Grafika komputerowa i komunikacja człowiek komputer

Laboratorium 2

Modelowanie obiektów 3-D w OpenGL

Wykonał:

Ivan Hancharyk 264511

Termin zajęć:

WT/TP/8:00

1. Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z technikami modelowania i wizualizacji obiektów trójwymiarowych **(3D)**, z wykorzystaniem biblioteki **OpenGL** oraz **GLUT** oraz zbudowaniu 3D obiektów: jajka (w trzech różnych postaciach) oraz czajnika.

2. Wstęp teoretyczny

Podstawą realizacji ćwiczenia jest stworzenie modelu jajka w przestrzeni 3D. W projekcie wykorzystano **równania parametryczne**, aby opisać powierzchnię jajka. Parametry u i v są wykorzystywane do stworzenia równomiernej siatki punktów, a następnie do wyliczenia ich współrzędnych w przestrzeni trójwymiarowej.

Model jajka może być przedstawiany na kilka różnych sposobów, w zależności od trybu wybranego przez użytkownika:

- **Punkty**: W tej wersji jajko jest przedstawione jako zbiór oddzielnych punktów. Każdy punkt reprezentuje konkretną współrzędną obliczoną na podstawie równań parametrycznych.
- **Siatka**: Jajko może być również przedstawione jako siatka złożona z odcinków łączących sąsiednie punkty w poziomie i pionie.
- **Trójkąty**: Ta metoda wizualizacji nazywa się wypełnionymi trójkątami, ponieważ powierzchnia modelu jest utworzona z małych, połączonych trójkątów, co pozwala uzyskać gładki i realistyczny wygląd obiektu.

Ważnym elementem projektu jest wprowadzenie możliwości **interakcji użytkownika** z modelem. Użytkownik może za pomocą klawiszy wybrać sposób wizualizacji modelu (np. **p** dla punktów, **s** dla siatki, **t** dla trójkątów), a także aktywować lub dezaktywować automatyczną rotację modelu (spacja). Klawisze strzałek umożliwiają ręczną rotację modelu wokół osi X i Z, co pozwala użytkownikowi na dokładniejsze przyjrzenie się różnym częściom obiektu (WAŻNE: sterowanie klawiszami nie jest możliwe podczas działania rotacji automatycznej). W tym celu wykorzystywane są odpowiednie funkcje **GLUT**, takie jak *glutKeyboardFunc*() do obsługi klawiszy oraz *glutSpecialFunc*() do obsługi specjalnych klawiszy, jak strzałki. Skończyć działanie programu można po przez zamknięcia okna bądź naciśnięcie klawiszu **esc**.

Rotacja jajka jest realizowana za pomocą macierzy transformacji geometrycznych. Do wykonania obrotów wykorzystano funkcję *glRotatef*(), która modyfikuje położenie obiektu względem osi X, Y lub Z, co pozwala na stworzenie wrażenia ruchu. Automatyczna rotacja jest realizowana dzięki funkcji *glutIdleFunc*(), która umożliwia wykonywanie określonych operacji w momencie,

gdy program jest bezczynny. W tym przypadku, funkcja ta jest wykorzystywana do ciągłej modyfikacji kątów obrotu jajka, co powoduje jego animację.

3. Realizacja zadania

Algorytm definiowania punktów:

1. Zadanie liczby N, która będzie określała na ile przedziałów podzielony zostanie bok kwadratu jednostkowego dziedziny parametrycznej.

2. Deklaracja tablicy o rozmiarze NxN, która będzie służyła do zapisywania współrzednych punktów w przestrzeni 3-D.

```
Point3D points[N][N];
```

Każdy element tablicy zawierał będzie trzy liczby będące współrzędnymi x, y, z jednego punktu.

```
struct Point3D {
    GLfloat  x;
    GLfloat  y;
    GLfloat  z;
};
```

3. Nałożyć na kwadrat jednostkowy dziedziny parametrycznej równomierną siatkę NxN punktów.

4. Dla każdego z punktów u,v obliczyć współrzędne x,y,z na podstawie równań parametrycznych i zapisać do wcześniej stworzonej tablicy.

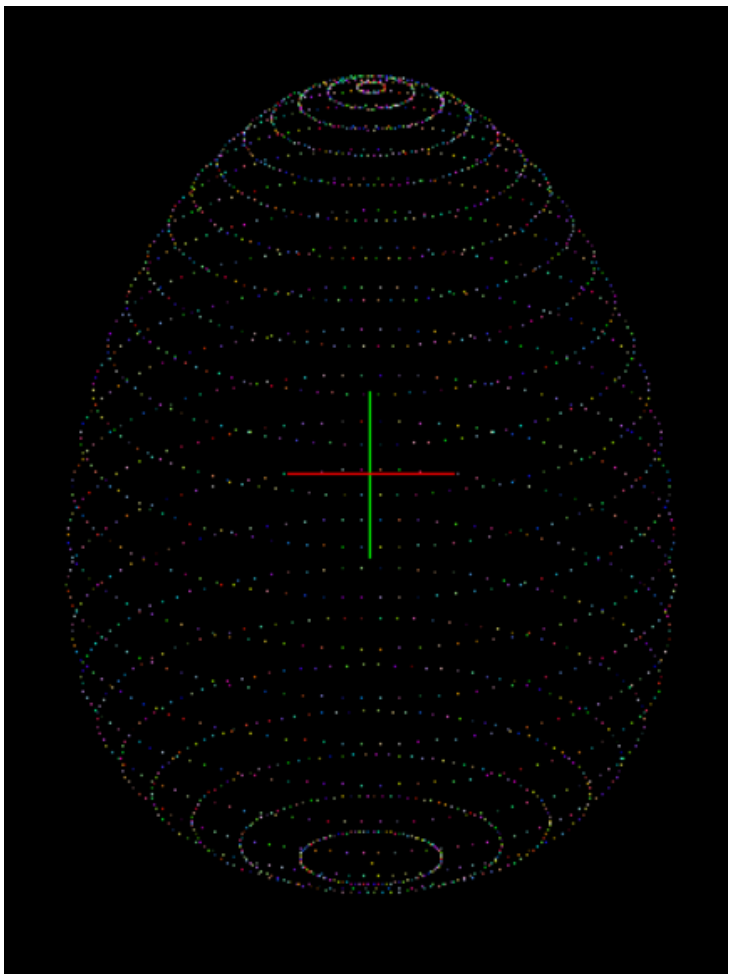
```
void Points() {
    for (int i = 0; i < N; i++) {
        float u = (float)i / (N - 1);
        for (int j = 0; j < N; j++) {
            float v = (float)j / (N - 1);

            points[i][j].x = (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);
            points[i][j].y = -5 + 160 * pow( x: u,  y: 4) - 320 * pow( x: u,  y: 3) + 160 * pow( x: u,  y: 2);
            points[i][j].z = (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);
        }
    }
}
```

5. Do tablicy NxN colors przypisywane są losowe kolory RGB dla każdego punktu.

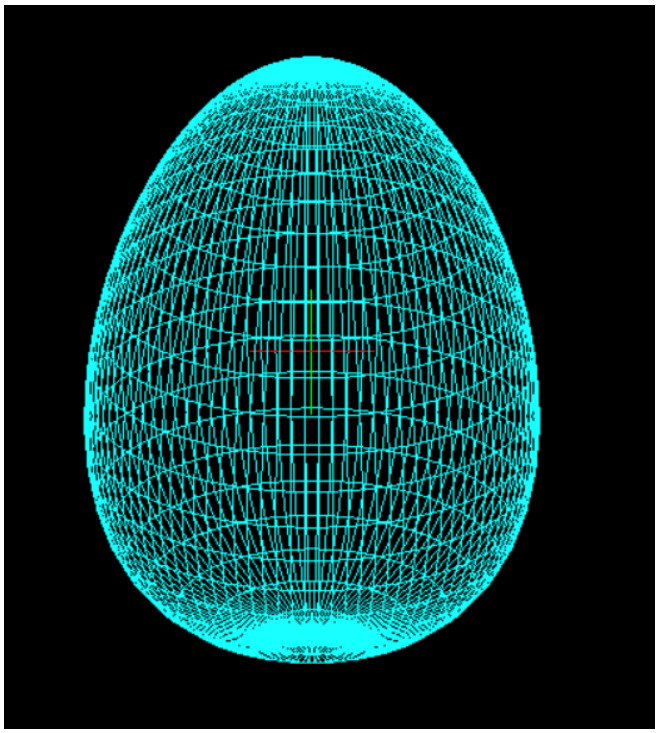
```
void Colors() {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            colors[i][j].x = (float(rand() % 11)) / 10;
            colors[i][j].y = (float(rand() % 11)) / 10;
            colors[i][j].z = (float(rand() % 11)) / 10;
        }
    }
}
```

W zależności od wybranego modelu jajko rysowane jest na jeden z trzech sposobów: chmura punktów (klawisz p), siatka (klawisz s), trójkąty (klawisz t). Przy wyborze czwartego modelu rysowany jest czajnik (klawisz c).

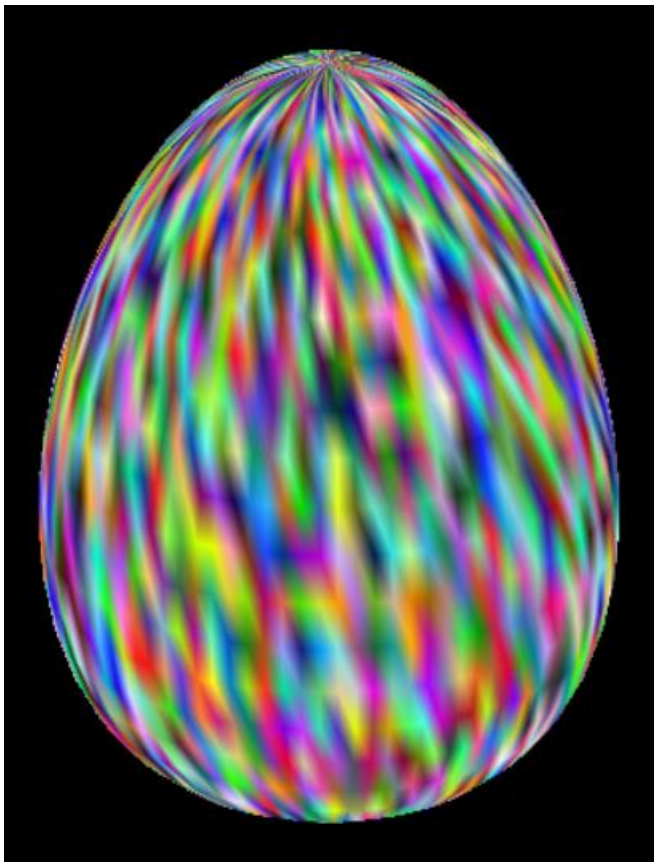
• Jajko z punktów



• Jajko z linii poziomych, pionowych i ukośnych



• Jajko z trójkątów



Program umożliwia włączenie automatycznej rotacji oraz rotacji względem osi x i z przez użytkownika. Funkcja zwrotna spinEgg() wywoływana za pomocą funkcji glutIdleFunc(spinEgg).

Funkcje spinX(), spinZ() opisują obrót wokół osi.

```
void spinX() {
    if (theta[0] > 360.0) theta[0] -= 360.0;
    if (theta[0] < -360.0) theta[0] += 360.0;
    glutPostRedisplay();
}

void spinZ() {
    if (theta[2] > 360.0) theta[2] -= 360.0;
    if (theta[2] < -360.0) theta[2] += 360.0;
    glutPostRedisplay();
}
```

Funkcja autospinEgg() opisuje automatyczną rotacji wokół każdej z osi.

```
void autospinEgg() {
    theta[0] -= 0.005;
    if (theta[0] > 360.0) theta[0] -= 360.0;

    theta[1] -= 0.005;
    if (theta[1] > 360.0) theta[1] -= 360.0;

    theta[2] -= 0.005;
    if (theta[2] > 360.0) theta[2] -= 360.0;

    glutPostRedisplay();
}
```

4. Wnioski

Podczas wykonania ćwiczenia zapoznałem się z techniką modelowania obiektów w przestrzeni 3D, z funkcjami obsługi zdarzeń klawiatury oraz wprowadzaniu obiektów w obrót.