Grafika komputerowa i komunikacja człowiek komputer

Laboratorium 4

Oświetlanie scen 3-D

Wykonał:

Ivan Hancharyk 264511

Termin zajęć:

WT/TP/8:00

1. Cel ćwiczenia

Celem ćwiczenia jest ilustracja możliwości oświetlania obiektów na scenach 3D z wykorzystaniem biblioteki OpenGL.

2. Wstęp teoretyczny

W procesie generacji obrazu obiektów 3D w grafice komputerowej kluczowym aspektem jest prawidłowe obliczanie oświetlenia. Mechanizm renderujący OpenGL używa różnych modeli oświetlenia, a jednym z najbardziej powszechnych jest **model oświetlenia Phonga**. Jest to model oświetlenia punktowego, który pozwala na realistyczne symulowanie interakcji światła z powierzchnią obiektów w przestrzeni 3D. Dzięki temu możliwe jest uzyskanie efektów takich jak cienie, połyski czy refleksje.

Model Phonga oblicza oświetlenie na podstawie trzech składowych:

- **Ambient (światło otoczenia)** – jest to światło, które oświetla wszystkie obiekty w scenie w równym stopniu, niezależnie od ich położenia względem źródła światła. Odpowiada za podstawowy, stały poziom oświetlenia, który nie zależy od kąta padania światła.
- **Diffuse (światło rozproszone)** – jest to część światła, która pada na powierzchnię obiektu i zostaje rozproszona w różnych kierunkach. Oświetlenie rozproszone zależy od kąta padania światła na powierzchnię obiektu, co daje efekt cieniowania.
- **Specular (światło odbite)** – symuluje odbicie światła od połyskliwych powierzchni obiektów. To światło sprawia, że na powierzchni pojawia się wyraźny biały punkt refleksji, który umożliwia określenie, z jakiego kierunku pada światło.

Dodatkowo, w celu uzyskania realistycznych efektów, **shininess** (połysk) kontroluje stopień połysku powierzchni obiektu, czyli jak gładka i błyszcząca jest dana powierzchnia. Wyższa wartość shininess sprawia, że powierzchnia staje się bardziej lustrzana, co zwiększa wyrazistość odbicia światła.

Aby uzyskać pełną kontrolę nad wyglądem obiektu w scenie, OpenGL umożliwia modyfikację zarówno materiałów obiektów, jak i właściwości źródeł światła. Dzięki funkcjom takim jak **glMaterialf()** i **glMaterialfv()**, można ustawić parametry materiału obiektu, które obejmują powyższe składowe (ambient, diffuse, specular, shininess). Z kolei dla źródeł światła funkcje **glLightfv()** i **glLightf()** pozwalają na ustawienie takich właściwości jak pozycja, rodzaj światła, czy współczynniki tłumienia.

3. Realizacja zadania

Obliczanie wektorów normalnych

```
// obliczenie wektorów normalnych
Xu = (-450 * pow( x: u,  y: 4) + 900 * pow( x: u,  y: 3) - 810 * pow( x: u,  y: 2) + 360 * u - 45) * cos( X: M_PI * v);
Xv = M_PI * (90 * pow( x: u,  y: 5) - 225 * pow( x: u,  y: 4) + 270 * pow( x: u,  y: 3) - 180 * pow( x: u,  y: 2) + 45 * u) * sin( X: M_PI * v);
Yu = -5 + 640 * pow( x: u,  y: 3) - 960 * pow( x: u,  y: 2) + 320 * u;
Yv = -5;
Zu = (-450 * pow( x: u,  y: 4) + 900 * pow( x: u,  y: 3) - 810 * pow( x: u,  y: 2) + 360 * u - 45) * sin( X: M_PI * v);
Zv = -M_PI * (90 * pow( x: u,  y: 5) - 225 * pow( x: u,  y: 4) + 270 * pow( x: u,  y: 3) - 180 * pow( x: u,  y: 2) + 45 * u) * cos( X: M_PI * v);;

vectX = Yv * Zu - Yu * Zv;
vectY = Zv * Xu - Zu * Xv;
vectZ = Xv * Yu - Xu * Yv;

//normalizacja
float vectorLen = sqrt( X: pow( x: vectX,  y: 2) + pow( x: vectY,  y: 2) + pow( x: vectZ,  y: 2));
if (vectorLen == 0) vectorLen = 1;
vectors[i][j][0] = vectX / vectorLen;
vectors[i][j][1] = vectY / vectorLen;
vectors[i][j][2] = vectZ / vectorLen;

if (i < (N + 1) / 2) {
    vectors[i][j][0] *= -1;
    vectors[i][j][1] *= -1;
    vectors[i][j][2] *= -1;
}
```

Powyższy fragment kodu oblicza wektory normalne dla punktów na powierzchni 3D, która jest opisana parametrami u i v. Współrzędne punktów na powierzchni są obliczane na podstawie funkcji matematycznych, a następnie wyliczane są pochodne tych współrzędnych względem u i v (Xu, Xv, Yu, Yv, Zu, Zv). Na podstawie tych pochodnych obliczany jest wektor normalny, który jest wynikiem iloczynu wektorowego dwóch wektorów stycznych do powierzchni. Następnie wektor normalny jest normalizowany, co zapewnia, że ma jednostkową długość. W przypadku punktów znajdujących się w jednej z pierwszych połówek powierzchni, wektor normalny jest odwracany, aby zachować odpowiednią orientację.

Sterowanie ruchem źródła światła

```
// Handle light 0 (yellow)
if (statusLeft == 1) {
    azymuth_light0 -= delta_x * pix2angle / 10;
    elevation_light0 += delta_y * pix2angle / 10;

    if (azymuth_light0 <= -360.0)
        azymuth_light0 = 0.0;
    if (elevation_light0 >= 360.0)
        elevation_light0 = 0.0;

    // Update yellow light position based on the updated azimuth and elevation
    yellow_light_position[0] = rViewer * cos( x: azymuth_light0) * cos( x: elevation_light0);
    yellow_light_position[1] = rViewer * sin( x: elevation_light0);
    yellow_light_position[2] = rViewer * sin( x: azymuth_light0) * cos( x: elevation_light0);

    glLightfv( light: GL_LIGHT0,  pname: GL_POSITION,  params: yellow_light_position);
}
```

```
// Handle light 1 (blue)
if (statusRight == 1) {
    azymuth_light1 -= delta_x * pix2angle / 10;
    elevation_light1 += delta_y * pix2angle / 10;

    if (azymuth_light1 <= -360.0)
        azymuth_light1 = 0.0;
    if (elevation_light1 >= 360)
        elevation_light1 = 0.0;

    // Update blue light position based on the updated azimuth and elevation
    blue_light_position[0] = rViewer * cos( x: azymuth_light1) * cos( x: elevation_light1);
    blue_light_position[1] = rViewer * sin( x: elevation_light1);
    blue_light_position[2] = rViewer * sin( x: azymuth_light1) * cos( x: elevation_light1);

    glLightfv( light: GL_LIGHT1,  pname: GL_POSITION,  params: blue_light_position);
}
```

Kiedy lewy przycisk myszy jest wciśnięty (**statusLeft == 1**), pozycja żółtego światła (źródło **GL_LIGHT0**) jest aktualizowana na podstawie ruchów myszy w kierunku poziomym (**delta_x**) i pionowym (**delta_y**). Zmienne **azymuth_light0** i **elevation_light0** przechowują kąty azymutu i elewacji, które określają pozycję światła w przestrzeni. Po obliczeniu nowych kątów, współrzędne żółtego światła są aktualizowane, a funkcja **glLightfv()** ustawia nowe położenie tego światła w scenie.

Podobnie, gdy prawy przycisk myszy jest wciśnięty (**statusRight == 1**), pozycja niebieskiego światła (źródło **GL_LIGHT1**) jest aktualizowana w zależności od ruchu myszy. Analogicznie do żółtego światła, zmienne **azymuth_light1** i **elevation_light1** przechowują wartości kątów dla tego źródła. Kiedy kąt azymutu lub elewacji przekroczy określony zakres (np. -360 lub 360), jest on resetowany do wartości początkowej, co zapewnia płynność ruchu światła wokół obiektu. Po zaktualizowaniu pozycji, współrzędne niebieskiego światła są ustawiane za pomocą funkcji **glLightfv()**, co umożliwia dynamiczne dostosowywanie jego lokalizacji w przestrzeni 3D.

Definicja parametrów materiału i źródeł światła

```
GLfloat mat_ambient[] = { [0]: 1.0, [1]: 1.0, [2]: 1.0, [3]: 1.0 };
GLfloat mat_diffuse[] = { [0]: 1.0, [1]: 1.0, [2]: 1.0, [3]: 1.0 };
GLfloat mat_specular[] = { [0]: 1.0, [1]: 1.0, [2]: 1.0, [3]: 1.0 };
GLfloat mat_shininess = { 20.0 };

GLfloat light_ambient[] = { [0]: 0.1, [1]: 0.0, [2]: 0.0, [3]: 0.25 };
GLfloat yellow_light_diffuse[] = { [0]: 1.0, [1]: 1.0, [2]: 0.0, [3]: 0.0 };
GLfloat blue_light_diffuse[] = { [0]: 0.0, [1]: 0.0, [2]: 1.0, [3]: 1.0 };
GLfloat light_specular[] = { [0]: 0.5, [1]: 0.5, [2]: 0.5, [3]: 1.0 };
GLfloat att_constant = { 1.0 };
GLfloat att_linear = { 0.05 };
GLfloat att_quadratic = { 0.001 };
```

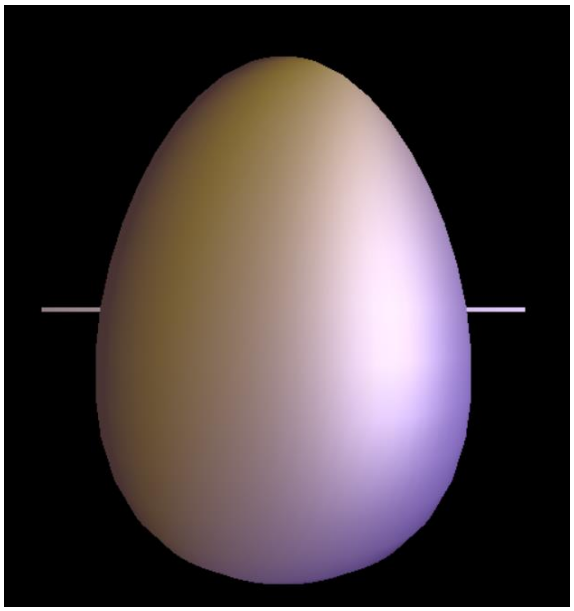
Ustawienie parametrów materiału i źródeł światła

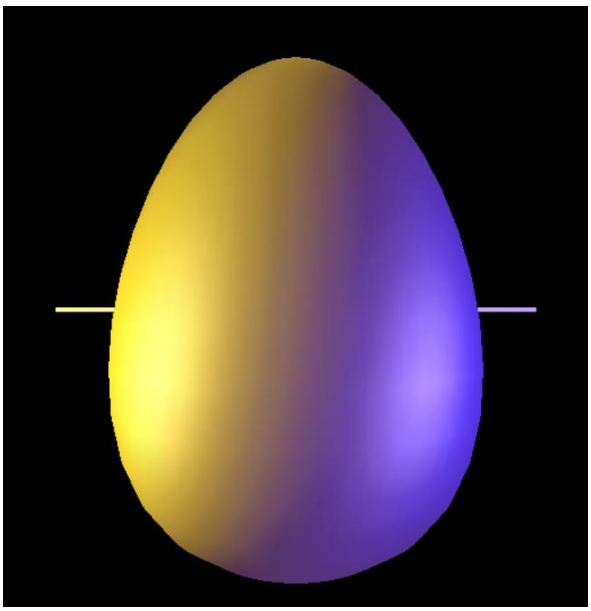
```
glMaterialfv( face: GL_FRONT, pname: GL_SPECULAR, params: mat_specular);
glMaterialfv( face: GL_FRONT, pname: GL_AMBIENT, params: mat_ambient);
glMaterialfv( face: GL_FRONT, pname: GL_DIFFUSE, params: mat_diffuse);
glMaterialf( face: GL_FRONT, pname: GL_SHININESS, param: mat_shininess);

glLightfv( light: GL_LIGHT0, pname: GL_AMBIENT, params: light_ambient);
glLightfv( light: GL_LIGHT0, pname: GL_DIFFUSE, params: yellow_light_diffuse);
glLightfv( light: GL_LIGHT0, pname: GL_SPECULAR, params: light_specular);
glLightfv( light: GL_LIGHT0, pname: GL_POSITION, params: yellow_light_position);
glLightf( light: GL_LIGHT0, pname: GL_CONSTANT_ATTENUATION, param: att_constant);
glLightf( light: GL_LIGHT0, pname: GL_LINEAR_ATTENUATION, param: att_linear);
glLightf( light: GL_LIGHT0, pname: GL_QUADRATIC_ATTENUATION, param: att_quadratic);

glLightfv( light: GL_LIGHT1, pname: GL_AMBIENT, params: light_ambient);
glLightfv( light: GL_LIGHT1, pname: GL_DIFFUSE, params: blue_light_diffuse);
glLightfv( light: GL_LIGHT1, pname: GL_SPECULAR, params: light_specular);
glLightfv( light: GL_LIGHT1, pname: GL_POSITION, params: blue_light_position);
glLightf( light: GL_LIGHT1, pname: GL_CONSTANT_ATTENUATION, param: att_constant);
glLightf( light: GL_LIGHT1, pname: GL_LINEAR_ATTENUATION, param: att_linear);
glLightf( light: GL_LIGHT1, pname: GL_QUADRATIC_ATTENUATION, param: att_quadratic);
```

Ustawienie opcji systemu oświetlania sceny

```
glShadeModel( mode: GL_SMOOTH);
glEnable( cap: GL_LIGHTING);
glEnable( cap: GL_LIGHT0);
glEnable( cap: GL_LIGHT1);
glEnable( cap: GL_DEPTH_TEST);
glClearColor( red: 0.0f, green: 0.0f, blue: 0.0f, alpha: 1.0f);
```

Przykłady działania:





Rysunek 1 i 2: początkowe położenia światła na starcie programu i przy zmienionym położeniu źródeł światła

4. Wnioski

Projekt umożliwia interaktywną manipulację oświetleniem oraz obiektami 3D przy użyciu bibliotek OpenGL i GLUT. Pozwolił on na zrozumienie podstaw techniki oświetlania obiektów w przestrzeni 3D oraz ze sposobem wyznaczania wektorów normalnych do punktów na powierzchni opisanej przy pomocy równań parametrycznych.