POLITECHNIKA WROCŁAWSKA
Instytut Informatyki, Automatyki i Robotyki
Zakład Systemów Komputerowych

Wprowadzenie do grafiki komputerowej

Kurs: INE4234L

Sprawozdanie z ćwiczenia nr 3

TEMAT ĆWICZENIA OpenGL - modelowanie obiektów 3-D

Wykonał:	**Bartłomiej Sawicki**
Termin:	**PN/P 7.30-10.30**
Data wykonania ćwiczenia:	**25.10.2021r.**
Data oddania sprawozdania:	**8.11.2021r.**
Ocena:	

Uwagi prowadzącego:

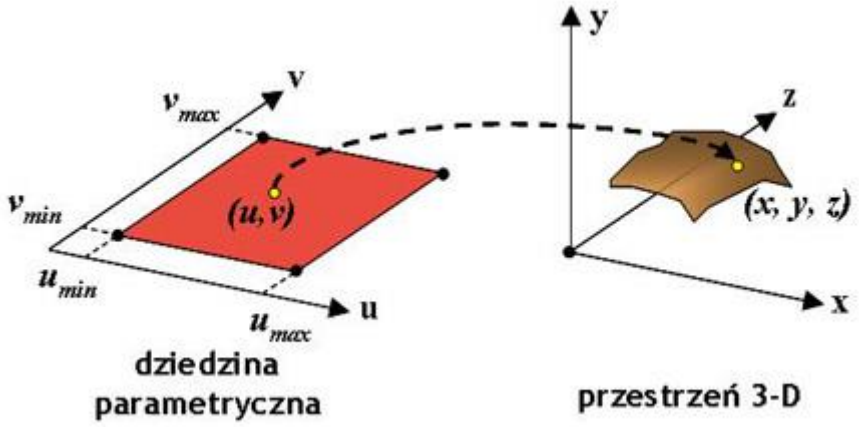
1. Wstęp teoretyczny

Zadanie polegało na rysowaniu modelu jajka w przestrzeni 3-D. Modelowany obiekt będzie rysowany z wykorzystaniem powierzchni opisanej równaniami parametrycznymi.

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

Przebieg generowania punktów:

1. Podanie liczby n która będzie informować na ile części trzeba podzielić kwadrat jednostkowy
2. Tablica kwadratowa o wymiarach n x n będzie przechowywać punkty.
3. Na kwadrat jednostkowy nakładamy siatkę n x n.
4. Dla każdego punku u, v obliczamy współrzędne punktów z równań parametrycznych.



2. Realizacja zadania

Szkielet programu bazuje na kodzie zawartym w instrukcji laboratoryjnej. Funkcja odpowiedzialna za rysowanie jajka jest wywoływana w procedurze `RenderScene`.

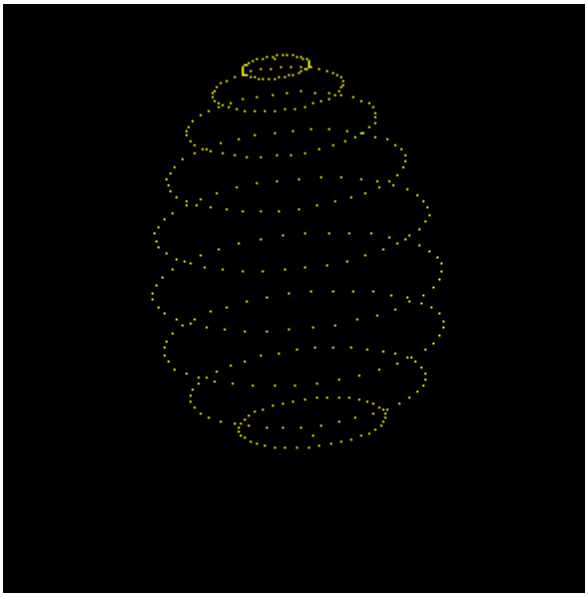
2.1. Fragment kodu odpowiedzialny za generowanie punktów płaszczyzny jajka.

```
// tablica punktow
point3 array[n][n];
float u, v;
//generowanie wspolrzednych punktow w 3D
for (int i = 0; i < n; i++)
{
    u = float(i) / n;
    for (int j = 0; j < n; j++)
    {
        v = float(j) / n;
        array[i][j][0] = (-90 * pow(u, 5) + 225 * pow(u, 4) - 270 * pow(u, 3) + 180 * pow(u, 2) - 45 * u) * cos(3.14 * v);
        array[i][j][1] = 160 * pow(u, 4) - 320 * pow(u, 3) + 160 * pow(u, 2) - 3.5f;
        array[i][j][2] = (-90 * pow(u, 5) + 225 * pow(u, 4) - 270 * pow(u, 3) + 180 * pow(u, 2) - 45 * u) * sin(3.14 * v);
    }
}
```

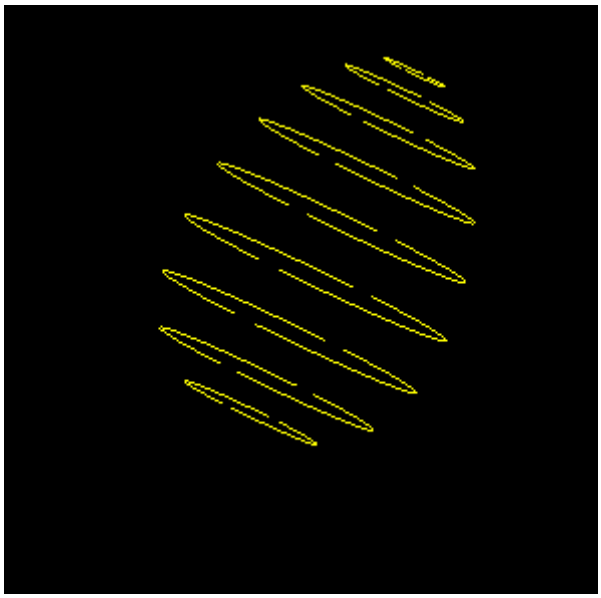
2.2. Rysowanie jajka może odbyć się na 3 różne sposoby – chmura punktów, siatka lub trójkąty

Rysowanie chmury punktów odbywa się przez GL_POINTS

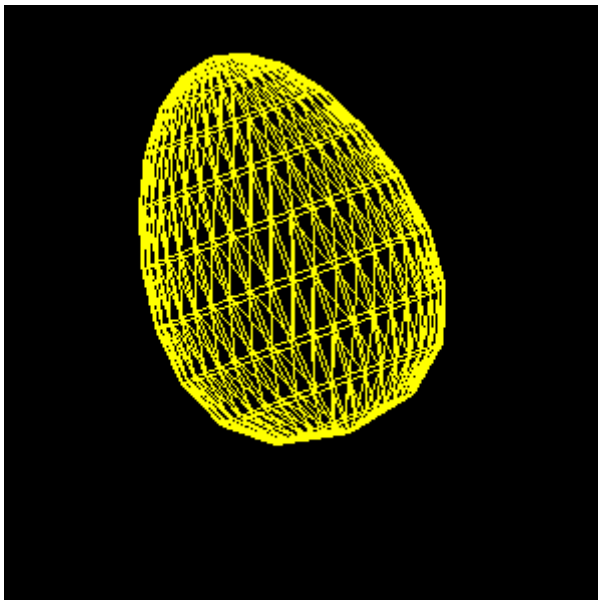
```
glColor3f(1.0f, 1.0f, 0);
glBegin(GL_POINTS);
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        glVertex3fv(array[i][j]);
    }
}
glEnd();
```



Rysowanie siatki było znacznie bardziej wymagające od rysowania chmury punktów. Siarkę uzyskałem przez łączenie sąsiadujących punktów. Problem pojawiał się gdy trzeba było połączyć pierwszy punkt na danym piętrze z ostatnim punktem na tym piętrze (tak aby uzyskać okrąg a nie dwa półokręgi). Podobną trudność napotkałem w przypadku rysowania linii skośnych. Aby połączyć w poprawny sposób te punkty należało obliczyć które punkty trzeba ze sobą łączyć. Jeżeli tablica array[][] przechowuje informacje o punktach z których zbudowane jest jajko i przyjmiemy, że ostatni punkt na danym poziomie znajduje się w tej tablicy pod indeksami i, n-1 (array[i][n-1]) to pierwszy punkt znajduje się pod indeksami (n-i)%n, 0 (array[(n-i)%n][0]). Poniższy rysunek przedstawia źle rysowane poziome linie siatki.

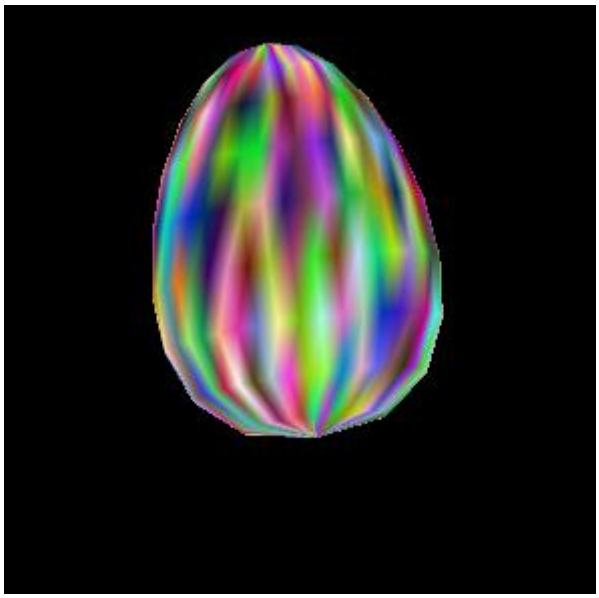


```
glColor3f(1.0f, 1.0f, 0);
glBegin(GL_LINES);
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        //linie pionowe
        glVertex3fv(array[i][j]);
        glVertex3fv(array[(i + 1) % n][j]);
        if (j + 1 < n)
        {
            //linie poziome
            glVertex3fv(array[i][j]);
            glVertex3fv(array[i][j + 1]);
        }
        else
        {
            //polczenie linii poziomch w pelny okrag
            glVertex3fv(array[i][j]);
            glVertex3fv(array[(n - i) % n][0]);
        }
        if (j + 1 < n)
        {
            //linie skosne
            glVertex3fv(array[i][j]);
            glVertex3fv(array[(i + 1) % n][j + 1]);
        }
        else
        {
            //lczenie linii skosnych
            glVertex3fv(array[i][j]);
            glVertex3fv(array[(n - (i + 1)) % n][0]);
        }
    }
}
glEnd();
```



Rysowanie trójkątów, z których składa się jajko nie było dużo trudniejsze od rysowania siatki. Polegało również na odpowiednim znajdowaniu punktów na końcu półokręgu. Aby uzyskać kolory na wierzchołkach trójkątów przygotowałem wcześniej tablicę colors[][] którą uzupełniłem losowymi kolorami.

```
glBegin(GL_TRIANGLES);
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        if (j + 1 < n)
        {
            glColor3fv(colors[i][j]);
            glVertex3fv(array[i][j]);
            glColor3fv(colors[(i + 1) % n][j]);
            glVertex3fv(array[(i + 1) % n][j]);
            glColor3fv(colors[(i + 1) % n][j + 1]);
            glVertex3fv(array[(i + 1) % n][j + 1]);
        }
        else
        {
            //laczenie trojkatow w osi poziomej
            glColor3fv(colors[i][j]);
            glVertex3fv(array[i][j]);
            glColor3fv(colors[(i + 1) % n][j]);
            glVertex3fv(array[(i + 1) % n][j]);
            glColor3fv(colors[(n - (i + 1)) % n][0]);
            glVertex3fv(array[(n - (i + 1)) % n][0]);
        }
        if (j + 1 < n)
        {
            glColor3fv(colors[i][j]);
            glVertex3fv(array[i][j]);
            glColor3fv(colors[i][j + 1]);
            glVertex3fv(array[i][j + 1]);
            glColor3fv(colors[(i + 1) % n][j + 1]);
            glVertex3fv(array[(i + 1) % n][j + 1]);
        }
        else
        {
            //laczenie trojkatow w osi poziomej
            glColor3fv(colors[i][j]);
            glVertex3fv(array[i][j]);
            glColor3fv(colors[(n - i) % n][0]);
            glVertex3fv(array[(n - i) % n][0]);
            glColor3fv(colors[(n - (i + 1)) % n][0]);
            glVertex3fv(array[(n - (i + 1)) % n][0]);
        }
    }
}
glEnd();
```



2.3. Obracanie wyświetlanych obiektów
Program umożliwia obracanie stworzonego obiektu w każdej z osi

Funkcja odpowiedzialna za ustawienie odpowiedniej wartości każdego z kątów

```
void spinEgg()
{
    theta[0] -= 0.5;
    if (theta[0] > 360.0) theta[0] -= 360.0;

    theta[1] -= 0.5;
    if (theta[1] > 360.0) theta[1] -= 360.0;

    theta[2] -= 0.5;
    if (theta[2] > 360.0) theta[2] -= 360.0;

    glutPostRedisplay(); //odświeżenie zawartości aktualnego okna
}
```

Gdzie theta to tablica zawierająca trzy elementy – kąty obrotu w każdej z osi.

```
static GLfloat theta[] = { 0.0, 0.0, 0.0 }; // trzy kąty obrotu
```

Funkcje RenderScene zmodyfikowano w taki sposób by na ekranie było rysowane jajko, które jest cały czas obracane w każdej z osi.

```
    glRotatef(theta[0], 1.0, 0.0, 0.0);

    glRotatef(theta[1], 0.0, 1.0, 0.0);

    glRotatef(theta[2], 0.0, 0.0, 1.0);

    Egg();

    glFlush();
```

Aby rotacja wykonywana była cały czas funkcję spinEgg należy wywołać w funkcji main polecenie `glutIdleFunc(spinEgg);`

2.4. Obsługa zdarzeń klawiatury

Program umożliwia zmianę sposobu rysowania jajka podczas poprzez wciśnięcie odpowiedniego klawisza na klawiaturze: p – chmura punktów, w – siatka, s – trójkąty.

Do obsługi zdarzeń klawiatury wykorzystałem funkcje zawartą w instrukcji.

```
void keys(unsigned char key, int x, int y)
{
    if (key == 'p') model = 1;
    if (key == 'w') model = 2;
    if (key == 's') model = 3;

    RenderScene(); // przerysowanie obrazu sceny
}
```

Funkcja ta jest wywoływana każdorazowo w procedurze main przez polecenie `glutKeyboardFunc(keys)`, które jest wywoływane przez naciśnięcie klawisza na klawiaturze.

3. Wnioski

Instrukcja laboratoryjna była głównym źródłem informacji o tworzeniu obiektów 3-D. Procedura `Egg` działa poprawnie. Funkcja rysuje model jajka w zależności od podanego przez użytkowna sposobu rysowania p – chmura punktów, w – siatka, s – trójkąty. Dzięki procedurze `spinEgg` jajko obraca się cały czas w każdej osi.