Światła i materiały

1. Krótkie podsumowanie wykładu:

   (a) Dla określenia siły oświetlenia danego punktu obiektu potrzebna jest informacja o lokalnej (w otoczeniu tego punktu) geometrii powierzchni, co wyraża się przez określenie w tym punkcie wektora prostopadłego do powierzchni obiektu (ściśle: prostopadłego do płaszczyzny stycznej do obiektu w tym punkcie). Wektor taki zwyczajowo nazywamy *wektorem normalnym*.

   (b) Specyfikację źródła światła określamy przez podanie kilku jego aspektów:

   - położenia w przestrzeni świata
   - emitowanej energii dla składowych RGB (w abstrakcyjnych jednostkach), przy czym energia jest podzielona na cztery części (każda z nich opisana składowymi RGB):
     - energia bezpośrednio padająca ze źródła, która będzie odbijana w sposób idealnie rozpraszający (wartość odbicia jest proporcjonalna do kosinusa kąta padania, tzn. do kąta pomiędzy wektorem światła i normalną w punkcie) - w OpenGL ten typ energii kodowany jest przez stałą `GL_DIFFUSE`,
     - energia bezpośrednio padająca ze źródła, która będzie odbijana w sposób kierunkowy (tylko w części kierunków odbicie jest niezerowe) - wartość odbicia jest maksymalna w kierunku *idealnego (lustrzanego) odbicia* i maleje wraz z odchyleniem od tego kierunku (szybkość zmniejszania się wartości odbicia jest określona przez pewien wykładnik potęgowy: im większy wykładnik tym węższy zakres kątów odbicia) - w OpenGL ten typ energii kodowany jest przez stałą `GL_SPECULAR`.
     - energii, która wysłana ze źródła dotrze do danego punktu niebezpośrednio (przez odbicia się od innych obiektów) - ponieważ w OpenGL nie ma możliwości śledzenia takich ścieżek, więc energię tą interpretujemy jako średnią ilość energii, która dotrze do każdego (czyli przeciętnego) punktu sceny, co oznacza, że każdy punkt otrzyma taką samą jej ilość - w OpenGL ten typ energii kodowany jest przez stałą `GL_AMBIENT`,
   - głównego kierunku transportu energii - określamy go opcjonalnie - domyślnie źródło światła wysyła energię we wszystkich kierunkach jednakowo - w OpenGL kierunek ten kodowany jest przez stałą `GL_SPOT_DIRECTION`
   - zakresu kątowego emisji światła względem kierunku głównego - w OpenGL odchylenie to jest kodowane przez stałą `GL_SPOT_CUTOFF`
   - szybkości wygaszania się energii w stożku światła określonym w dwóch poprzednich punktach - szybkość ta wyrażona jest przez potęgę kosinusa

kąta odchylenia od kierunku głównego - im wyższa potęga, tym bardziej skupiona wiązka światła - w OpenGL potęga ta kodowana jest przez stałą `GL_SPOT_EXPONENT`

- szybkości wygaszania się energii w miarę oddalania od źródła światła - może być ona stała (jednostajne wygaszanie wraz z odległością), wyrażona przez funkcję liniową lub wyrażona przez funkcję kwadratową - w OpenGL do jej opisu używamy wartości związanych z parametrami kodowanymi odpowiednio przez stałe: `GL_CONSTANT_ATTENUATION`, `GL_LINEAR_ATTENUATION`, `GL_QUADRATIC_ATTENUATION`

(c) Specyfikacja materiału definiuje przede wssystkim siłę reakcji punktu obiektu na energię padającą. Reakcja ta określona jest dla wszystkich typów energii padającej: DIFFUSE, SPECULAR, AMBIENT. Wyraża ona wartość mówiącą ile (procentowo w języku ułamków) danej energii zostanie odbite. Wspomniany w punkcie 1(b) wykładnik potęgowy dla odbicia energii SPECULAR kodowany jest przez stałą `GL_SHININESS` i jest traktowany jako własność materiału, a nie źródła swiatła. Ponadto można jeszcze materiałowi przypisać pewne właściwości emisyjne, które w OpenGL kodowane są przez stałą `GL_EMISSION`.

(d) *Prosty przykład*
Załóżmy, że źródło światła wysyła równą ilość energii we wszystkich kierunkach, przy czym energia DIFFUSE ma wartość $E^d = (E^d_R, E^d_G, E^d_B) = (100, 0, 200)$, energia SPECULAR ma wartość $E^s = (E^s_R, E^s_G, E^s_B) = (20, 30, 50)$, natomiast energia AMBIENT ma wartość $E^a = (E^a_R, E^a_G, E^a_B) = (0, 0, 30)$. Określamy reakcję pewnego punktu obiektu, do którego ta energia dociera. Materiał w tym punkcie odbija energię DIFFUSE w następujacy sposób: 20% energii składowej czerwonej, 100% energii składowej zielonej i 15% energii składowej niebieskiej, co wyrażamy przez wektor odbicia $m^d = (m^d_R, m^d_G, m^d_B) = (0.2, 1.0, 0.15)$. Podobnie określamy wektory odbicia dla energii SPECULAR i AMBIENT : $m^s = (m^s_R, m^s_G, m^s_B) = (1.0, 1.0, 1.0)$, $m^a = (m^a_R, m^a_G, m^a_B) = (0.5, 0.4, 1.0)$. W efekcie odbicia z punktu "wyjdzie" *energia odbita* podzielona na odbicie DIFFUSE $e^d = (e^d_R, e^d_B, e^d_G)$, , odbicie SPECULAR $e^s = (e^s_R, e^s_B, e^s_G)$ oraz odbicie AMBIENT $e^a = (e^a_R, e^a_B, e^a_G)$, przy czym

$$(e^d_R, e^d_B, e^d_G) = (m^d_R E^d_R, m^d_G E^d_G, m^d_B E^d_B) = (0.2{*}100, 1{*}0, 0.15{*}200) = (20, 0, 30),$$

$$(e^s_R, e^s_B, e^s_G) = (m^s_R E^s_R, m^s_G E^s_G, m^s_B E^s_B) = (1 * 20, 1 * 30, 1 * 50) = (20, 30, 50),$$

$$(e^a_R, e^a_B, e^a_G) = (m^a_R E^a_R, m^a_G E^a_G, m^a_B E^a_B) = (0.5 * 0, 0.4 * 0, 1 * 30) = (0, 0, 30).$$

2. *Orientacja powierzchni w OpenGL*

Każdy wielokąt ma dwie strony: przednią (względem obserwatora) i tylną. Funkcja

`glFrontFace(tryb)`

pozwala zdefiniować te pojęcia precyzyjnie. Argument `tryb` może przyjmować dwie wartości: `GL_CCW` (domyślna) i `GL_CW`. `GL_CCW` oznacza, że przednią stroną wielokąta będzie dla obserwatora ta, przed którą stojąc zobaczy wierzchołki definiujące wielokąt w orientacji przeciwnej do ruchu wskazówek zegara, tzn. poprawna definicja wielokąta przy domyślnej orientacji powinna być zrobiona przez wyobrażenie sobie, że stoimy naprzeciw przedniej strony ściany, którą chcemy zdefiniować i podanie listy wierzchołków przeciwnie do wskazówek ruchu zegara. Tak zdefniowana strona przednia jest identyfikowana przez stałą `GL_FRONT`. Przeciwna do niej strona tylna ma identyfikator `GL_BACK`. Jeżeli zmienimy domyślną orientację, to musimy też zmienić sposób generowania wielokątów.

Jeżeli wszystkie wielokąty w scenie mają jednakową orientację oraz wszystkie obiekty są bryłami, to w czasie rysowania sceny wielokąty, na które obserwator patrzy od strony tylnej są oczywiście zawsze zasłonięte przez pewne wielokąty obserwowane od przodu. W związku z tym i tak nie będą widoczne, gdy działa poprawny algorytm zasłaniania. Biblioteka OpenGL pozwala zrezygnować w ogóle z rysowania wielokątów tylnych (bądź przednich), przez wywołanie funkcji

`glCullFace(GL_BACK)`

(lub `glCullFace(GL_FRONT)`) oraz włączenie mechanizmu nierysowania określonych przez funkcję `glCullFace()` wielokątów:

`glEnable(GL_CULL_FACE).`

3. *Definiowanie wektora normalnego OpenGL*

   Bieżąca wartość wektora normalenego określona jest przez podanie trzech jego współrzędnych jako argumentów funkcji

   `glNormal3*(nx,ny,nz),`

   gdzie znak `*` przyjmuje wartości `f,d,s,i`, zależnie od typu argumentów. Istnieją też wektorowe wersje tej funkcji:

   `glNormal3*v(n),`

   gdzie `n` jest tablicą trzech liczb, oznaczających współrzędne wektora normalnego.

   Istotne jest (zob. definicje modeli odbicia), żeby długość wektora normalnego była zawsze równa 1. Biblioteka OpenGL jest w stanie sama o to dbać - wystarczy uruchomić ten mechanizm:

`glEnable(GL_NORMALIZE)`.

4. *Definiowanie źródeł światła w OpenGL*

   Do określenia parametrów źródła światła będacych wektorami i liczbami używamy odpowiednio funkcji:

   `glLight*v(nr, parametr, wartosc), glLight*(nr, parametr, wartosc),`

   gdzie `*` jest równe `i` lub `f`, zależnie od typu danego parametru. Argument `nr` przyjmuje wartość jednej z ośmiu stałych, określających numer źródła światła: `GL_LIGHT0, GL_LIGHT1, ... , GL_LIGHT7`. Argument `parametr` przyjmuje wartość stałej oznaczającej opisywany aspekt źródła światła, tak jak zostało to wyjaśnione w punkcie 1(b). Wreszcie argument `wartosc` jest wartością (wektorem lub liczbą) przypisywaną danemu aspektowi.

   Przykładowo: zdefiniujmy położenie zerowego źródła światła i przypiszmy mu energie DIFFUSE i SPECULAR oraz AMBIENT:

```
float P[4] = {1.0, 3.0, 5.0, 1.0};//wektor położenia źródła światła

float E_d[4] = {10.0, 70.0, 0.0, 1.0};//wektor energii diffuse

float E_s[4] = {15.0, 30.0, 5.0, 1.0};//wektor energii specular

float E_a[4] = {1.0, 2.0, 3.0, 1.0};//wektor energii ambient

glLightfv(GL_LIGHT0, GL_POSITION, P);

glLightfv(GL_LIGHT0, GL_DIFFUSE, E_d);

glLightfv(GL_LIGHT0, GL_SPECULAR, E_s);

glLightfv(GL_LIGHT0, GL_AMBIENT, E_a);
```

5. *Definiowanie właściwości materiałów w OpenGL*

   Do określenia parametrów materiału będacych wektorami i liczbami używamy odpowiednio funkcji:

```
glMaterial*v(strona, parametr, wartosc),

glMaterial*(strona, parametr, wartosc),
```

gdzie * jest równe i lub f, zależnie od typu danego parametru. Argument strona przyjmuje wartość jednej z trzech stałych, określających stronę powierzchni, do której stosuje się opis materiału: GL_FRONT, GL_BACK lub GL_FRONT_AND_BACK. Argument parametr przyjmuje wartość stałej oznaczającej daną właściwość materiału: reakcję na dany typ energii źródła światła - stałe GL_DIFFUSE, GL_SPECULAR, GL_AMBIENT, wykładnik potęgowy dla energii SPECULAR (zob. 1(b)) - stała GL_SHININESS oraz właściwości emisyjne materiału - stała GL_EMISSION. Wreszcie argument wartosc jest wartością (wektorem lub liczbą) danego parametru.

Przykładowo, określmy reakcję strony zewnętrznej obiektu o powierzchni z pewnego materiału na energie określone w przykładzie w punkcie 4.

```
float m_d[4] = {0.5, 1.0, 1.0, 1.0};//reakcja na  energię diffuse

float m_s[4] = {0.7, 0.5, 0.2, 1.0};//reakcja na  energię specular

float m_a[4] = {1.0, 1.0, 1.0, 1.0};//reakcja na energię ambient

int s = 20;//współczynnik potęgowy dla energii specular

glMaterialfv(GL_FRONT, GL_DIFFUSE, m_d);

glMaterialfv(GL_FRONT, GL_SPECULAR, m_s);

glMateriali(GL_FRONT, GL_SHININESS, s);

glLightfv(GL_LIGHT0, GL_AMBIENT, m_a);
```

6. *Pewne ogólne ustawienia*

Kilka aspektów procesu opisanego w poprzednich punktach można doprecyzować. Służą do tego funkcje

```
glLightModel*v(parametr, wartosc),   glLightModel*(parametr, wartosc),
```

gdzie * jest równe i lub f, zależnie od typu danego parametru. Argument wartosc specyfikuje wartość parametru określonego argumentem parametr, który może przyjąć postać jednej z trzech stałych

- `GL_LIGHT_MODEL_LOCAL_VIEWER` - wartość 0 dla tego parametru oznacza, że odbicia energii SPECULAR są równoległe i mierzone używając jako wektora obserwatora wektora [0,0,-1], zamiast rzeczywistego wektora obserwatora (wartość 1),
- `GL_LIGHT_MODEL_TWO_SIDE` - wartość 0 tego parametru oznacza, że liczone jest tylko odbicie dla strony przedniej powierzchni, wartość 1 powoduje obliczenia dla obu stron, przy czym wektor normalny jest automatycznie odwracany dla strony tylnej,
- `GL_LIGHT_MODEL_AMBIENT` - wartością tego parametru jest wektor oznaczający *globalną energię AMBIENT*, tzn. jakby średnią z energii AMBIENT wyspecyfikowanych przy każdym ze źródeł punktowych. Oczywiście nie ma żadnego sensu fizycznego jednoczesne specyfikowanie energii AMBIENT globalnie i dla źródeł punktowych.

7. *Włączenie mechanizmu obliczania świateł i materiałów*

   Domyślnie biblioteka OpenGL nie używa świateł i materiałów pomimo, że niektóre ich domyślne wartości są niezerowe. Oznacza to, że kolor obiektu widoczego w danym pikselu obrazu jest pobierany z funkcji typu `glColor*()`. Umożliwienie obliczania oświetlenia wymaga dwóch czynności

   (a) włączenia mechanizmu obliczania oświetlenia - `glEnable(GL_LIGHTING)` - w efekcie przestają działać wywołania typu `glColor*()`

   (b) włączenia każdego źródła punktowego oddzielnie - `glEnable(GL_LIGHT0)`, `glEnable(GL_LIGHT1)` itd.

   W przypadku, gdy żadne źródło punktowe nie jest włączone tylko *globalna energia AMBIENT* jest uwzględniana.

   Kolor obiektu widocznego w danym pikselu otrzymujemy przez proporcjonalne przeskalowanie do zakresu [0,1] sumy wartości odbitych w tym punkcie energii (zob. 1(d)). Oczywiście uwzględniona w pikselu wartość odbitej energii SPECULAR zależy od położenia obserwatora, gdyż obraz (czyli piksele) jest obliczany dla jego konkretnego położenia.

8. *Funkcja* `glColorMaterial()`

   W sytuacji, gdy wprowadzamy światła do sceny, w której są już wyspecyfikowane kolory przez funkcje typu `glColor*()` możemy użyć określonych w nich współrzędnych RGB jako wektorów reakcji na dany typ energii (zob. np. 1(d)). W tym celu musimy wywołać funkcję

`glColorMaterial(strona, energia),`

gdzie argument `strona` pełni taką samą rolę jak w funkcji typu `glMaterial*()`, natomiast argument `energia` przyjmuje jedną z wartości: `GL_EMISSION`, `GL_AMBIENT`, `GL_DIFFUSE`, `GL_SPECULAR`, `GL_AMBIENT_AND_DIFFUSE`.

Przykładowo wywołanie `glColorMaterial(GL_FRONT, GL_DIFFUSE)` oznacza, że wektor m^d (zob. 1(d)) jest tworzony z bieżącej wartości koloru i oczywiście specyfikacja tego aspektu materiału przez funkcję typu `glMaterial*()` przestaje mieć znaczenie (o ile w ogóle istnieje).

Plik `ogl6.cpp` zawiera kod jedenastu funkcji rysujących proste sceny i demonstrujących różne aspekty oświetlenia i materiałów. Zawarte są w nich wszystkie wystarczające komentarze.

9. Wartości domyślne

   - `GL_POSITION`: (0,0,1,0) - jest to źródło w nieskończoności (czwarta współrzędna równa 0), czyli wszystkie promienie światła są równoległe i w tym przypadku ich kierunek jest określony przez wektor [0,0,-1],
   - `GL_AMBIENT`:
     - światło globalne: (0.2,0.2,0.2,1),
     - światło: (0,0,0,1),
     - materiał: (0.2,0.2,0.2,1.0)
   - `GL_DIFFUSE`
     - światło: (1,1,1,1),
     - materiał: (0.8,0.8,0.8,1.0)
   - `GL_SPECULAR`
     - światło: (1,1,1,1),
     - materiał: (0,0,0,1.0)
   - `GL_SPOT_DIRECTION`: (0,0,-1),
   - `GL_SPOT_EXPONENT`: 0,
   - `GL_SPOT_CUTOFF`: 180,
   - `GL_CONSTANT_ATTENUATION`: 1 - brak stałego wygaszania z odległością
   - `GL_LINEAR_ATTENUATION`: 1 - brak liniowego wygaszania z odległością
   - `GL_QUADRATIC_ATTENUATION`: 1 - brak kwadratowego wygaszania z odległością
   - `GL_EMISSION`: (0,0,0,1),
   - `GL_SHININESS`: 0,

- `GL_LIGHT_MODEL_LOCAL_VIEWER`: 0,
- `GL_LIGHT_MODEL_TWO_SIDE`: 0,
- `GL_LIGHT_MODEL_AMBIENT`: (0.2,0.2,0.2,1.0),

10. *Zadanie*

    W scenie zawierającej pokój (zob. zadania w pliku `opengl2.pdf`) dodać dwa obiekty: lampę na suficie i lampkę na stole. Co najmniej w przypadku lampki na stole określić stożek oświetlenia dla źródła. Określić właściwości materiałów dla ścian, krzeseł, stołu i lamp adekwatne do danego obiektu.