Ćwiczenie 5

OpenGL - oświetlanie scen 3-D

1. Cel ćwiczenia

Celem ćwiczenia jest ilustracja możliwości oświetlania obiektów na scenach 3D z wykorzystaniem biblioteki *OpenGL* z rozszerzeniem *GLUT*. Przykłady jakie podano w opisie ćwiczenia ilustrują jak można opisać własności materiału z którego jest wykonany oświetlany obiekt, jak na scenie zdefiniować źródło światła i jak dobrać jego parametry. Pokazano także przykład pozwalający poznać sposób wyznaczania wektorów normalnych do punktów na powierzchni opisanej przy pomocy równań parametrycznych. Na zakończenie postawione zostanie bardziej złożone zadanie polegające na realizacji sterowania położeniem dwóch barwnych źródeł światła oświetlających nieruchomy obiekt.

2. Wprowadzenie na scenę 3D źródła światła

W ćwiczeniu 4 został napisany program wyświetlający obraz modelu czajnika do herbaty (teapot) i dodatkowo pozwajajacy na obracanie modelu przy pomocy ruchów myszy. Pierwszy przykład niniejszego ćwiczenia ma za zadanie pokazać, jak będzie wyglądał obraz modelu czajnika po wprowadzeniu na scenę źródła światła. W tym celu należy program z ćwiczenia 4 zmodyfikować przez wprowadzenie dwóch zmian:

- W miejsce funkcji `glutWireTeapot()` definiujęcej model szkieletowy należy umieścić funkcję `glutSolidTeapot()`, która opisuje model czajnika zbudowany z wypełnionych wieloboków.
- W funkcji `MyInit()` dodać nowy fragment kodu w postaci:

```
/*************************************************************************************/
//  Definicja materiału z jakiego zrobiony jest czajnik
//  i definicja źródła światła
/*************************************************************************************/

/*************************************************************************************/
// Definicja materiału z jakiego zrobiony jest czajnik

    GLfloat mat_ambient[]  = {1.0, 1.0, 1.0, 1.0};
    // współczynniki ka =[kar,kag,kab] dla światła otoczenia

    GLfloat mat_diffuse[]  = {1.0, 1.0, 1.0, 1.0};
    // współczynniki kd =[kdr,kdg,kdb] światła rozproszonego

    GLfloat mat_specular[] = {1.0, 1.0, 1.0, 1.0};
    // współczynniki ks =[ksr,ksg,ksb] dla światła odbitego

    GLfloat mat_shininess  = {20.0};
    // współczynnik n opisujący połysk powierzchni

/*************************************************************************************/
// Definicja źródła światła

     GLfloat light_position[] = {0.0, 0.0, 10.0, 1.0};
    // położenie źródła

    GLfloat light_ambient[] = {0.1, 0.1, 0.1, 1.0};
    // składowe intensywności świecenia źródła światła otoczenia
    // Ia = [Iar,Iag,Iab]

    GLfloat light_diffuse[] = {1.0, 1.0, 1.0, 1.0};
    // składowe intensywności świecenia źródła światła powodującego
    // odbicie dyfuzyjne Id = [Idr,Idg,Idb]

    GLfloat light_specular[]= {1.0, 1.0, 1.0, 1.0};
    // składowe intensywności świecenia źródła światła powodującego
    // odbicie kierunkowe Is = [Isr,Isg,Isb]

    GLfloat att_constant  = {1.0};
    // składowa stała ds dla modelu zmian oświetlenia w funkcji
    // odległości od źródła

    GLfloat att_linear    = {0.05};
    // składowa liniowa dl dla modelu zmian oświetlenia w funkcji
    // odległości od źródła

    GLfloat att_quadratic  = {0.001};
    // składowa kwadratowa dq dla modelu zmian oświetlenia w funkcji
    // odległości od źródła

/*************************************************************************************/

// Ustawienie parametrów materiału i źródła światła
/*************************************************************************************/
// Ustawienie patrametrów materiału

    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
    glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);

/*************************************************************************************/
// Ustawienie parametrów źródła

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

```
    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);

    glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, att_constant);

    glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, att_linear);
    glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, att_quadratic);

/*****************************************************************************/
// Ustawienie opcji systemu oświetlania sceny

    glShadeModel(GL_SMOOTH); // właczenie łagodnego cieniowania
    glEnable(GL_LIGHTING);   // właczenie systemu oświetlenia sceny
    glEnable(GL_LIGHT0);     // włączenie źródła o numerze 0
    glEnable(GL_DEPTH_TEST); // włączenie mechanizmu z-bufora

/*****************************************************************************/
```

Efekt działania tak zmodyfikowanego programu pokazany został na rysunku 1.

Rys. 1. Oświetlony białym światłem czajnik do herbaty wykonany z białego materiału

Przy generacji obrazu obiektu do obliczeń oświetlenia mechanizm renderujący *OpenGL* wykorzystuje model oświetlenia Phonga. Model ten (omówiony obszerniej na wykładzie) służy do obliczania oświetlenia punktu leżącego na powierzchni obiektu 3D dla dość ogólnego przypadku, zarówno w sensie oświetlenia, jak i charakterystyki powierzchni obiektu. Model pozwala na połączenie własności rozpraszania światła i odbicia kierunkowego oraz na intuicyjne powiązanie zależności otrzymywanego efektu oświetlenia z wartościami liczbowymi parametrów.

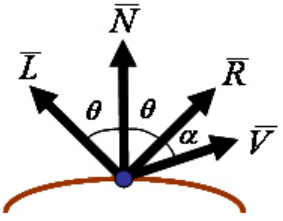
Model Phonga zadany jest wzorami pozwalającymi na obliczenie trzech składowych (R, G, B) intensywności oświetlenia analizowanego punktu powierzchni obiektu.

$$I_R = k_{aR} \cdot I_{aR} + \frac{1}{\left(a + bd_l + cd_l^2\right)}\left(k_{dR} \cdot I_{dR} \cdot \left(\overline{N} \cdot \overline{L}\right) + k_{sR} \cdot I_{sR}\left(\overline{R} \cdot \overline{V}\right)^n\right)$$

$$I_G = k_{aG} \cdot I_{aG} + \frac{1}{\left(a + bd_l + cd_l^2\right)}\left(k_{dG} \cdot I_{dG} \cdot \left(\overline{N} \cdot \overline{L}\right) + k_{sG} \cdot I_{sG}\left(\overline{R} \cdot \overline{V}\right)^n\right)$$

$$I_B = k_{aB} \cdot I_{aB} + \frac{1}{\left(a + bd_l + cd_l^2\right)}\left(k_{dB} \cdot I_{dB} \cdot \left(\overline{N} \cdot \overline{L}\right) + k_{sB} \cdot I_{sB}\left(\overline{R} \cdot \overline{V}\right)^n\right)$$

Wielkości oznaczone literami ***k***, ***I***, ***a***, ***b***, ***c*** i ***n*** występują w wyżej podanym kodzie, przy czym indeksy *a*, *d*, i *s* dotyczą odpowiednio światła otoczenia (*ambient*), światła rozproszonego (*diffuse*) i światła kierunkowego (*specular)*. Symbole oznaczone dużymi literami ***N***, ***L***, ***R***, i ***V*** są wektorami jednostkowymi, a ich mnożenie oznacza operację iloczynu skalarnego. Wektor ***N*** jest wektorem normalnym do powierzchni w analizowanym punkcie. Wektor ***L*** wyznacza kierunek padania światła na oświetlany punkt. Wektor ***R*** określa kierunek odbicia promienia dla idealnego zwierciadła a wektor ***V*** kierunek obserwacji punktu. Interpretację wymienionych wektorów pokazano na rysunku 2.



Rys. 2. Układ wektorów dla modelu Phonga

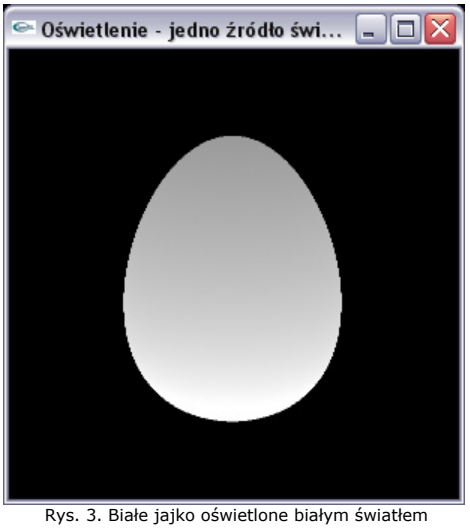
Jak można łatwo policzyć dla zdefiniowania materiału z jakiego wykonany jest obiekt i opisania jednego na razie źródła światła potrzeba aż 22 parametrów liczbowych i trzech wektorów (wektor ***R*** jest przekształconym wektorem ***L***). Parametry liczbowe zostały bezpośrednio podane w przykładowym kodzie, natomiast informacja o wektorach zawarta jest w opisie sceny w sposób pośredni i odpowiednio interpretowana przez system generacji efektów oświetlenia. Wektor ***L*** wynika z położenia źródła światła i oświetlonego punktu powierzchni. Współrzędne położenia źródła światła zostały jawnie wpisane w kodzie. Wektor ***N***, czyli wektor normalny do powierzchni w analizowanym punkcie na razie w kodzie nie występuje, a wektor opisujący kierunek obserwacji ***V*** wynika ze sposobu rzutowania, który zdefiniowano w w funkcji `ChangeSize()`.

Znajomość modelu Phonga pozwala na w miarę proste i intuicyjne manipulowanie parametrami materiału i źródła światła.

3. Oświetlenie własnego modelu obiektu 3-D

Kolejne zdanie polega na wygenerowaniu obrazu oświetlonego modelu jajka, który został opracowany w ćwiczeniu 3. W tym celu należy w poprzednim programie (nie zmieniając nic w opisie materiału i oświetlenia) zastąpić model czajnika czyli funkcję

`glutSolidTeapot()` modelem jajka. Należy wykorzystać model w postaci siatki trójkątów bez podawania dla poszczególnych wierzchołków informacji o kolorze. Obraz oświetlonego jajka wygląda tak jak to pokazane zostało na rysunku 3.



Rys. 3. Białe jajko oświetlone białym światłem

Łatwo zauważyć, że obraz nie wygląda tak dobrze jak otrzymany w poprzednim punkcie obraz czajnika. Przyczyną takiej sytuacji jest brak w modelu jajka pełnej informacji o geometrii obiektu potrzebnej systemowi oświetlenia. Wyjaśnienie otrzymanego rezultatu jest następujące. Model jajka, który został zbudowany w ćwiczeniu 3 składa się z trójkątów jakie powstały w wyniku triangulacji parametrycznej powierzchni opisanej równaniami analitycznymi. Model zawiera więc dane opisujące położenie w przestrzeni wierzchołków i sposobie ich połączenia w trójkąty. System oświetlenia, który bazuje na modelu Phonga potrzebuje jednak prócz współrzędnych oświetlanego punktu także informacji o wektorze normalnym do powierzchni. Takiej informacji w modelu jajka niestety nie ma. W modelu czajnika do herbaty pochodzącym z biblioteki *GLUT*, który został wykorzystany w pierwszym przykładzie, prócz punktów siatki i sposobie ich połączenia znajdują się jeszcze wektory normalne do powierzchni i dlatego obraz czajnika na rysunku 2, wygląda znacznie lepiej niż jajko z rysunku 3.

Poprawienie obrazu jajka wymaga więc dodania informacji o wektorach normalnych do powierzchni w punktach siatki stanowiącej model. Dotychczasowa konstrukcja opisująca model była mniej więcej taka:

```
 glBegin(GL_TRIANGLES);
...
glVertex3fv(...)
//współrzędne kolejnego wierzchołka trójkąta
...
glEnd();
```

Należy ją teraz uzupełnić wprowadzeniem informacji o wektorze normalnym w następujący sposób:

```
glBegin(GL_TRIANGLES);
...
glNormal3fv(...);
//wektor normalny dla następnego wierzchołka
glVertex3fv(...)
//współrzędne kolejnego wierzchołka trójkąta
...
glEnd();
```

Przepis jest więc dość prosty. Pojawia sie jednak pytanie. Skąd wziąć wektory normalne do powierzchni jajka dla leżących na niej punktów ?

4. Obliczanie wektora normalnego w punkcie leżącym na powierzchni opisanej parametrycznie

W ćwiczeniu 3 budowano model jajka na postawie równań parametrycznych powierzchni w postaci:

$$x(u,v) = (-90u^5 + 225u^4 - 270u^3 + 180u^2 - 45u)\cos(\pi v)$$
$$y(u,v) = 160u^4 - 320u^3 + 160u^2$$
$$z(u,v) = (-90u^5 + 225u^4 - 270u^3 + 180u^2 - 45u)\sin(\pi v)$$
$$0 \le u \le 1$$
$$0 \le v \le 1$$

Wektor normalny do punktu leżącego na tak opisanej powierzchni można znaleźć posługując się wzorami:

$$N(u,v) = \left[\begin{vmatrix} y_u & z_u \\ y_v & z_v \end{vmatrix}, \begin{vmatrix} z_u & x_u \\ z_v & x_v \end{vmatrix}, \begin{vmatrix} x_u & y_u \\ x_v & y_v \end{vmatrix}\right] =$$
$$= \left[y_u \cdot z_v - z_u y_v\ ,\quad z_u \cdot x_v - x_u z_v\ ,\quad x_u \cdot y_v - y_u x_v\right] \neq 0$$

przy czym

$$x_u = \frac{\partial x(u,v)}{\partial u}, \quad x_v = \frac{\partial x(u,v)}{\partial v}$$

$$y_u = \frac{\partial y(u,v)}{\partial u}, \quad y_v = \frac{\partial y(u,v)}{\partial v}$$

$$z_u = \frac{\partial z(u,v)}{\partial u}, \quad z_v = \frac{\partial z(u,v)}{\partial v}$$

Po wykonaniu operacji różniczkowania otrzymuje się wzory pozwalające na łatwe obliczenie wektora normalnego.

$$x_u = \frac{\partial x(u,v)}{\partial u} = \left(-450u^4 + 900u^3 - 810u^2 + 360u - 45\right) \cdot \cos(\pi v)$$

$$x_v = \frac{\partial x(u,v)}{\partial v} = \pi \cdot \left(90u^5 - 225u^4 + 270u^3 - 180u^2 + 45u\right) \cdot \sin(\pi v)$$

$$y_u = \frac{\partial y(u,v)}{\partial u} = 640u^3 - 960u^2 + 320u$$

$$y_v = \frac{\partial y(u,v)}{\partial v} = 0$$

$$z_u = \frac{\partial z(u,v)}{\partial u} = \left(-450u^4 + 900u^3 - 810u^2 + 360u - 45\right) \cdot \sin(\pi v)$$

$$z_v = \frac{\partial z(u,v)}{\partial v} = -\pi \cdot \left(90u^5 - 225u^4 + 270u^3 - 180u^2 + 45u\right) \cdot \cos(\pi v)$$

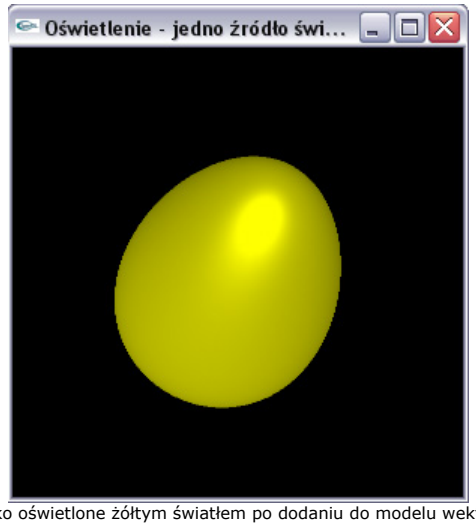
UWAGA: Po wyliczeniu wektorów normalnych z podanych wyżej wzorów należy je znormalizować, czyli doprowadzić do sytuacji, w której wektory będą miały długość 1. Normalizacja wektora polega na podzieleniu każdej składowej wektora przez jego długość. Długość wektora jest pierwiastkiem z sumy kwadratów jego składowych.

5. Zadania do wykonania

Zadanie 1. Należy zmodyfikować ostatni program wykonany w ćwiczeniu 2 (obracające się jajko) przez:

- Wprowadzenie na scenę jednego źródła światła (np. w kolorze żółtym).
- Zdefiniowane materiału z jakiego wykonane jest jajko (np. białe, połyskujące).
- Dodanie dla poszczególnych wierzchołków modelu jajka informacji o wektorach normalnych wyliczonych na podstawie zależności podanych w poprzednim punkcie.

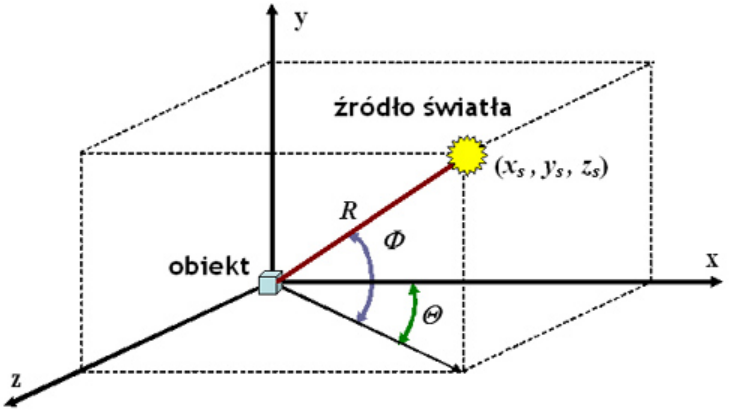
Rysunek jaki pojawi się na ekranie po wprowadzeniu tych zmian powinien wyglądać mniej więcej tak:



Rys. 4. Białe jajko oświetlone żółtym światłem po dodaniu do modelu wektorów normalnych

Zadanie 2. Zadanie polega na napisaniu programu, pozwalającego na oświetlanie modelu jajka przy pomocy dwóch źródeł barwnego światła i umożliwieniu manipulowania położeniem źródeł przy pomocy myszy. Wizualizacja modelu i sterowanie położeniem źródeł światła powinno odbywać się przy następujących założeniach:

- Jajko znajduje się jest w środku układu współrzędnych.
- Punkt, w którym umieszczone jest źródło może poruszać się po powierzchni sfery o promieniu ***R*** i środku leżącym w środku układu współrzędnych.
- Sterowanie położeniem źródła światła odbywać się (tak jak w ćwiczeniu 4) przy pomocy dwóch kątów. Pierwszy z nich określa kierunek świecenia na obiekt i nosi nazwę **azymutu** i oznaczany będzie jako ***Θ***. Drugi oznaczony przez ***Φ***, określa pośrednio wysokość położenia źródła światła nad hipotetycznym horyzontem i nazywa się kątem **elewacji**. Właściwy układ geometryczny został zilustrowany na rysunku 5.



Rys. 5. Obiekt, źródło światła i kąty azymutu oraz elewacji

Na podstawie rysunku 5 można dość łatwo wyznaczyć zależności wiążące położenie źródła światła z azymutem, kątem elewacji i promieniem sfery na powierzchni, której znajduje się obserwator. Są one następujące:

$$x_s(\Theta,\Phi) = R\cos(\Theta)\cos(\Phi)$$

$$y_s(\Theta,\Phi) = R\sin(\Phi)$$

$$z_s(\Theta,\Phi) = R\sin(\Theta)\cos(\Phi)$$

$$0 \le \Theta \le 2\pi$$

$$0 \le \Phi \le 2\pi$$

Ostatecznie zadanie brzmi tak:

- Po uruchomieniu programu, jajko powinno zostać oświetlone dwoma źródłami, jednym świecącym w kolorze niebieskim a drugim w żółtym, które znajdują się w położeniach początkowych (położenia te są dowolne)
- Przy wciśniętym lewym klawiszu myszy, ruch kursora myszy w kierunku poziomym powinien powodować proporcjonalną zmianę azymutu (kąta ***Θ***), natomiast ruch kursora myszy w kierunku pionowym zmianę kąta elewacji ***Φ*** dla pierwszego źródła.
- Przy wciśniętym prawym klawiszu myszy, ruch kursora myszy w kierunku poziomym powinien powodować proporcjonalną zmianę azymutu (kąta ***Θ***), natomiast ruch kursora w kierunku pionowym zmianę kąta elewacji ***Φ*** dla drugiego źródła.

Pozwala to na sterowanie położeniem źródeł światła na powierzchni sfer (promienie sfer dla poszczególnych źródeł mogą być oczywiście różne). Efekt działania tak napisanego programu pokazany jest na dwóch kolejnych rysunkach.

Rys. 6. Jajko oświetlone dwoma źródłami światła - przykład 1

Rys. 7. Jajko oświetlone dwoma źródłami światła - przykład 2

Ćwiczenie opracował: Jacek Jarnicki, korekta Marek Woda (2016-09-08)

Back