POLITECHNIKA WROCŁAWSKA
Instytut Informatyki, Automatyki i Robotyki
Zakład Systemów Komputerowych

Grafika komputerowa i komunikacja człowiek - komputer

Kurs: INEK00012L

Sprawozdanie z ćwiczenia nr 5

TEMAT ĆWICZENIA: OpenGL – Oświetlanie scen 3-D

Wykonał:	**Bartosz Szymczak, nr indeksu 252734**
Termin:	**Poniedziałek TN 7:30-10:30**
Data wykonania ćwiczenia:	**13.12.2021r.**
Data oddania sprawozdania:	
Ocena:	

Uwagi prowadzącego:

Spis treści

1. Cel ćwiczenia

Celem ćwiczenia jest ilustracja możliwości oświetlania obiektów na scenach 3D z wykorzystaniem biblioteki OpenGL z rozszerzeniem GLUT. Na przykładzie programu z instrukcji zostaną wykonane dwa zadania.

Pierwsze zadanie polega na oświetleniu jajka jednym, żółtym źródłem światła. W zadaniu drugim jajko powinno być oświetlanie przez dwa źródła światła, czerwone i niebieskie, którymi użytkownik będzie w stanie poruszać.

2. Wstęp teoretyczny

Biblioteki OpenGL z rozszerzeniem GLUT zawierają liczne funkcje, opisujące materiały z jakich zrobione są narysowane przedmioty, definiujące właściwości źródeł światła oraz ustawiające opcje systemu oświetlania sceny. Do prawidłowego oświetlenia przedmiotów wykorzystamy model oświetlenia Phonga, który pozwala na oświetlenie punktu leżącego na powierzchni 3D. Model Phonga składa się z układu wektorów jednostkowych. Wzory pozwalające na obliczenie wektora normalnego:

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

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

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

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

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

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

3. Opis programu

Zadanie 1

Znaczące różnice w porównaniu do poprzednich programów pojawiły się w metodach MyInit() oraz Egg().

W klasie Egg() wywoływana jest funkcja pomocnicza generateNormals(), która zgodnie z instrukcją oblicza wartości wektorów normalnych. Dalej w trakcie rysowania trójkątów, dodano informację o wektorach normalnych przed poinformowaniem o współrzędnych kolejnych wierzchołków trójkąta.

```
//Metoda generuje wektory normalne
void generateNormals() {
   float xu, xv, yu, yv = 0, zu, zv, uf, vf;
   float x, y, z, length;
   for (int u = 0; u < n; u++) {
      for (int v = 0; v < n; v++) {
         uf = (float)u / ((float)n - 1);
         vf = (float)v / ((float)n - 1);
         //Obliczenie wartości wektorów na podstawie gotowych wzorów
         xu = (-450 * pow(uf, 4) + 900 * pow(uf, 3) - 810 * pow(uf, 2) + (360 * uf) - 45) * cos(M_PI * vf);
         xv = M_PI * (90 * pow(uf, 5) - 225 * pow(uf, 4) + 270 * pow(uf, 3) - 180 * pow(uf, 2) + (45 * uf)) *
sin(M_PI * vf);
         yu = 640 * pow(uf, 3) - 960 * pow(uf, 2) + 320 * uf;
         zu = (-450 * pow(uf, 4) + 900 * pow(uf, 3) - 810 * pow(uf, 2) + (360 * uf) - 45) * sin(M_PI * vf);
         zv = -M_PI * (90 * pow(uf, 5) - 225 * pow(uf, 4) + 270 * pow(uf, 3) - 180 * pow(uf, 2) + (45 * uf)) *
cos(M_PI * vf);
         x = yu * zv - zu * yv;
         y = zu * xv - xu * zv;
         z = xu * yv - yu * xv;
         //Obliczenie długości wektora
         length = sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2));
         //Normalizacja wektorów dla pierwszej połowy jajka
         if (u < n / 2)
         {
            normals[u][v][0] = x / length;
            normals[u][v][1] = y / length;
            normals[u][v][2] = z / length;

         }
         //Normalizacja wektorów dla drugiej połowy jajka
         else if (u > n / 2)
         {
            normals[u][v][0] = -1 *x / length;
            normals[u][v][1] = -1 * y / length;
            normals[u][v][2] = -1 * z / length;
         }
         else if (u == 0 || u == n)
         {
            normals[u][v][0] = 0;
            normals[u][v][1] = -1;
            normals[u][v][2] = 0;
         }
         else
         {
            normals[u][v][0] = 0;
```

```
            normals[u][v][1] = 1;
            normals[u][v][2] = 0;
        }
    }
  }
}

//Funkcja rysująca jajko w podany sposób na podstawie wygenerowanych punktów
void Egg()
{
  //Generacja punktów
  eggPoints();
  //Generacja wektorów znormalizowanych
  generateNormals();
  //Model jajka stworzony z punktów połączonych w trójkąty
  for (int u = 0; u < n; ++u)
  {
    for (int v = 0; v < n; ++v)
    {
      //Zabezpieczenie przed wyjściem poza tablice
      int vdiff = 0, udiff = 0;
      if (v == n - 1) {
        vdiff = 1;
      }
      if (u == n - 1) {
        udiff = 1;
      }

      glBegin(GL_TRIANGLES);

      glNormal3fv(normals[u][v + 1 - n * vdiff]);
      glVertex3fv(points[u][(v + 1 - n * vdiff)]);

      glNormal3fv(normals[u + 1 - n * udiff][v]);
      glVertex3fv(points[(u + 1 - n * udiff)][v]);

      glNormal3fv(normals[u][v]);
      glVertex3fv(points[u][v]);

      glEnd();

      glBegin(GL_TRIANGLES);

      glNormal3fv(normals[u][v + 1 - n * vdiff]);
      glVertex3fv(points[u][v + 1 - n * vdiff]);

      glNormal3fv(normals[u + 1 - n * udiff][v]);
      glVertex3fv(points[(u + 1 - n * udiff)][v]);

      glNormal3fv(normals[u + 1 - n * udiff][v + 1 - n * vdiff]);
      glVertex3fv(points[(u + 1 - n * udiff)][(v + 1 - n * vdiff)]);

      glEnd();
    }
  }
}
```

Funkcja myInit() została rozszerzona o deklaracje różnych zmiennych odpowiedzialnych za definiowanie materiałów, szczegółów oświetlenia itd.

```
// Funkcja ustalająca stan renderowania
void MyInit(void)
{

   glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

   //Definicja materiału z jakiego zrobione jest jajko
   //Współczynniki dla światła otoczenia
   GLfloat mat_ambient[] = { 1.0,1.0, 1.0, 1 };
   //Współczynniki dla światła rozproszonego
   GLfloat mat_diffuse[] = { 1.0, 1.0, 1.0, 1 };
   //Współczynniki dla świata odbitego
   GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
   //Współczynnik opisujący połysk
   GLfloat mat_shininess = { 100.0 };
   //Definicja źródła światła
   //Położenie źródła światła
   GLfloat light_position[] = { 0.0, 0.0, 0.0, 1.0 };
   //Współczynniki intensywności światła otoczenia
   GLfloat light_ambient[] = { 0.1, 0.1, 0.1, 1.0 };
   //Współczynniki intensywności światła powodującego odbicie dyfuzyjne
   GLfloat light_diffuse[] = { 1.0, 1.0, 0.0, 1.0 };
   //Współczynniki intensywności światła powodującego odbicie kierunkowe
   GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };
   //Definicja stałej zmian oświetlenia w funkcji
   GLfloat att_constant = { 1.0 };
   //Definicja zmiennej liniowej zmian oświetlenia w funkcji
   GLfloat att_linear = { (GLfloat)0.05 };
   //Definicja zmiennej kwadratowej zmian oświetlenia w funkcji
   GLfloat att_quadratic = { (GLfloat)0.001 };

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

Zadanie 2

Zmiany w zadaniu 2 polegały na analogicznym dodaniu kolejnego źródła światła oraz dodaniu funkcjonalności poruszania źródłami światła tak jak poruszano wcześniej jajkiem.

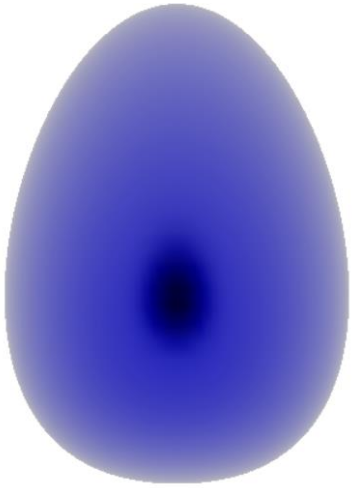
Reszta metod i funkcjonalności pozostały bez zmian w porównaniu z poprzednimi ćwiczeniami.

4. Wyniki

Zrzuty ekranu został odwrócone kolorystycznie w celu oszczędzenia tuszu.

```
W celu modyfiakcji polozenia obserwatora nalezy uzyc:
'w' i 's' - ruch wzdluz osi x
'a' i 'd' - ruch wzdluz osi y
```

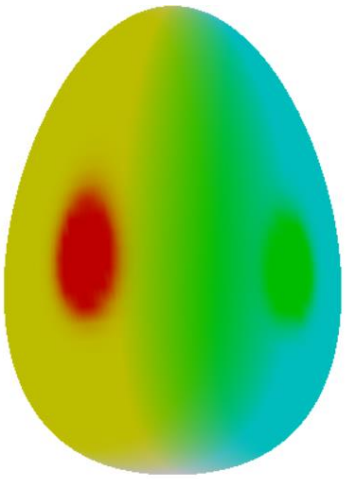
Rysunek 1: Wynik zadania pierwszego w oknie konsolowym



Rysunek 2: Wynik zadania pierwszego w oknie aplikacji

```
W celu modyfiakcji polozenia obserwatora nalezy uzyc:
'w' i 's' - ruch wzdluz osi x
'a' i 'd' - ruch wzdluz osi y
W celu modyfikacji polozenia zrodel swiatla nalezy uzyc:
LPM + ruch myszka - zmiana polozenia pierwszego zrodla swiatla
PPM + ruch myszka - zmiana polozenia drugiego zrodla swiatla
```

Rysunek 3: Wynik zadania drugiego w oknie konsolowym



Rysunek 4: Wynik zadania drugiego w oknie aplikacji

5. Wnioski

Dzięki dokumentacji oraz instrukcji do ćwiczenia, byłem w stanie stworzyć program ilustrujący oświetlenie oraz fakturę przedmiotu. Symulacja zwróciła oczekiwane efekty, program został wykonany poprawnie.