Kurs:
Grafika komputerowa i komunikacja człowiek-komputer - Laboratorium

Modelowanie i wizualizacja obiektu 3D opisanego równaniem parametrycznym

Autor:
Aleksandra Chrustek, 263900

Prowadzący:
dr inż. Jan Nikodem

29.11.2023

1 Wstęp

Celem drugiego laboratorium było zapoznanie się z prostym modelowaniem 3D opisanym równaniem parametrycznym. Do zrozumienia modelowania niezbędne było zapoznanie się z nowymi funkcjami bibliotek OpenGL i GLUT, jak również z instrukcją zamieszczoną na stronie ZSK, która pozwalała na osiągnięcie wiedzy postawowej i niezbędnej, aby przystąpienie do ćwiczenia było możliwe.

2 Przebieg laboratorium

Pierwszym uruchomionym programem był układ współrzędnych, stworzony na podstawie kodu przedstawionego w instrukcji. Następnie na układzie współrzędnych został wygenerowany dzbanek do herbaty. Przedstawione w instrukcji funkcje bibliotek graficznych pozwoliły na obrócenie dzbanka o określoną ilość stopni. Kolejną częścią laboratorium było zadanie do samodzielnego zrealizowania - stworzenie trójwymiarowego jajka według podanych w instrukcji równań parametrycznych. Jajko miało zostać stworzone w trzech wersjach modelu. Pierwsza z nich to jajko składające się z kropek, druga to jajko stworzone za pomocą linii, a trzecia wersja była modelem złożonym z siatki trójkątów, które następnie miały zostać pokolorowane.

3 Realizacja zadania

3.1 Obliczenie współrzędnych punktów

W instrukcji były podane trzy równania parametryczne:

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

Rysunek 1: Równania parametryczne

Do tablicy trójwymiarowej NxN należało wprowadzić wartości dla osi x, y i z. Najpierw została zadeklarowana trójwymiarowa tablica przechowująca współrzędne punktów na powierzchni jajka. Zmienne u i v są zmiennymi pomocniczymi do parametryzacji powierzchni jajka. U jest używane do parametryzacji w kierunku poziomym, natomiast v w kierunku pionowym. W dwóch pętlach generowane są współrzędne punktów na

powierzchni jajka w zależności od parametrów u i v według podanych równań parametrycznych. Obie pętle generują pary indeksów (i,j), a następnie używają tych indeksów do obliczania wartości u i v. Te następnie są używane do obliczania współrzędnych punktu na powierzchni jajka w trzech wymiarach: tab[i][j][0], tab[i][j][1], tab[i][j][2]. Poniżej widoczny jest kod odpowiedzialny za wyżej opisane operacje.

```
float tab[100][100][3];
float u = 0.0;
float v = 0.0;

for (int i = 0; i < N; i++) {
for (int j = 0; j < N; j++) {
u = (float)i / (N - 1);
        // Obliczenie stosunku wartości i do zakresu (N - 1),
        // co daje wartość u w zakresie od 0.0 do 1.0
v = (float)j / (N - 1);
        // Obliczenie stosunku wartości j do zakresu (N - 1),
        // co daje wartość v w zakresie od 0.0 do 1.0

tab[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);
//X(u,v)
tab[i][j][1] = (160 * pow(u, 4) - 320 * pow(u, 3) +160 * pow(u, 2));
//Y(u,v)
tab[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);
//Z(u,v)
```

3.2 Osie układu współrzędnych

Funkcja Axes podana w instrukcji rysuje trzy osie układu współrzędnych w przestrzeni 3D. point3 jest aliasem dla tablicy trzech wartości zmiennoprzecinkowych, reprezentujących współrzędne punktu w trójwymiarowej przestrzeni. x_min, x_max, y_min, y_max, z_min, z_max są punktami określającymi początki i końce trzech osi układu współrzędnych (x, y, z).

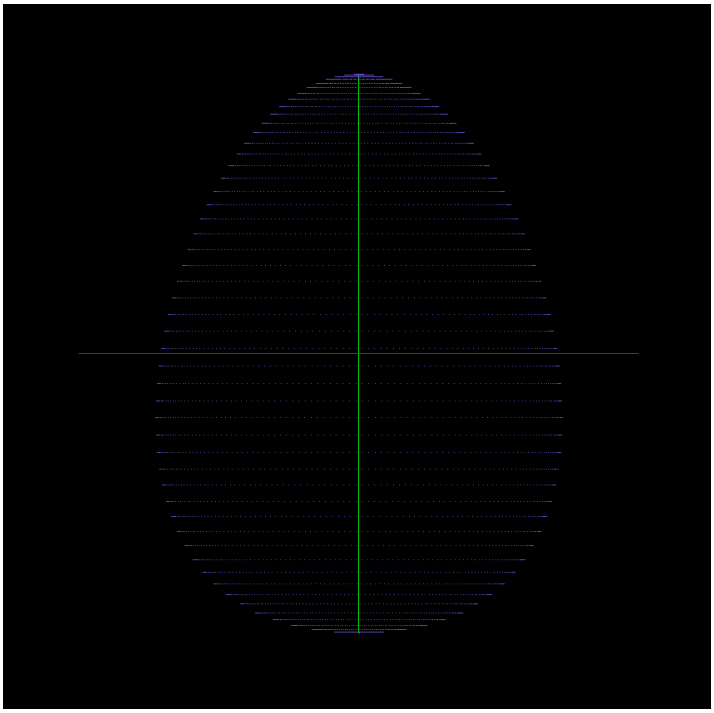
```
void Axes(void)
{
point3  x_min = { -5.0, 0.0, 0.0 };
point3  x_max = { 5.0, 0.0, 0.0 };
// początek i koniec obrazu osi x
```

```
point3  y_min = { 0.0, -5.0, 0.0 };
point3  y_max = { 0.0,  5.0, 0.0 };
// początek i koniec obrazu osi y

point3  z_min = { 0.0, 0.0, -5.0 };
point3  z_max = { 0.0, 0.0,  5.0 };
//  początek i koniec obrazu osi z

glColor3f(1.0f, 0.0f, 0.0f);  // kolor rysowania osi - czerwony
glBegin(GL_LINES); // rysowanie osi x
glVertex3fv(x_min);
glVertex3fv(x_max);
glEnd();

glColor3f(0.0f, 1.0f, 0.0f);  // kolor rysowania - zielony
glBegin(GL_LINES);  // rysowanie osi y
glVertex3fv(y_min);
glVertex3fv(y_max);
glEnd();

glColor3f(0.0f, 0.0f, 1.0f);  // kolor rysowania - niebieski
glBegin(GL_LINES); // rysowanie osi z
glVertex3fv(z_min);
glVertex3fv(z_max);
glEnd();
}
```

3.3 Pierwszy model

Pierwszy model jajka miał składać się z chmury punktów. Ten model wyświetla się po uruchomieniu programu, jak również w przypadku, gdy użytkownik wciśnie klawisz p. W tym celu zostało zainicjowane rysowanie punktów funkcją glBegin(GL_POINTS). Oznacza to, że każda kolejna funkcja glVertex3fv będzie interpretowana jako pojedynczy punkt w trójwymiarowej przestrzeni. Następnie w dwóch pętlach zostają dodane punkty o współrzędnych tab[i][j] do aktualnego bufora rysowania przy pomocy funkcji glVertex3fv(tab[i][j]). Tablica tab zawiera wcześniej obliczone współrzędne punktów na powierzchni jajka. Poniżej został przedstawiony kod odpowiedzialny za wyrysowanie tego modelu jajka.

```
if (model == 1) {

glBegin(GL_POINTS);
for (int i = 0; i < N - 1; ++i) {
for (int j = 0; j < N - 1; ++j) {
glVertex3fv(tab[i][j]);
    }
}
glEnd();
}
```



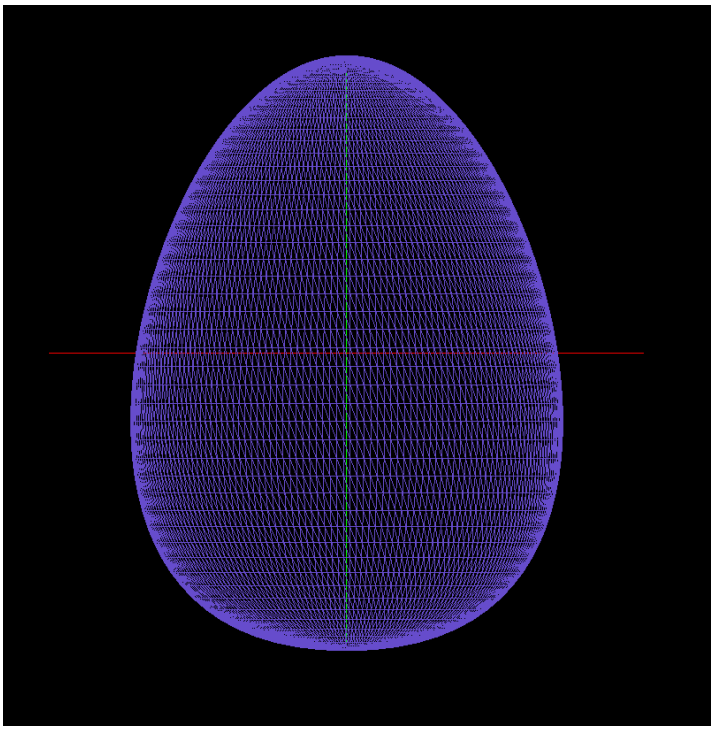
Rysunek 2: Model złożony z kropek

3.4 Drugi model

Drugi model jajka miał składać się z linii. Ten model wyświetla się w przypadku, gdy użytkownik wciśnie klawisz w. W tym celu zostało zainicjowane rysowanie linii funkcją glBegin(GL_LINES). Oznacza to, że każda para kolejnych funkcji glVertex3fv będzie interpretowana jako koniec jednej linii w trójwymiarowej przestrzeni. Następnie w dwóch pętlach zostają dodane linie do aktualnego bufora rysowania przy pomocy par funkcji glVertex3fv(). Tablica tab zawiera wcześniej obliczone współrzędne punktów na powierzchni jajka. Pary funkcji glVertex3fv(tab[i][j]) i glVertex3fv(tab[i][j + 1]) reprezen-

tują linie między kolejnymi punktami w jednym wierszu. Funkcje glVertex3fv(tab[i][j]) i glVertex3fv(tab[i + 1][j]) reprezentują linie między kolejnymi punktami w jednej kolumnie. Pary funkcji glVertex3fv(tab[i + 1][j]) i glVertex3fv(tab[i][j + 1]) reprezentują linie między sąsiednimi punktami na sąsiednich przekątnych. Funkcje glVertex3fv(tab[i + 1][j]) i glVertex3fv(tab[i + 1][j + 1]) reprezentują linie między kolejnymi punktami w jednym wierszu na przekątnej. Poniżej został przedstawiony kod odpowiedzialny za wyrysowanie tego modelu jajka.

```
if (model == 2) {

glBegin(GL_LINES);
for (int i = 0; i < N - 1; ++i) {
for (int j = 0; j < N - 1; ++j) {

glVertex3fv(tab[i][j]);
glVertex3fv(tab[i][j + 1]);

glVertex3fv(tab[i][j]);
glVertex3fv(tab[i + 1][j]);

glVertex3fv(tab[i + 1][j]);
glVertex3fv(tab[i][j + 1]);

glVertex3fv(tab[i + 1][j]);
glVertex3fv(tab[i + 1][j + 1]);
}
}
glEnd();
}
```



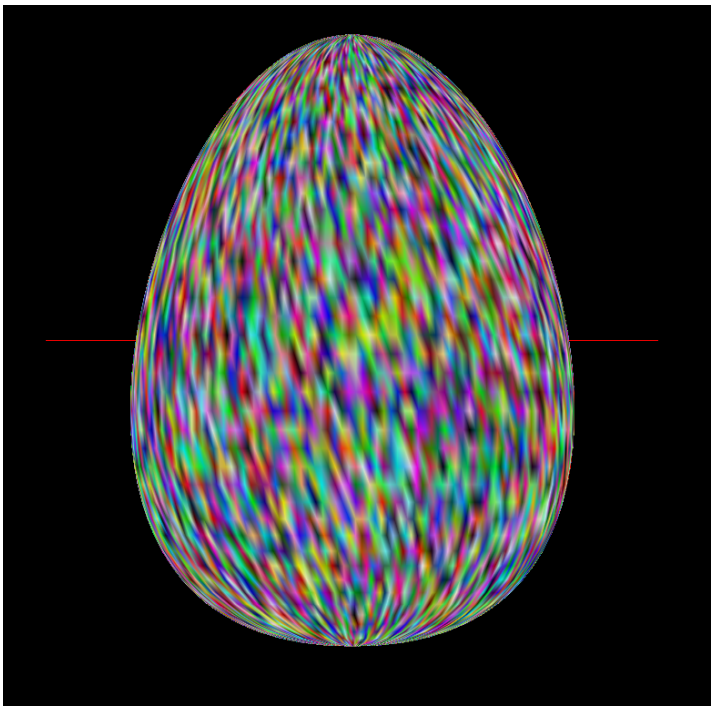
Rysunek 3: Model złożony z linii

3.5 Trzeci model

Trzeci model jajka miał składać się z siatki trójkątów pokolorowanych. Ten model wyświetla się w przypadku, gdy użytkownik wciśnie klawisz s. W tym celu zostało zainicjowane rysowanie trójkątów funkcją glBegin(GL_TRIANGLES). Oznacza to, że każde trzy kolejne pary funkcji glColor3fv i glVertex3fv będą reprezentować jeden trójkąt w trójwymiarowej przestrzeni. Następnie w dwóch pętlach zostają dodane trójkąty do aktualnego bufora rysowania przy pomocy trójek funkcji glVertex3fv(). Tablica tab zawiera wcześniej obliczone współrzędne punktów na powierzchni jajka. Kolor trójkąta jest ustawiany na podstawie koloru zdefiniowanego w macierzy colours przy pomocy funkcji glColor3fv(colours[i][j]). Poniżej został przedstawiony kod odpowiedzialny za wyrysowanie tego modelu jajka.

```
if (model == 3) {

for (int i = 0; i < N - 1; ++i) {
for (int j = 0; j < N - 1; ++j) {

glBegin(GL_TRIANGLES);
```

```
glColor3fv(colours[i][j + 1]);
glVertex3fv(tab[i][j + 1]);

glColor3fv(colours[i + 1][j]);
glVertex3fv(tab[i + 1][j]);

glColor3fv(colours[i + 1][j + 1]);
glVertex3fv(tab[i + 1][j + 1]);

glColor3fv(colours[i + 1][j]);
glVertex3fv(tab[i + 1][j]);

glColor3fv(colours[i][j + 1]);
glVertex3fv(tab[i][j + 1]);

glColor3fv(colours[i][j]);
glVertex3fv(tab[i][j]);

}
}
glEnd();
}
```



Rysunek 4: Model złożony z siatki trójkątów

3.6 Generowanie kolorów

Do pokolorowania jajka złożonego z trójkątów potrzebne było najpierw wygenerowanie losowych kolorów, które nastepnie zostały przypisane do trójwymiarowej macierzy kolorów colours. Funkcja makeRandomNumber() jest to funkcja, która zwraca losową liczbę z zakresu [0, 1) jako liczbę zmiennoprzecinkową. Funkcja makeRandomColour() jest to funkcja, która przypisuje losowe wartości kolorów do trójwymiarowej macierzy colours. Trzy zagnieżdżone pętle for są używane do iteracji po trzech wymiarach macierzy colours (indeksy i, j i k). W efekcie, funkcja makeRandomColour() wypełnia macierz colours losowymi wartościami, które zostały wykorzystane jako kolory dla trójkątów przy rysowaniu jajka.

```
float makeRandomNumber() {
    float random;
double const generateRandom = rand() % 10;

random = generateRandom / 10;

return random;
}
```

```
void makeRandomColour() {

for (int i = 0; i < N - 1; ++i) {
for (int j = 0; j < N - 1; ++j) {
for (int k = 0; k < N - 1; ++k) {

colours[i][j][k] = makeRandomNumber();
}
}
}
}
```

3.7 Obsługa za pomocą klawiatury

Funkcja keys obsługuje zdarzenia klawiszy (keyboard input) w programie. Przyjmuje trzy argumenty: unsigned char key - reprezentuje wciśnięty klawisz jako kod ASCII, int x, int y - określają współrzędne myszy w momencie wciśnięcia klawisza. Funkcja sprawdza, który z klawiszy p, w i s został wciśnięty i na tej podstawie wyświetlony zostaje odpowiedni model. Nastepnie zostaje wywołana funkcja RenderScene(), która jest odpowiedzialna za ponowne renderowanie sceny. Po zmianie wartości zmiennej model, ta funkcja zapewnia, że scena zostanie odświeżona z uwzględnieniem nowego trybu rysowania.

```
void keys(unsigned char key, int x, int y)
{
if (key == 'p') model = 1; // chmura punktów
if (key == 'w') model = 2; // model w postaci siatki
if (key == 's') model = 3; // model złożony z wypełnionych trójkątów

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

3.8 Obrót jajka

Funkcja spinEgg odpowiada za animację obracania jajka w trzech wymiarach. theta[0], theta[1], theta[2] są trzema kątami obrotu wokół osi x, y i z przechowywanymi w tablicy theta. Odejmowanie 0.5 stopnia od każdego z kątów obrotu odpowiada za animację obrotu, ponieważ przy każdym wywołaniu funkcji kąty są modyfikowane, co sprawia, że obiekt obraca się. Koneiczne jest także sprawdzenie, czy wartości kątów przekroczyły 360 stopni. Jeśli tak, to należy odjąć 360 stopni, aby zachować kąty w odpowiednich

zakresach. Następnie wywoływana jest funkcja glutPostRedisplay(), która powoduje ponowne odrysowanie aktualnego okna, co skutkuje animacją obrotu.

```
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
}
```

W celu obrotu należało dodać dodatkowe linie kodu w funkcji RenderScene():

```
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);
```

theta[0] jest kątem obrotu wokół osi x. Zapis 1.0, 0.0, 0.0 wskazuje na oś x, ponieważ pierwszy argument jest ustawiony na 1.0 (oś x), a pozostałe na 0.0 (osie y i z). Analogicznie jest to wykonane dla osi y i z.

4 Wnioski

Zadanie pozwoliło na poznanie i zastosowanie równań parametrycznych do generowania trójwymiarowego modelu jajka. Równania te stanowiły podstawę do obliczeń współrzędnych punktów na powierzchni jajka. Projektowanie trójwymiarowych modeli wymaga zrozumienia podstawowych koncepcji matematycznych, takich jak równania parametryczne, trygonometria czy algebra liniowa. W praktyce okazało się, że solidna podstawa matematyczna jest kluczowa dla skutecznego modelowania. Korzystanie z bibliotek OpenGL i GLUT okazało się skutecznym narzędziem do tworzenia trójwymiarowych scen. Ich funkcje, takie jak rysowanie punktów, linii czy trójkątów, umożliwiły stworzenie różnorodnych modeli jajka. Eksperymentowanie z różnymi trybami rysowania, takimi jak chmura punktów, linie i siatka trójkątów, pozwoliło zobaczyć, jak różne

metody prezentacji wpływają na odbiór grafiki 3D. Każdy z tych trybów ma swoje zastosowanie w różnych kontekstach wizualnych. Zastosowanie generowania losowych kolorów dla trójkątów modelu jajka wprowadziło element estetyki do projektu. Kolory te wpłynęły na wrażenia wizualne i nadają dodatkowy wymiar prezentowanym modelom. Mechanizm animacji obracania jajka w trzech wymiarach dodaje dynamiczności prezentacji. Pozwala to na lepsze zrozumienie struktury modelu oraz potencjalnych detali z różnych perspektyw. Teoretycznie, animacje w grafice 3D polegają na manipulacji transformacjami obiektów w trójwymiarowej przestrzeni. W praktyce animacje często są wykorzystywane do poprawy doświadczenia użytkownika oraz do eksploracji trójwymiarowych struktur. W rezultacie, zdobyta wiedza i umiejętności w zakresie modelowania 3D oraz korzystania z OpenGL i GLUT są cennym dodatkiem do mojej wiedzy programistycznej. Eksperymentowanie z różnymi aspektami grafiki komputerowej otworzyło drzwi do dalszego zgłębiania tego fascynującego obszaru informatyki.