Grafika komputerowa i komunikacja człowiek-komputer

OpenGL – Oświetlanie scen 3-D

Autor: Luka Mitrović
Numer indeksu: 253907
Grupa: PT 16:25 TN
Prowadzący: dr inż. Jan Nikodem

Politechnika Wrocławska
Wrocław 26.01.2022

Wstęp teoretyczny

Model Phonga jest pozwala na obliczenia intensywności oświetlenia punktu powierzchni obiektu z użyciem trzech składowych R,G,B.

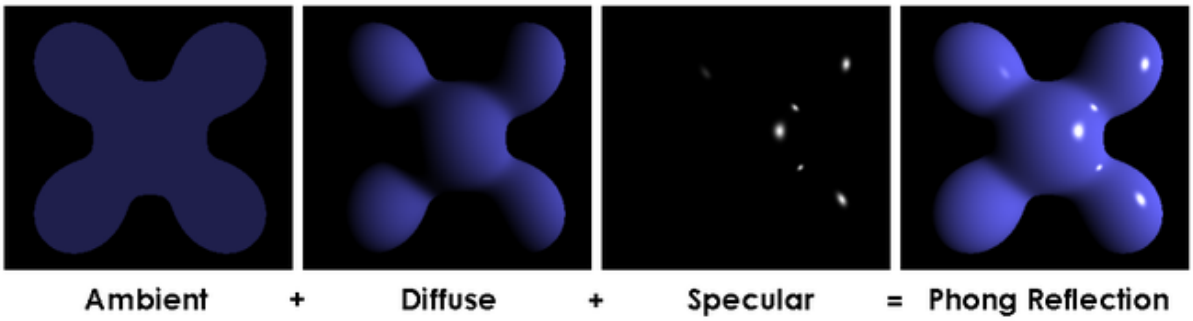
$$I_R = k_{aR} \cdot I_{aR} + \frac{1}{(a + bd_l + cd_l^2)}\left(k_{dR} \cdot I_{dR} \cdot (\overline{N} \cdot \overline{L}) + k_{sR} \cdot I_{sR} (\overline{R} \cdot \overline{V})^n\right)$$

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

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

Żeby uzyskać oświetlenia na scenie, należy zdefiniować zachowanie oświetlanego obiektu. Światło może być opisane na różne sposoby, ale w ramach laboratorium zastosowany został model Phonga, który pozwala jeszcze opisać światło trzema składowymi.

- światło otoczenia (ambient) - opisuje odbijanie światła otoczenia o stałym natężeniu.
- światło rozproszone (diffuse) - opisuje strumień światła wychodzący ze źródła i padający na obiekty.
- światło odbite (specular) - symuluje biały punkt na powierzchni połyskujących obiektów, co pozwala użytkownikowi na ustalenie miejsca padania światła na obiekt.



Obliczając parametry światła dla wektorów normalnych każdego punktu obiektu pozwala oświetlić go w sposób potrzebny użytkownikowi. Obliczanie wektorów normalnych jest wykonywane na podstawie wzorów przedstawionych poniżej.

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

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

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

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

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

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

Polecenia OpenGL używane podczas realizacji zadania:

- glMaterialfv(face, pname, *params) - funkcja pozwalająca na ustawienie parametrów materiału. Argument face przyjmuje jedną z dwóch wartości: front lub back. W programie wykorzystany jest zawsze z wartością front, co oznacza, że opisujemy płaszczyznę frontową figur prymitywnych. Pname to nazwa opisywanego parametru materiału, w programie wykorzystywane są następujące parametry: specular, ambient, diffuse, shininess.
- glLightf(light, pname, param) - funkcja ustawiająca parametry źródła światła. Light to wybrane źródło światła, pname to nazwa aktualizowanego parametru, a param to jego nowa wartość. Wykorzystane parametry to opisane wcześniej ambient, diffuse, specular oraz position (pozycja źródła światła), constant attentuation, linear attentuation, quadratic attentuation. 3 Ostatnie parametry są wykorzystywane w wypadku światła punktowego - pozwalają one na zmianę natężenia oświetlenia w zależności od położenia źródła światła względem obiektu.
- glShadeModel(mode) - włącza cieniowanie sceny. Parametr mode może przyjmować jedną z dwóch wartości: smooth lub flat. W programie wykorzystane jest cieniowanie w trybie smooth. Takie dokonuje cieniowania figury na podstawie interpolacji, co z reguły skutkuje przypisaniem różnych odcieni koloru do każdego piksela. Cieniowanie w trybie flat ustawia kolor całej figury na jednolity.
- glEnable(ligthning) - włącza oświetlenie sceny
- glEnable(light source) - włącza zadane źródło światła

Realizacja zadania

Wektory normalne są obliczane tylko raz, razem z wektorami opisującymi powierzchnie Beziera. Wektory zapisujemy do tablicy normalizedVector, co pozwala na szybki dostęp do ich wartości, kosztem większego wykorzystania pamięci, niż gdyby były obliczane przy każdym renderowaniu sceny.

```
v = float(j) / n;
arr[i][j][0] = (-90 * pow(u, 5) + 225 * pow(u, 4) - 270 * pow(u, 3) + 180 *
pow(u, 2) - 45 * u) * cos(M_PI * v);
arr[i][j][1] = (160 * pow(u, 4) - 320 * pow(u, 3) + 160 * pow(u, 2));
arr[i][j][2] = (-90 * pow(u, 5) + 225 * pow(u, 4) - 270 * pow(u, 3) + 180 *
pow(u, 2) - 45 * u) * sin(M_PI * v);

GLfloat Xu, Xv, Yu, Yv, Zu, Zv;
Xu = (-450*pow(u,4) + 900*pow(u,3) - 810*pow(u,2) + 360*u -45 )*cos(M_PI *
v);
Yu = 640*pow(u,3) - 960*pow(u,2) + 320*u;
Zu = (-450*pow(u,4) + 900*pow(u,3) - 810*pow(u,2) + 360*u -45 )*sin(M_PI *
v);
Xv = M_PI * (90*pow(u,5) - 225*pow(u,4) + 270*pow(u,3)- 180*pow(u,2) +
45*u) * sin(M_PI * v);
Yv = 0;
Zv = -M_PI * (90*pow(u,5) - 225*pow(u,4) + 270*pow(u,3)- 180*pow(u,2) +
45*u) * cos(M_PI * v);

GLfloat Nx, Ny, Nz;
Nx = Yu * Zv - Zu * Yv;
Ny = Zu * Xv - Xu * Zv;
```

```
Nz = Xu * Yv - Yu * Xv;

GLfloat vectorLength = sqrt(pow(Nx,2) + pow(Ny,2) + pow(Nz,2));
if(vectorLength == 0) vectorLength = 1;

normalizedVector[i][j][0] = Nx/vectorLength;
normalizedVector[i][j][1] = Ny/vectorLength;
normalizedVector[i][j][2] = Nz/vectorLength;
```

Żeby ustawić materiał rysowanego przedmiotu zostały zdefiniowane wektory mat ambient, mat diffuse, mat specular, mat shininess. Określają one pożądane wartości kolejnych parametrów.

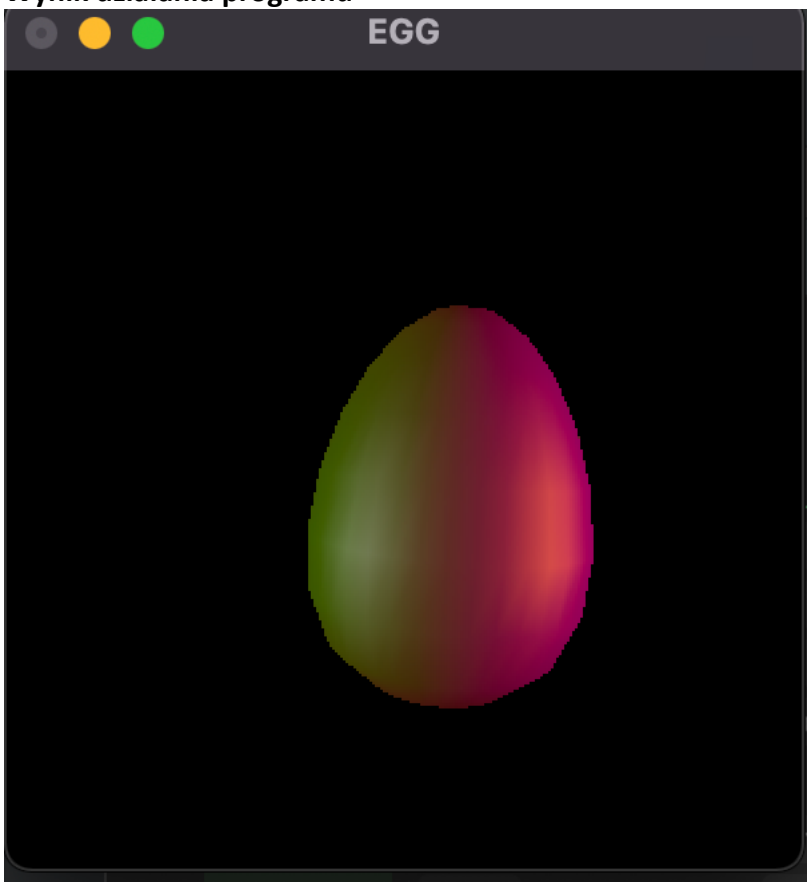
```
GLfloat mat_ambient[] = { 1.0f, 0.3f, 0.3f, 1.0f };
GLfloat mat_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat mat_shininess = { 20.0 };
//
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);
```

Żeby opisać światło najpierw definiowane są wektory określające wartości potrzebnych parametrów: light position, light ambient, light diffuse, light specular różne dla obu wykorzystywanych źródeł światła. Wartości att constant, att linear, att quadratic, potrzebne są do obliczania osłabienia światła.

```
GLfloat light_position[] = { 30.0, 0.0, 0.0, 1.0 };
GLfloat light_position1[] = { -30.0, 0.0, 0.0, 1.0 };

GLfloat light_ambient[] = { 0.1f, 0.0f, 0.0f, 0.25f };

GLfloat light_diffuse[] = { 1.0, 0.0, 1.0, 0.0 };
GLfloat light_diffuse1[] = { 0.0, 1.0, 0.0, 1.0 };

GLfloat light_specular[] = { 1.0f, 1.0f, 0.0f, 1.0f };
GLfloat light_specular1[] = { 0.7f, 0.7f, 1.0f, 1.0f };

GLfloat att_constant = { 1.0f };
GLfloat att_linear = { 0.05f };
GLfloat att_quadratic = { 0.001f };
```

```
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 parametrów światła
    glLightfv(GL_LIGHT1, GL_AMBIENT, light_ambient);
    glLightfv(GL_LIGHT1, GL_DIFFUSE, light_diffuse1);
    glLightfv(GL_LIGHT1, GL_SPECULAR, light_specular1);
    glLightfv(GL_LIGHT1, GL_POSITION, light_position1);
    glLightf(GL_LIGHT1, GL_CONSTANT_ATTENUATION, att_constant);
    glLightf(GL_LIGHT1, GL_LINEAR_ATTENUATION, att_linear);
    glLightf(GL_LIGHT1, GL_QUADRATIC_ATTENUATION, att_quadratic);
```

Wynik działania programu



Wnioski

Model Phonga pozwala na realistyczne oświetlenie sceny . W wersji programu nie udało dojść do efektu wymaganego w opisie zadania dokładnie, to może być spowodowane niepoprawnym dobraniem parametrów opisujących światło.