pwr**POLITECHNIKA WROCŁAWSKA**

**Instytut Informatyki, Automatyki i Robotyki**

**Zakład Systemów Komputerowych**

**Grafika komputerowa i komunikacja człowiek – komputer**

**Kurs: INEK00012L**

**Sprawozdanie z ćwiczenia nr 3**

**„OpenGL – modelowanie obiektów 3D”**

|  |  |
| --- | --- |
| **Wykonał:** | **Adrian Frydmański** |
| **Termin:** | **PN/P 12:00-15:00** |
| **Data wykonania ćwiczenia:** | **27 X 2015** |
| **Data oddania sprawozdania:** | **9 XI 2015** |
| **Ocena:** |  |

|  |
| --- |
| **Uwagi prowadzącego:** |

# Wstęp teoretyczny

Celem ćwiczenia było narysowanie jajka w postaci wierzchołków, wierzchołków połączonych odcinkami i całych trójkątów. Odbywa się to przez zamianę punktów w przestrzeni trójwymiarowej na punkty na dwuwymiarowym kwadracie jednostkowym.

# Kod źródłowy

#include <windows.h>

#include <gl/gl.h>

#include <gl/glut.h>

#include <math.h>

#include <time.h>

#include <iostream>

using namespace std;

typedef float point3[3];

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Stałe i zmienne globalne:

const float pi = 3.14159265; // pi

point3 \*\*pTab; // tablica na punkty

point3 \*\*pRGB; // tablica na kolory

int n = 15; // do posziału kwadratu jednostkowego

int model = 1; // 1-punkty, 2-siatka, 3-kolorowe trojkaty

float squareLen = 1.0; // długość boku kwadratu jednostkowego

float len = 2; // długość osi

point3 v = {0.05, 0.05, 0.05}; // szybkość obracania się

static GLfloat theta[] = { 0.0, 0.0, 0.0 };

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Funkcje wyliczające współrzędne punktu (u,v) w przestrzeni 3D

float transformTo3D\_x(float u, float v)

{

float x, a = v \* pi;

x = (-90 \* pow(u, 5) + 225 \* pow(u, 4) - 270 \* pow(u, 3) + 180 \* pow(u, 2) - 45 \* u) \* cos(a);

return x;

}

float transformTo3D\_y(float u, float v)

{

float y;

y = 160 \* pow(u, 4) - 320 \* pow(u, 3) + 160 \* pow(u, 2);

return y - 5; // obniżenie jajka, żeby się ładnie mieściło

}

float transformTo3D\_z(float u, float v)

{

float z, a = v \* pi;

z = (-90 \* pow(u, 5) + 225 \* pow(u, 4) - 270 \* pow(u, 3) + 180 \* pow(u, 2) - 45 \* u) \* sin(a);

return z;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Generowanie siatki punktów:

void eggGenerate()

{

float step = squareLen / n;

// współrzędne 2D -> 3D

float u, v;

for (int i = 0; i<n + 1; i++)

for (int j = 0; j<n + 1; j++)

{

u = j \* step;

v = i \* step;

pTab[i][j][0] = transformTo3D\_x(u, v);

pTab[i][j][1] = transformTo3D\_y(u, v);

pTab[i][j][2] = transformTo3D\_z(u, v);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Renderowanie jajka

void Egg()

{

// Generowanie siatki

eggGenerate();

// Ustawienie koloru białego

glColor3f(1.0, 1.0, 1.0);

// switch w zależności od modelu

switch (model)

{

case 1: // punkty

{

glBegin(GL\_POINTS);

for (int i = 0; i<n; i++)

for (int j = 0; j<n; j++)

glVertex3fv(pTab[i][j]);

glEnd();

}

break;

case 2: // siatka

{

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

{

glBegin(GL\_LINES);

// pionowo

glVertex3fv(pTab[i][j]);

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

// poziomo

glVertex3fv(pTab[i][j]);

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

// w prawo w dół

glVertex3fv(pTab[i][j]);

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

glEnd();

}

}

break;

case 3: // trójkąty

{

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

{

// w jedną stronę

glBegin(GL\_TRIANGLES);

glColor3fv(pRGB[i][j + 1]);

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

glColor3fv(pRGB[i + 1][j]);

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

glColor3fv(pRGB[i + 1][j + 1]);

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

// w drugą stronę

glColor3fv(pRGB[i][j]);

glVertex3fv(pTab[i][j]);

glColor3fv(pRGB[i + 1][j]);

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

glColor3fv(pRGB[i][j + 1]);

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

glEnd();

}

}

break;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Funkcja rysująca osie układu współrzędnych

void Axes(float len)

{

// początek i koniec obrazu osi x

point3 x\_min = { -len, 0.0, 0.0 };

point3 x\_max = { len, 0.0, 0.0 };

// początek i koniec obrazu osi y

point3 y\_min = { 0.0, -len, 0.0 };

point3 y\_max = { 0.0, len, 0.0 };

// początek i koniec obrazu osi z

point3 z\_min = { 0.0, 0.0, -len };

point3 z\_max = { 0.0, 0.0, len };

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();

}

void allocate()

{

//Dynamiczna alokacja tablicy punktow

pTab = new point3\*[n + 1];

for (int i = 0; i < n + 1; i++)

pTab[i] = new point3[n + 1];

//Dynamiczna alokacja tablicy i wygenerowanie kolorow losowych dla punktow

pRGB = new point3\*[n + 1];

for (int i = 0; i < n + 1; i++)

pRGB[i] = new point3[n + 1];

for (int i = 0; i < n + 1; i++)

for (int j = 0; j < n + 1; j++)

{

pRGB[i][j][0] = ((float)(rand() % 10) + 1) / 10;

pRGB[i][j][1] = ((float)(rand() % 10) + 1) / 10;

pRGB[i][j][2] = ((float)(rand() % 10) + 1) / 10;

}

}

void relase()

{

//Zwolnienie pamięci

for (int i = 0; i < n + 1; i++)

{

delete[] pTab[i];

delete[] pRGB[i];

pTab[i] = 0;

pRGB[i] = 0;

}

delete[] pTab;

delete[] pRGB;

pTab = 0;

pRGB = 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Funkcja określająca co ma być rysowane (zawsze wywoływana, gdy trzeba przerysować scenę)

void RenderScene()

{

// Czyszczenie okna aktualnym kolorem czyszczącym

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

// Czyszczenie macierzy bieżącej

glLoadIdentity();

//Rotacje

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);

//Renderowanie osi

Axes(len);

//Renderowanie jajka

Egg();

// Przekazanie poleceń rysujących do wykonania

glFlush();

glutSwapBuffers();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Funkcja zwrotna dla klawiszy

void Keys(unsigned char key, int x, int y)

{

// zmiana modelu wyświetlania

if (key == 'p' || key == '1') model = 1;

if (key == 's' || key == '2') model = 2;

if (key == 't' || key == '3') model = 3;

// zmiana "rozdzielczości" jajka

if ( key == '+' || key == '-' || key == 72 || key == 80)

{

relase();

if (key == '+' || key == 72) n += 5;

if ((key == '-' || key == 80) && n > 0) n -= 5;

allocate();

}

// zmiana długości osi współrzędnych

if (key == '\*') len += 0.25;

if (key == '/' && len > 0) len -= 0.25;

// zmiana rędkości kręcenia się

if (key == '7') v[0] += 0.05;

if (key == '4') v[0] -= 0.05;

if (key == '8') v[1] += 0.05;

if (key == '5') v[1] -= 0.05;

if (key == '9') v[2] += 0.05;

if (key == '6') v[2] -= 0.05;

RenderScene();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Funkcja zwrotna dla obrotu

void spinEgg()

{

theta[0] -= v[0];

if (theta[0] > 360.0) theta[0] -= 360.0;

theta[1] -= v[1];

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

theta[2] -= v[2];

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

glutPostRedisplay(); //odświeżenie zawartości okna

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Funkcja ustalająca stan renderowania

void MyInit(void)

{

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

// Kolor czyszcący (wypełnienia okna) ustawiono na czarny

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Funkcja ma za zadanie utrzymanie stałych proporcji rysowanych obiektów w przypadku zmiany rozmiarów okna.

// Parametry vertical i horizontal są przekazywane do funkcji za każdym razem gdy zmieni się rozmiar okna.

void ChangeSize(GLsizei horizontal, GLsizei vertical)

{

// Deklaracja zmiennej AspectRatio określającej proporcję wymiarów okna

GLfloat AspectRatio;

if (vertical == 0) // Zabezpieczenie przed dzieleniem przez 0

vertical = 1;

// Ustawienie wielkościokna okna widoku (viewport)

// W tym przypadku od (0,0) do (horizontal, vertical)

glViewport(0, 0, horizontal, vertical);

// Przełączenie macierzy bieżącej na macierz projekcji

glMatrixMode(GL\_PROJECTION);

// Czyszcznie macierzy bieżącej

glLoadIdentity();

// Wyznaczenie współczynnika proporcji okna

// Gdy okno nie jest kwadratem wymagane jest określenie tak zwanej

// przestrzeni ograniczającej pozwalającej zachować właściwe

// proporcje rysowanego obiektu.

// Do okreslenia przestrzeni ograniczjącej służy funkcja

// glOrtho(...)

AspectRatio = (GLfloat)horizontal / (GLfloat)vertical;

if (horizontal <= vertical)

glOrtho(-7.5, 7.5, -7.5 / AspectRatio, 7.5 / AspectRatio, 10.0, -10.0);

else

glOrtho(-7.5\*AspectRatio, 7.5\*AspectRatio, -7.5, 7.5, 10.0, -10.0);

// Przełączenie macierzy bieżącej na macierz widoku modelu

glMatrixMode(GL\_MODELVIEW);

// Czyszcenie macierzy bieżącej

glLoadIdentity();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Główny punkt wejścia programu. Program działa w trybie konsoli

void main(void)

{

//włączenie polskich znaków

setlocale(LC\_ALL, "");

cout << "Sterowanie (najwygodniej na klawiaturze numerycznej):\n"

" p/1 punkty\n"

" s/2 siatka\n"

" t/3 trójkąty\n"

" + zwiększ rozdzielczosc jajka\n"

" - zmniejsz rozdzielczość jajka\n"

" \* wydłuż osie\n"

" / skróć osie\n"

" 7/8/9 zwiększ prędkość obrotu w osi X/Y/Z\n"

" 4/5/6 zmniejsz prędkość obrotu w osi X/Y/Z\n";

// zaalokuj tablicę

allocate();

// inicjowanie randa

srand((unsigned)time(NULL));

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(1024, 800);

glutCreateWindow("Modelowanie obiektow 3D - jajko (instrukcja w konsoli)");

// Określenie funkcji RenderScene jako funkcji zwrotnej (callback function)

// Będzie ona wywoływana za każdym razem, gdy zajdzie potrzba przeryswania okna

glutDisplayFunc(RenderScene);

// Dla aktualnego okna ustala funkcję zwrotną odpowiedzialną za zmiany rozmiaru okna

glutReshapeFunc(ChangeSize);

// Funkcja MyInit() wykonuje wszelkie inicjalizacje konieczne przed przystąpieniem do renderowania

MyInit();

// Włączenie mechanizmu usuwania powierzchni niewidocznych

glEnable(GL\_DEPTH\_TEST);

// Funkcja zwrotna dla klawiatury

glutKeyboardFunc(Keys);

// Funkcja zwrotna obrotu

glutIdleFunc(spinEgg);

// główna petla GLUTa

glutMainLoop();

// zwolnij pamięć

relase();

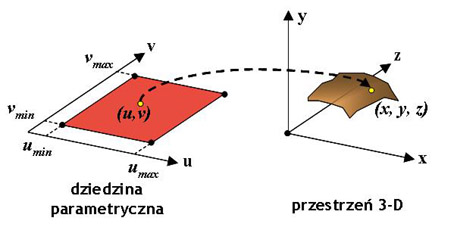
}

# Opis działania

Głównym zadaniem jest renderowanie jajka. Aby je wykonać trzeba wygenerować punkty należące do jajka. Będą one przechowywane w dwuwymiarowej tablicy trójelementowych tablic liczb zmiennoprzecinkowych (point3). Dzieje się to w funkcji eggGenerate. Zmienna step mówi o odległości między punktami w kwadracie jednostkowym. Transformacja z przestrzeni 3D na 2D odbywa się dzięki równaniom parametrycznym:

in (równaniom parametrycznym.ryzy punktami w kwadracie jednostkowym.

Od wyniku y zostało odjęte 5 w celu obniżenia pozycji jajka i widoczności całego w oknie – inaczej spód byłby w punkcie (0, 0, 0).

  
Rys. 1 Przekształcenie dziedziny parametrycznej w powierzchnię w przestrzeni 3-D

W zależności od wybranego trybu jajko zostaje narysowane jako zbiór punktów, krawędzie pomiędzy tymi punktami albo trójkąty utworzone z tychże krawędzi.

Narysowanie punktów sprowadza się do polecenia rysowania każdego punktu z dwuwymiarowej tablicy reprezentującej kwadrat jednostkowy.

Linie rysowane są w trzech pozycjach: pionowo, poziomo i przechylone w prawo w dół, jak to widać na poniższym schemacie:

Rys. 2 Łączenie wierzchołków

Rysowanie trójkątów odbywa się przez podanie trzech kolejnych wierzchołków dwóch trójkątów w kwadracie: A, B, D i D, B, C. Dodatkowo utworzona jest tablica podobna do tej zawierającej współrzędne punktów, jednakże zawiera ona składowe kolorów RGB każdego punktu, losowane przy inicjacji. Podczas rysowania trójkątów zmieniany jest kolor przed polecenie rysowania poszczególnego wierzchołka.

Rys. 3 Rysowanie trójkątów

D

C

A

B

Tablice punktów i kolorów są alokowane w głównej funkcji programu. Również tam zwalniana jest pamięć zajmowana przez nie, kiedy praca programu dobiega końca. Jednakże aby zapewnić możliwość zmiany rozdzielczości rysowania jajka podczas jego wizualizacji niezbędna jest zmiana tychże tablic. Następuje to w funkcji zwrotnej dla klawiszy przy wykryciu wciśnięcia + lub -. Wtedy stara tablica jest usuwana i po zmianie n wyrażającego liczbę rzędów i kolumn kwadratu jednostkowego tworzona jest nowa tablica.

Funkcja zwrotna reaguje również na wciśnięcia klawiszy p, s i t (lub 1, 2 i 3), zmieniając tryb wyświetlania pomiędzy punktami, siatką i trójkątami. Klawiszami 4 – 9 można zmienić prędkość obrotu względem poszczególnych osi – górne klawisze na klawiaturze numerycznej zwiększają prędkość (po kolei: X, Y i Z), a dolne zmniejszają. Dodatkowo, aby lepiej dostrzec obroty, można wydłużyć i skrócić osie współrzędnych. Odbywa się to za pomocą klawiszy \* i /.

# Podsumowanie

Początkowo rysowanie siatki jajka było problematyczne, gdyż skrajne punkty łączyły się z tymi po drugiej stronie poziom niżej, przechodząc przez wnętrze jajka i obie połówki były od siebie oddzielone. W innym przypadku z jajka wychodziły linie na zewnątrz. Ostatecznie, po zajęciach udało się w odpowiedni sposób połączyć punkty.

Obiekt jest kolorowany jest prawie prawidłowo – wierzchołki mają przypisane kolory, lecz skrajne różnią się od siebie przez co widać szew dookoła przekroju podłużnego jajka. Jedynie tego problemu nie udało się rozwiązać.