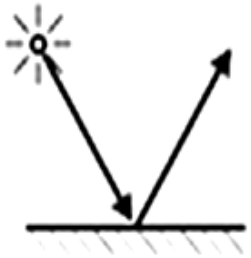
oświetlenie sceny:

Aby osiągnąć realizm obrazu, musimy rozważyć **problem oświetlenia**. Każda powierzchnia reaguje w jej właściwy sposób na padające na nią światło.
Barwy, faktury i inne właściwości przedmiotów postrzegamy dzięki temu, że przedmioty te są oświetlone (lub same emitują światło). Symulacja tych zjawisk pozwala oddać realny wygląd elementów wirtualnej sceny.

Można wyróżnić następujące niezależne przypadki:
- odbicie światła
- przenikanie światła (dla materiałów przezroczystych)
- pochłanianie światła

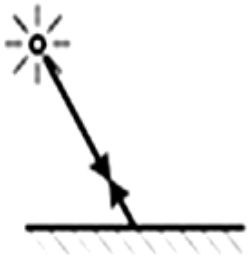
odbicie światła:

Odbicie światła od powierzchni materiału jest zjawiskiem złożonym.
Odbicie światła może być:
- **kierunkowe (lustrzane)** - padający promień odbija się pod kątem równym kątowi padania
- **rozproszone (dyfuzyjne**) - odbicie może być widoczne pod dowolnym kątem.



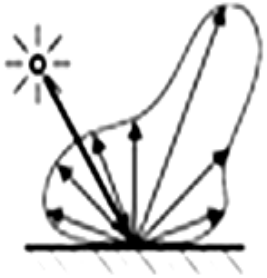
odbicie kierunkowe (idealne)

odbicie kierunkowo rozproszone



odbicie powrotne

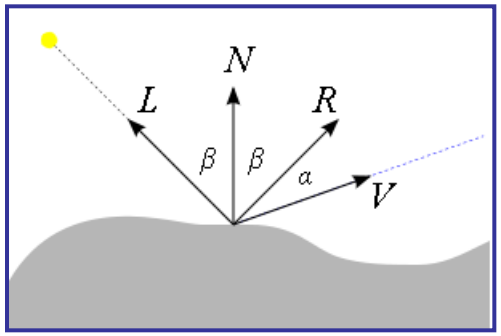
odbicie rozproszone



odbicie rzeczywiste

odbicie lambertowskie:

Model Lamberta: określa natężenie światła rozproszonego, odbitego od danej powierzchni zgodnie ze wzorem: $I_d=k_dI_P(L\circ N)$,
gdzie k_d – współczynnik odbicia światła rozproszonego,
I_P – natężenie światła punktowego,
L – wektor od powierzchni do źródła światła,
N – wektor normalny do danej powierzchni,
β – kąt między wektorami L i N.



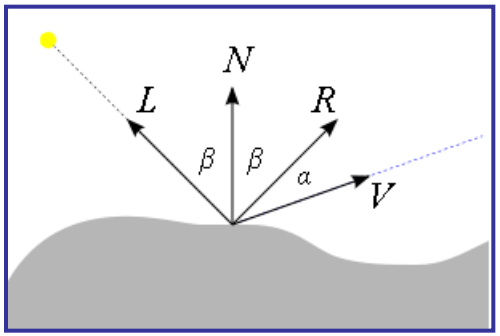
Gdy wektory L i N są znormalizowane, to $L\circ N = |L||N|\cos(\beta) = \cos(\beta)$.
Natężenie światła rozproszonego jest proporcjonalne do cosinusa kąta β.
Prawo Lamberta odnosi się do odbicia światła od idealnej powierzchni (tzw. powierzchni lambertowskiej) rozpraszającej światło jednakowo we wszystkich kierunkach (np. kreda). Takie powierzchnie wydają się równie jasne ze wszystkich kierunków obserwacji. Powierzchnie rzeczywiste odbijają światło zgodnie z prawem Lamberta tylko w pewnym zakresie kąta β.

model Phonga:

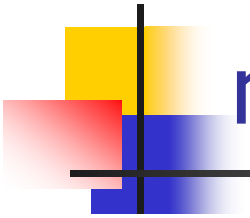
Najstarszy, z praktycznie wykorzystywanych w grafice komputerowej modeli odbicia, zaproponował Bui Tuong Phong (1975).
Model Phonga jest modelem eksperymentalnym, nieuzasadnionym fizycznie i niespełniającym zasady zachowania energii.
Wynikowe natężenie światła I określa wzór:

$$I = k_a I_a + f_{att} k_d I_P \cos(\beta) + f_{att} k_s I_P f(\beta) \cos^n(\alpha)$$

gdzie k_a – współczynnik odbicia światła otoczenia,
I_a – natężenie światła w otoczeniu obiektu,
k_d – współczynnik odbicia światła rozproszonego,
I_P – natężenie światła punktowego,
k_s – współczynnik odbicia światła kierunkowego,
f_{att} – współczynnik tłumienia światła wraz z odległością,
n – współczynnik gładkości powierzchni



N – wektor normalny
L – promień światła
R – promień odbity
V – kierunek do obserwatora



model Phonga:

Na natężenie światła docierającego do obserwatora składają się:

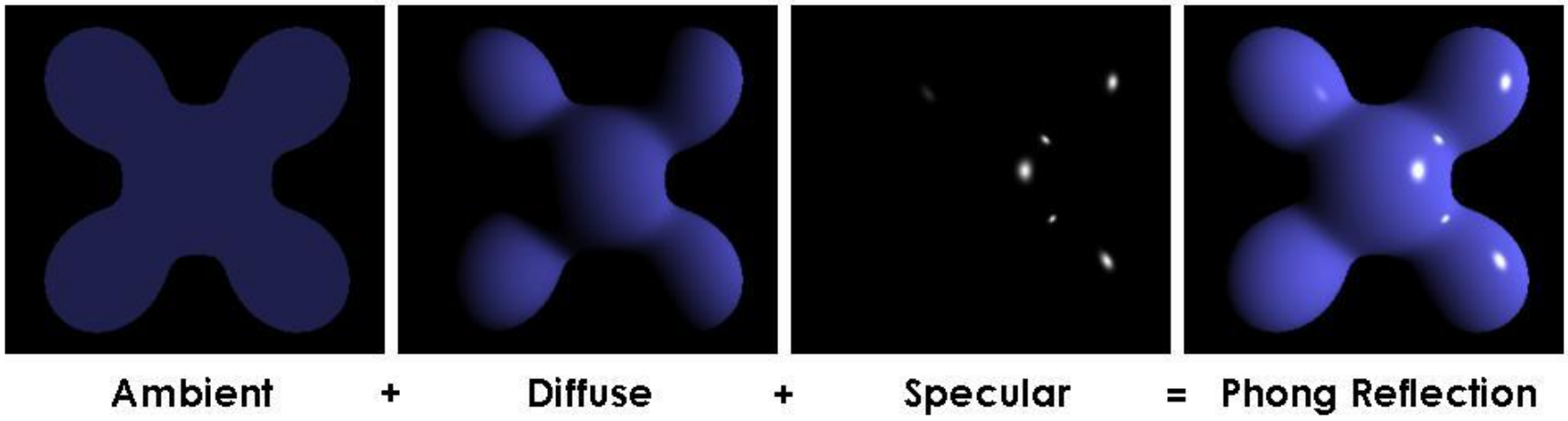
- natężenie światła otoczenia (**ambient**), które jest stałe i równe I_a (zakłada się, że jest ono rozproszone i bezkierunkowe oraz, że na skutek wielokrotnych odbić pada ono jednakowo pod wszystkimi kierunkami na rozpatrywane powierzchnie)
- natężenie światła rozproszonego I_d (**diffuse**), wyznaczane na podstawie modelu Lamberta
- natężenie światła odbijanego zwierciadlanie (**specular**): $I_s = I_P f(\beta)\cos^n(\alpha)$, gdzie maksimum natężenia światła odbitego zwierciadlanie występuje dla zerowego kąta α, natomiast wykładnik n we wzorze charakteryzuje właściwości odbiciowe danego materiału.

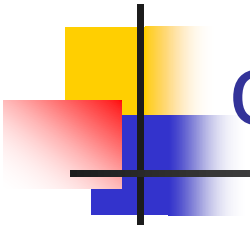
Każdy z tych składników może zostać przemnożony przez wartość współczynnika z przedziału [0,1], aby ustalić procentowy wpływ składowych na natężenie wynikowe.

model Phonga:

W praktyce okazało się, że dobre rezultaty można uzyskać dla współczynnika f_{att} postaci $f_{att} = 1/(c+r)$, gdzie c jest pewną stałą. Zauważmy, że im większa wartość wykładnika n, tym bardziej powierzchnia zbliża się do powierzchni lustrzanej (tym lepsze właściwości kierunkowe charakteryzują odbicie od tej powierzchni).

Przykład:



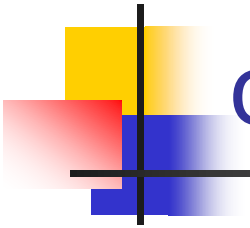


oświetlenie globalne:

Modelowanie lokalnego odbicia (lub przenikania) światła uwzględnia tylko lokalne właściwości powierzchni.

Problem oświetlenia globalnego jest natomiast opisem zależności związanych z rozchodzeniem się światła, ale uwzględniającym wzajemne oddziaływanie między powierzchniami, np. wielokrotne odbicie światła między różnymi przedmiotami.

Rozwiązanie tego problemu na poziomie modelu odbicia lokalnego sprowadza się do uwzględnienia średniej wartości oświetlenia we wszystkich punktach sceny – oświetlenia tła.
Taka składowa jest uwzględniona w modelu Phonga.

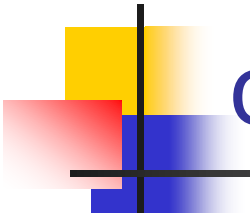


oświetlenie w OpenGL:

Model oświetlenia zastosowany w bibliotece OpenGL wykorzystuje trzy rodzaje światła:

- światło otaczające (ambient light), które nie pochodzi z żadnego konkretnego kierunku i równomiernie oświetla wszystkie elementy sceny
- światło rozproszone (diffuse light), które pada na obiekt z określonego kierunku, ale jest na nim rozpraszane we wszystkich kierunkach
- światło odbite (specular light), zwane także światłem kierunkowym, które pada na obiekt z określonego kierunku i odbijane jest także w ściśle określonym kierunku.

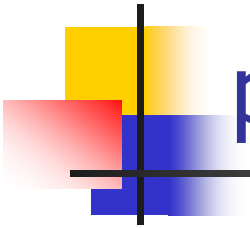
Ze światłem ściśle związane są pojęcia materiałów określające właściwości oświetlanych obiektów. Właściwości materiału, poza reakcją na opisane wyżej trzy rodzaje światła, uwzględniają także możliwość emitowania światła.



oświetlenie i źródła światła:

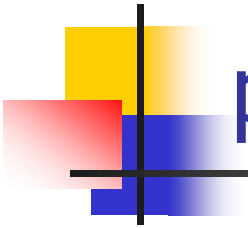
Domyślnie biblioteka OpenGL nie wykonuje żadnych obliczeń związanych z oświetleniem. Kolor piksela obiektu jest pobierany z funkcji typu glColor.
Uruchomienie obliczania oświetlenia wymaga wywołania funkcji **glEnable(GL_LIGHTING)**.
Ponadto zaleca się włączyć bufor głębokości.
Wyłączenie oświetlenia wymaga wywołania funkcji **glDisable(GL_LIGHTING)**.

Specyfikacja OpenGL określa, że minimalną ilość źródeł światła, którą musi obsługiwać każda implementacja biblioteki wynosi 8.
Każde źródło światła jest punktowe i ma swój unikatowy numer oznaczony jedną ze stałych: GL_LIGHT0, GL_LIGHT1, … , GL_LIGHT7.
Poszczególne źródła światła włączamy (wyłączamy) oddzielnie, niezależnie od innych, przy użyciu funkcji glEnable (glDisable) z parametrem określającym numer źródła światła.



parametry źródła światła:

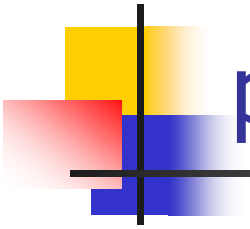
Funkcje:
void glLightf(GLenum light, GLenum pname, GLfloat param)
void glLighti(GLenum light, GLenum pname, GLint param)
gdzie light to numer źródła światła, którego parametr chcemy ustalić, pname określa parametr źródła światła, który chcemy ustalić, a param jest nową wartością parametru, pozwalają na modyfikację parametrów źródła światła określanych pojedynczą wartością.
Funkcje:
void glLightfv(GLenum light, GLenum pname,
const GLfloat *params)
void glLightiv(GLenum light, Glenum pname, const GLint *params)
gdzie light i pname określone są jak wyżej, a params jest tablicą nowych wartością parametru, pozwalają na modyfikację parametrów źródła światła określanych tablicą wartości.



parametry źródła światła:

Parametr pname może przyjąć następujące wartości:

- GL_AMBIENT - wartości składowych RGBA światła otaczającego, domyśnie 0.0, 0.0, 0.0, 1.0
- GL_DIFFUSE - wartości składowych RGBA światła idealnie rozpraszanego, domyśnie 1.0, 1.0. 1.0, 1.0 (GL_LIGHT0) i 0.0, 0.0, 0.0, 1.0 (pozostałe)
- GL_SPECULAR - wartości składowych RGBA światła odbitego kierunkowo, domyśnie 1.0, 1.0. 1.0, 1.0 (GL_LIGHT0) i 0.0, 0.0, 0.0, 1.0 (pozostałe)
- GL_POSITION – cztery współrzędne x, y, z, w, których interpretacja zależy od wartości w; jeżeli w = 1, oznacza to, że trzy pierwsze współrzędne określają położenie źródła światła; jeżeli w = 0, to źródło światła emituje światło kierunkowe, a jego promienie padają w kierunku zdefiniowanym przez trzy pierwsze współrzędne,
  domyślnie 0.0, 0.0, 1.0, 0.0



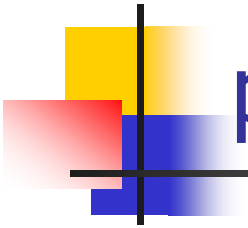
parametry źródła światła:

- GL_CONSTANT_ATTENUATION - stały współczynnik tłumienia światła k_c, domyśnie 1
- GL_LINEAR_ATTENUATION - liniowy współczynnik tłumienia światła k_l, domyśnie 0
- GL_QUADRATIC_ATTENUATION - kwadratowy współczynnik tłumienia światła k_q, domyślnie 0

Współczynnik osłabienia (tłumienia) światła jest obliczany ze wzoru:

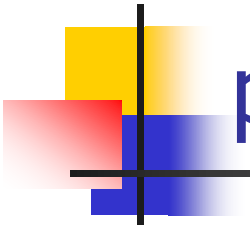
$$a = \frac{1}{k_c + k_l d + k_q d^2}$$

gdzie d jest odległością źródła światła od oświetlanego punktu.



parametry źródła światła:

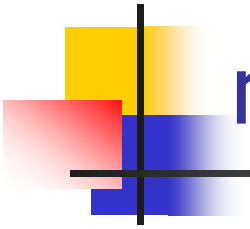
- GL_SPOT_DIRECTION - znormalizowany (długości 1) wektor o trzech składowych określających kierunek wysyłania światła (reflektora), domyślnie 0.0, 0.0, -1.0
- GL_SPOT_EXPONENT - wykładnik tłumienia kątowego reflektora; im wyższa potęga, tym bardziej skupiona jest wiązka światła; dopuszczalne są wartości z przedziału [0, 128], domyślnie 0
- GL_SPOT_CUTOFF - kąt (w stopniach) odcięcia reflektora; połowa kąta rozwarcia stożka światła; dopuszczalne są wartości z przedziału [0, 90] oraz 180, domyślnie 180



parametry źródła światła:

Przykład:

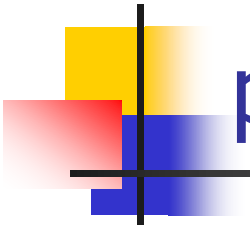
```
float P[4] = {1.0, 3.0, 5.0, 1.0}; //wektor położenia źródła światła
float E_a[4] = {1.0, 2.0, 3.0, 1.0}; //wektor ambient
float E_d[4] = {10.0, 70.0, 0.0, 1.0}; //wektor diffuse
float E_s[4] = {15.0, 30.0, 5.0, 1.0}; //wektor specular

glEnable(GL_LIGHT0); //włączenie źródła światła
glLightfv(GL_LIGHT0, GL_POSITION, P);
glLightfv(GL_LIGHT0, GL_AMBIENT, E_a);
glLightfv(GL_LIGHT0, GL_DIFFUSE, E_d);
glLightfv(GL_LIGHT0, GL_SPECULAR, E_s);
```



materiały w OpenGL:

Integralnym elementem modelu oświetlenia przyjętego w bibliotece OpenGL jest opis sposobu zachowania się powierzchni obiektów w reakcji na poszczególne rodzaje światła, czyli opis **właściwości materiałów**.

Właściwości materiału na ogół określa się podczas rysowania, bezpośrednio przed narysowaniem obiektu, albo nawet przed wyspecyfikowaniem każdego wierzchołka.

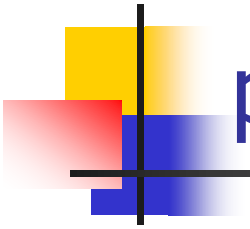


parametry materiału:

Funkcje:

void glMaterialf(GLenum face, GLenum pname, GLfloat param)
void glMateriali(GLenum face, GLenum pname, GLint param)
void glMaterialfv(GLenum face, GLenum pname,
const GLfloat *params)
void glMaterialiv(GLenum face, GLenum pname,
const GLint *params)

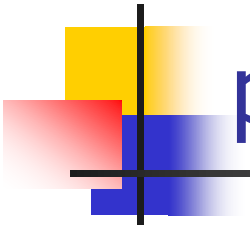
gdzie face ustala, której strony wielokąta dotyczy modyfikowany parametr (GL_FRONT - przednia strona wielokąta, GL_BACK - tylna strona wielokąta, GL_FRONT_AND_BACK - obie strony wielokąta), pname określa parametr materiału, który chcemy ustalić, a param (params) jest nową wartością (tablicą nowych wartości) parametru, pozwalają na modyfikację parametrów materiału określanych pojedynczą wartością (tablicą wartości).



parametry materiału:

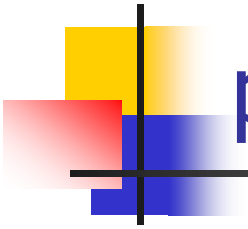
Parametr pname może przyjąć następujące wartości:

- GL_AMBIENT - składowe RGBA określające stopień (w postaci ułamka) odbicia światła otaczającego, domyślnie 0.2, 0.2, 0.2 1.0
- GL_DIFFUSE - składowe RGBA określające stopień (w postaci ułamka) rozproszenia światła rozproszonego, domyślnie 0.8, 0.8, 0.8, 1.0
- GL_AMBIENT_AND_DIFFUSE - składowe RGBA określające jednocześnie stopień odbicia światła otaczającego i stopień rozproszenia światła rozproszonego; jest to wartość domyślna
- GL_SPECULAR - składowe RGBA określające stopień (w postaci ułamka) odbicia światła odbitego, domyślnie 0.0, 0.0, 0.0, 1.0



parametry materiału:

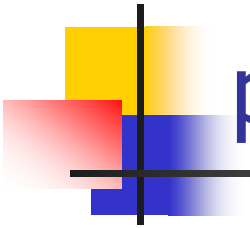
- GL_SHININESS - stała z przedziału [0, 128] określająca wykładnik n w modelu Phonga odbicia zwierciadlanego, im większa wartość parametru, tym lepsze odbicie zwierciadlane, domyślnie 0
- GL_EMISSION - składowe RGBA światła emitowanego przez obiekt; taki obiekt nie staje się źródłem światła i nie oświetla innych obiektów sceny, bowiem wymaga to utworzenia źródła światła, domyślnie 0.0, 0.0, 0.0, 1.0



parametry materiału:

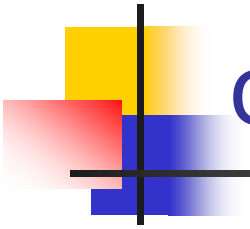
Przykład:

```
float m_a[4] = {1.0, 1.0, 1.0, 1.0}; //ambient materiału
float m_d[4] = {0.5, 1.0, 1.0, 1.0}; //diffuse materiału
float m_s[4] = {0.7, 0.5, 0.2, 1.0}; //specular materiału
int s = 20; //wykładnik potęgowy dla odbicia zwierciadlanego

glMaterialfv(GL_FRONT, GL_AMBIENT, m_a);
glMaterialfv(GL_FRONT, GL_DIFFUSE, m_d);
glMaterialfv(GL_FRONT, GL_SPECULAR, m_s);
glMateriali(GL_FRONT, GL_SHININESS, s);
```



parametry materiału:

Funkcja:
void glColorMaterial(GLenum face, GLenum mode)
gdzie face ustala, której strony wielokąta dotyczy modyfikowany parametr (GL_FRONT - przednia strona wielokąta, GL_BACK - tylna strona wielokąta, GL_FRONT_AND_BACK - obie strony wielokąta), a mode wskazuje, który parametr materiału (GL_EMISSION, GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR lub GL_AMBIENT_AND_DIFFUSE) ma być definiowany zgodnie z bieżącym kolorem wierzchołka, umożliwia definiowanie parametrów materiału na podstawie kolorów wierzchołków określonych wcześniej przy pomocy funkcji z grupy glColor (tzw. **śledzenie kolorów**). Śledzenie kolorów jest domyślne wyłączone, stąd przed użyciem funkcji glColorMaterial trzeba wywołać funkcję glEnable(GL_COLOR_MATERIAL). Wyłączenie śledzenia kolorów wymaga wywołania funkcji glDisable(GL_COLOR_MATERIAL).

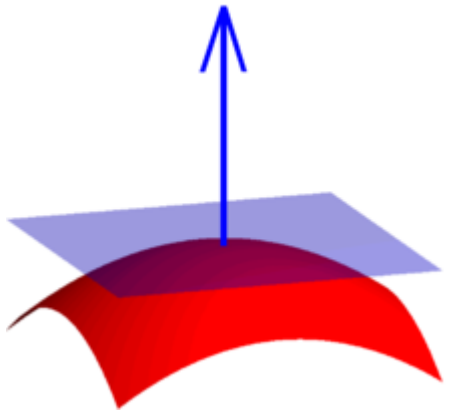


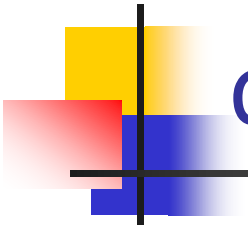
określenie stron wielokąta:

Funkcja:
void glFrontFace(GLenum mode)
gdzie mode przyjmuje wartości GL_CCW lub GL_CW, określa jako przednią strona wielokąta (GL_FRONT) tę, którą widzi obserwator, gdy wielokąt zbudowany jest z wierzchołków podanych w orientacji opisanej przez mode. Przeciwna (tylna) strona wielokąta określona jest stałą GL_BACK.

definiowanie wektora normalnego:

Funkcje:
void glNormal3d(GLdouble nx, GLdouble ny, GLdouble nz)
void glNormal3f(GLfloat nx, GLfloat ny, GLfloat nz)
void glNormal3i(GLint nx, GLint ny, GLint nz)
void glNormal3s(GLshort nx, GLshort ny, GLshort nz)
gdzie nx, ny, nz są współrzędnymi wektora, określają bieżącą wartość wektora normalnego.
Podobnie czynią to funkcje:
void glNormal3dv(const GLdouble * tab)
void glNormal3fv(const GLfloat * tab)
void glNormal3iv(const GLint * tab)
void glNormal3sv(const GLshort * tab)
gdzie tab jest tablicą trzech liczb, określających współrzędne wektora normalnego.

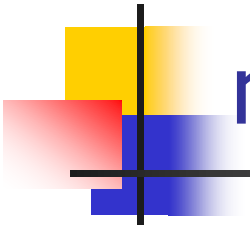




definiowanie wektora normalnego:

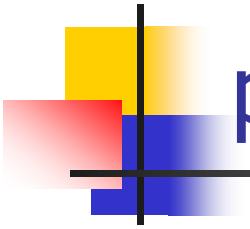
Aby obliczenia oświetlenia były wykonywane poprawne, wektor normalny musi mieć długość jednostkową (musi być znormalizowany).

Aby wymusić automatyczne normalizowanie wektorów normalnych, należy wywołać funkcję glEnable(GL_NORMALIZE). Domyślnie mechanizm ten jest nieaktywny.
Wyłączenie automatycznej normalizacji wektorów normalnych wymaga wywołania funkcji glDisable(GL_NORMALIZE).



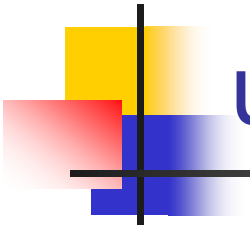
model oświetlenia:

Funkcje:
void glLightModelf(GLenum pname, GLfloat param)
void glLightModeli(GLenum pname, GLint param)
void glLightModelfv(GLenum pname, const GLfloat * params)
void glLightModeliv(GLenum pname, const GLint * params)
gdzie pname określa parametr modelu oświetlenia, który chcemy ustalić, a param (params) jest nową wartością (tablicą nowych wartości) parametru, pozwalają na modyfikację parametrów modelu oświetlenia określanych pojedynczą wartością (tablicą wartości).



parametry modelu oświetlenia:

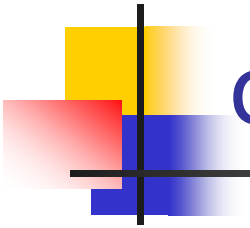
- GL_LIGHT_MODEL_LOCAL_VIEWER - sposób obliczania kąta odbicia światła odbitego (specular light); wartość 0 oznacza, że kąt odbicia światła obliczany jest na podstawie kierunku ujemnej osi OZ; wartość różna od 0 oznacza, że kąt obliczany jest na podstawie kierunku od początku układu współrzędnych do oświetlanego wierzchołka, domyślnie 0
- GL_LIGHT_MODEL_TWO_SIDE - określa, czy będą oświetlane obie strony wielokątów (wartość różna od 0), czy też tylko przednie strony wielokątów (wartość 0), domyślnie 0
- GL_LIGHT_MODEL_AMBIENT - określa składowe RGBA globalnego światła otaczającego (ambient light), domyślnie 0.2, 0.2, 0.2, 1.0



uwagi ogólne:

Wektory współrzędnych opisujących położenie źródeł światła lub kierunek osi reflektora poddawane są przekształceniu opisanemu przez bieżącą macierz na stosie GL_MODELVIEW. Zatem:

- aby położenie źródła światła było ustalone względem całej sceny, należy je określić po ustawieniu położenia obserwatora (czyli np. po wywołaniu funkcji gluLookAt)
- aby źródło światła było ustalone względem obserwatora, parametry położenia źródła światła należy określić po ustawieniu na wierzchołku stosu macierzy jednostkowej, przed wywołaniem funkcji gluLookAt
- aby związać źródło światła z dowolnym obiektem w scenie, trzeba położenie źródła światła określić po ustawieniu macierzy przekształcenia, która będzie ustawiona w czasie rysowania tego obiektu.



cieniowanie:

Cieniowanie (shading) to ustalanie barwy obiektów na podstawie światła odbitego. Wyznaczenie barwy związanej z modelem oświetlenia dla każdego punktu (piksela) jest zadaniem kosztownym.
Można zatem rozpatrywać cieniowanie (interpolację), które pozwoli wypełnić barwą wielokąty w sposób uproszczony.

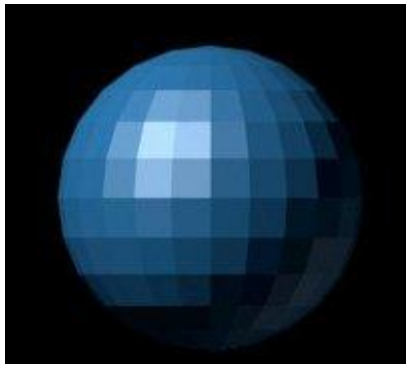
Stosuje się trzy warianty cieniowania:
- cieniowanie płaskie
- cieniowanie Gourouda
- cieniowanie Phonga.

cieniowanie płaskie:

Cieniowanie płaskie to cieniowanie stałą wartością, gdy cały wielokąt jest wypełniony taką samą barwą.

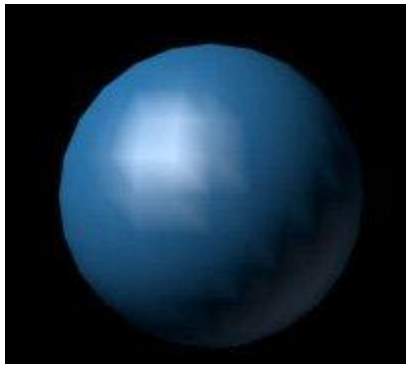
Cieniowanie płaskie jest zgodne z rzeczywistością, gdy obserwator lub źródło światła znajduje się w nieskończoności. Może być także stosowane, gdy wielokąt reprezentuje rzeczywiście powierzchnię modelowaną. Jeżeli złożony kształt powierzchni obiektu jest przybliżony wielościanem, to trzeba za pomocą cieniowania wygładzić obiekt niwelując wielościenny charakter.

Rozszerzeniem metody cieniowania płaskiego jest obliczanie barwy w kilku punktach wielokąta, a w pozostałych punktach barwa jest obliczana na drodze interpolacji.



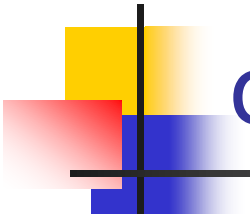
cieniowanie Gourouda:

Cieniowanie Gourouda wykorzystuje interpolowanie barwy.
Eliminuje nieciągłości barwy sąsiednich wielokątów.
W pierwszym etapie wyznaczamy (hipotetyczną) barwę w wierzchołkach wielościanu. W tym celu wyznaczamy hipotetyczny wektor normalny jako średnią arytmetyczną wektorów normalnych wszystkich ścian, do których ten wierzchołek należy. Następnie na podstawie wektora normalnego wyznaczamy barwę wierzchołka korzystając z wybranego modelu odbicia światła.
W drugim etapie dokonywana jest liniowa interpolacja barwy.



cieniowanie Phonga:

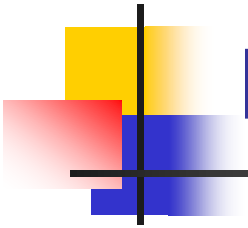
Cieniowanie Phonga wykorzystuje interpolację wektora normalnego. Eliminuje nieciągłości barwy sąsiednich wielokątów.
W pierwszym etapie wyznaczamy wektor normalny w wierzchołku w ten sam sposób, jak w cieniowaniu Gourauda.
W drugim etapie wyznaczamy interpolowany wektor normalny dla każdego piksela (punktu powierzchni odpowiadającego pikselowi).
Następnie wyznaczamy barwę piksela, na podstawie interpolowanego wektora normalnego, korzystając z wybranego modelu odbicia światła.
Cieniowanie Phonga daje lepsze rezultaty pod względem odwzorowania rozjaśnień obiektu (odbić zwierciadlanych). Ma jednak znacznie większą złożoność, bo konieczne jest obliczanie wektora normalnego dla każdego piksela.



cieniowanie w OpenGL:

Biblioteka OpenGL udostępnia standardowo dwa modele cieniowania: cieniowanie płaskie oraz cieniowanie gładkie.
W cieniowaniu płaskim wielokąt otrzymuje jeden kolor określony dla ostatniego wierzchołka (wyjątek stanowi prymityw GL_POLYGON, o kolorze którego decyduje kolor określony dla pierwszego wierzchołka).
Cieniowanie gładkie wykorzystuje algorytm Gourauda.

Funkcja:
void glShadeModel(GLenum mode)
gdzie mode przyjmuje jedną z wartości:
- GL_FLAT - cieniowanie płaskie
- GL_SMOOTH - cieniowanie gładkie,
określa wybór rodzaju cieniowania. Domyślnie stosowane jest cieniowanie gładkie.



Literatura pomocnicza:

http://wazniak.mimuw.edu.pl/index.php?title=Grafika_komputerowa_i_wizualizacja (kurs autorstwa Dariusza Sawickiego):
- Moduł 8: Modelowanie oświetlenia
- Moduł 9: Oświetlenie globalne
- Moduł 10: Dążenie do realizmu

Kurs OpenGL, C++
http://cpp0x.pl/kursy/Kurs-OpenGL-C++/101
OpenGL Programming Guide (Addison-Wesley Publishing Company)
http://neo.dmcs.pl/tgk/redbook.pdf
OpenGL Programming Guide
http://www.glprogramming.com/red/index.html
Wojciech Kowalewski: Wykłady z OpenGL – materiały dostępne na Contact.dir

c. d. n.