POLITECHNIKA WROCŁAWSKA
Instytut Informatyki, Automatyki i Robotyki
Zakład Systemów Komputerowych

Wprowadzenie do grafiki komputerowej

Kurs: INEK00012L

Sprawozdanie z ćwiczenia nr 5

TEMAT ĆWICZENIA: OpenGL – oświetlanie scen 3D

Wykonał:	**Marcel Guzik**
Termin:	**PN/P 7:30-10:00**
Data wykonania ćwiczenia:	**06-12-2021**
Data oddania sprawozdania:	**13-12-2021**
Ocena:	

Uwagi prowadzącego:

1 Wprowadzenie

Celem ćwiczenia jest ilustracja możliwości oświetlania obiektów na scenach 3D z wykorzystaniem biblioteki OpenGL z biblioteką GLUT. W trakcie realizacji ćwiczenia pokazano jak opisać własności materiału z którego jest wykonany oświetlany obiekt, jak na scenie zdefiniować źródło światła i jak dobrać jego parametry. Końcowa wersja programu realizuje sterowanie położeniem dwóch barwnych źródeł światła oświetlających nieruchomy obiekt, wraz z możliwością poruszania się obserwatora.

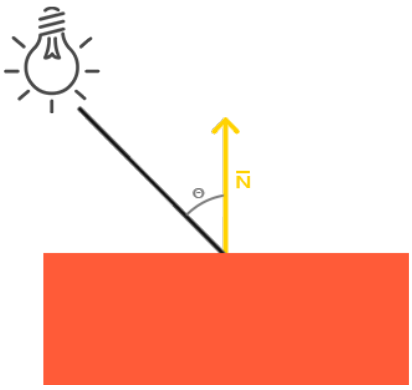
2 Realizacja oświetlenia

Do realizacji oświetlenia modelu wykonano dwie techniki, oświetlenie Phonga oraz cieniowanie Phonga.

2.1 Oświetlenie Phonga

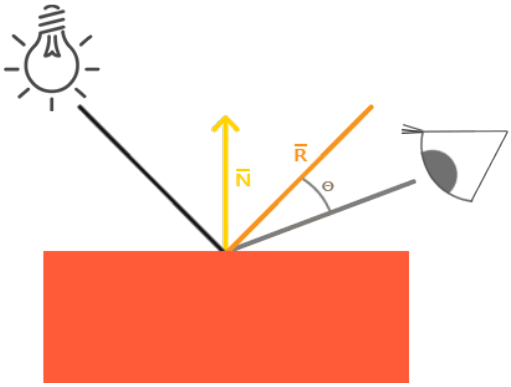
Oświetlenie Phonga jest modelem lokalnego oświetlenia punktów w przestrzeni trójwymiarowej. Według tego modelu, na oświetlenie punktu składa się kombinacja trzech różnych typów oświetlenia.

- **Oświetlenie ambientowe** - Reprezentuje tzw. światło otoczenia, czyli światło o niskiej intensywności, które, rozproszone po całej scenie, pozwala na dostrzeżenie konturów obiektu. Polega na naniesieniu na cały obiekt jednego, dosyć ciemnego koloru.
- **Oświetlenie rozproszone (dyfuzyjne)** - Reprezentuje światło które zostało odbite od obiektu i uległo rozproszeniu. Głównie z tego rodzaju światła składają się obiekty matowe.

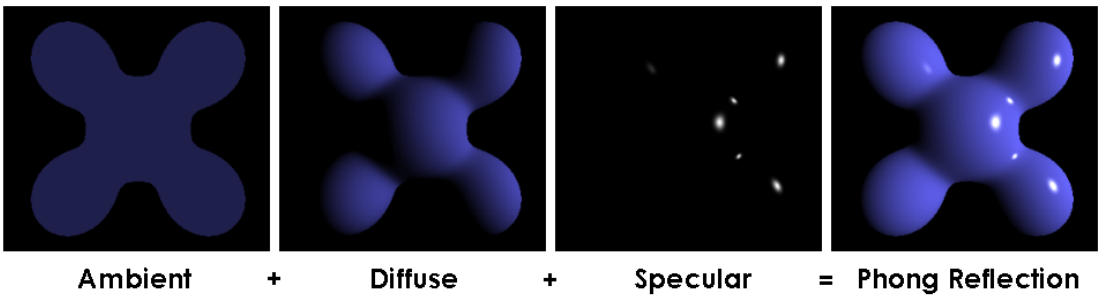


Rysunek 1: Oświetlenie rozproszone. Niezależnie od kąta pod którym obserwator patrzy na powierzchnię, jest ona oświetlona równomiernie.

- **Oświetlenie zwierciadlane (specular)** - Reprezentuje światło które uległo odbiciu zwierciadlanemu, tj. gdy kąt światła odbitego jest równy kątowi światła padającego. Głównie z tego rodzaju światła składają się obiekty błyszczące



Rysunek 2: Oświetlenie zwierciadlane. Kąt odbicia jest równy kątowi padania. Obserwator widzi różną intensywność światła w zależności od tego jak blisko odbitego promienia się znajduje.

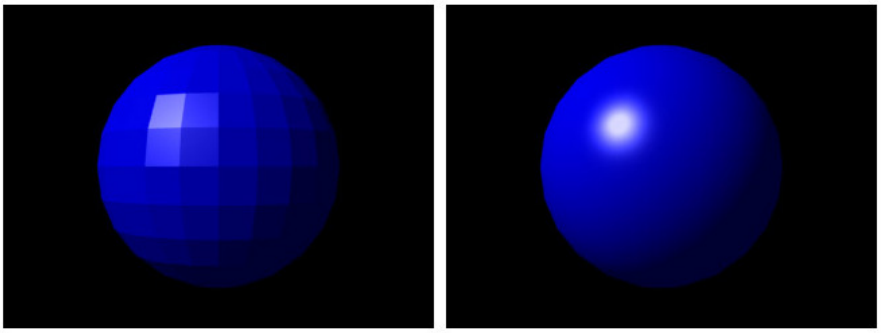


Rysunek 3: Przykład zastosowania modelu oświetlenia Phonga.

2.2 Cieniowanie Phonga

Cieniowanie Phonga jest techniką cieniowania polegającą na interpolacji wektora normalnego dla każdego rasteryzowanego fragmentu. Dzięki niemu uzyskujemy "gładkie" cieniowanie, pozbawione widocznych krawędzi i wierzchołków.

Cieniowanie Phonga bazuje na innej technice cieniowania zwanej cieniowaniem Gourauda. W tej technice dla wierzchołków modelu obliczany jest kolor zgodnie z modelem oświetlania Phonga, a następnie ten kolor jest interpolowany dla fragmentów pomiędzy wierzchołkami. Wada takiego modelu pojawia się gdy odbicie zwierciadlane pada na środek płaskiej powierzchni. Ponieważ odbicie występuje tylko na środku powierzchni, a nie przy jej wierzchołkach, odbicie znika. Cieniowanie Phonga eliminuje tą wadę przez to że nie interpoluje koloru, lecz wektory normalne, a dla każdego fragmentu kolor liczony jest z osobna.



Rysunek 4: Przykład cieniowania Phonga.

3 Realizacja programu

3.1 Konfiguracja oświetlenia

Oświetlenie jaja realizowane jest przez funkcje `glMaterial()` i `glLight()`. Poniższe fragmenty znajdują się w funkcji `init()` wywoływanej raz na początku programu, i służą do konfiguracji systemu oświetlenia.

```
// 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);
```

Funkcje z rodziny `glMaterial()` służą do ustawiania właściwości materiału z którego składa się obiekt. W powyższym fragmencie dla powierzchni które widzimy z przodu (czyli w wypadku naszego jaja, z zewnątrz) ustawiamy wartości reflektywności (czyli wartości kolorów w modelu RGBA) materiału dla każdego z trzech typów światła w modelu oświetlenia Phonga. Ponadto parametr `GL_SHININESS` określa jak dobrze materiał odbija światło. Im wyższa wartość tego parametu, tym mniejszy i bardziej "skoncentrowane" jest odbicie zwierciadlane.

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

float pos[4] = {0.0, 0.0, 0.0, 1.0};
angles_to_coords(light0Angles, pos);
glLightfv(GL_LIGHT0, GL_POSITION, pos);

glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, att_constant);
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, att_linear);
glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, att_quadratic);
```

Następnie ustawiamy wartości koloru światła oraz jego pozycję za pomocą funkcji `glLightfv()` oraz jej parametrów `GL_AMBIENT`, `GL_DIFFUSE`, `GL_SPECULAR` oraz `GL_POSITION`. Następnie parametry `GL_CONSTANT_ATTENUATION`, `GL_LINEAR_ATTENUATION` i `GL_QUADRATIC_ATTENUATION` służą do modyfikacji intensywności światła jeżeli jest to światło punktowe a nie kierunkowe.

Powyższy fragment odpowiada za skonfigurowanie pierwszego światła, czerwonego. Fragment ten powtórzony jest dla drugiego światła, które świeci na kolor niebieski.

```
glShadeModel(GL_SMOOTH); // właczenie łagodnego cieniowania
glEnable(GL_LIGHTING);   // włączenie oświetlenia
glEnable(GL_LIGHT0);     // włączenie źródła o numerze 0
glEnable(GL_LIGHT1);     // włączenie źródła o numerze 1
```

Na koniec wybieramy model gładkiego cieniowania oraz włączamy oświetlanie OpenGL oraz oba nasze światła.

3.2 Generowanie wektorów normalnych

Aby możliwe było oświetlenie powierzchni jaja, niezbędne było wyliczenie wektorów normalnych dla wszystkich jego wierzchołków. W tym celu przekształcono odpowiednio równania parametryczne użyte przy obliczaniu wierzchołków.

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

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

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

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

$$= \left[y_u \cdot z_v - z_u y_v \ , \quad z_u \cdot x_v - x_u z_v \ , \quad x_u \cdot y_v - y_u x_v \right] \neq 0$$

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

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

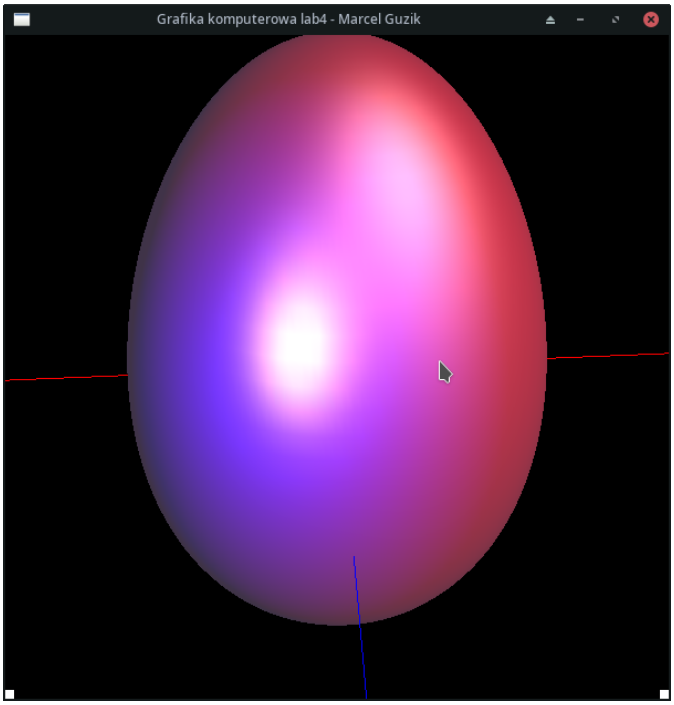
$$y_u = \frac{\partial y(u,v)}{\partial u} = 640u^3 - 960u^2 + 320u$$

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

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

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

3.3 Rezultat



4 Podsumowanie

Zadanie zaprezentowało możliwości realizacji oświetlenia i cieniowania w bibliotece OpenGL. Możliwe jest poruszanie źródłami światła oraz obserwatorem po powierzchni jaja, a także ich przybliżanie i oddalanie.