Grafika komputerowa i komunikacja człowiek - komputer

Open GL – modelowanie obiektów 3D

28.10.2022r.

Maciej Radecki
album: 253257

Prowadzący: dr inż. Jan Nikodem

Termin: Piątek TN, 16:25

1. Wstęp

Celem laboratorium było wprowadzenie za pomocą biblioteki OpenGL z rozszerzeniem GLUT idei modelowania i wizualizacji scen oraz obiektów w 3-D.

Do rysowania obiektów w 3-D wykorzystujemy przede wszystkim 3 osie układu współrzędnych – oś X, oś Y i oś Z. Każdy punkt znajdujący się na ekranie konsoli ma zdefiniowane położenie na każdej z tych osi.

Rysowanie obiektów w 3-D jest możliwe poprzez wywołanie funkcji rysujących już zaimplementowane domyślnie w bibliotece (np. bardzo często spotykany w literaturze czajnik do herbaty, wywoływany funkcją glutWireTeapot()) lub napisanie własnych implementacji obiektów (np. opisana dalej funkcja autorska Egg()).

Na obiektach możliwe jest stosowanie wielu różnych transformacji geometrycznych, np. obrót o liczbę stopni równą angle wokół osi wyznaczonej przez punkty o współrzędnych (0, 0, 0) i (x, y, z) za pomocą funkcji glRotated(TYPE angle, TYPE x, TYPE y, TYPE z) lub skalowanie za pomocą funkcji glScaled(TYPE x, TYPE y, TYPE z)), gdzie TYPE jest typem do ustalenia przez programistę (zazwyczaj float).

Do transformacji wykorzystywane są macierze, wykorzystujące operacje rachunku macierzowego. Kluczowe są dwie funkcje – glMatrixMode() – w skrócie ustala macierz bieżącą, która będzie następnie modyfikowana do zmiany macierzy bieżącej na inną. Istnieją 3 tryby macierzy, jednak w kontekście realizacji tego ćwiczenia istotne są dwa – GL_MODELVIEW (który powoduje, że modyfikowana będzie macierz widoku modelu, odpowiedzialna za przeliczanie geometrii) lub GL_PROJECTION (operacje macierzowe będą dotyczyć macierzy projekcji, odpowiedzialnej za rzutowanie).

Aby zbudować własny model obiektu 3-D, konieczne jest określenie go jako powierzchni opisanej równaniami parametrycznymi. Przykładowo, dla jajka wzory te wyglądają następująco:

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

Aby otrzymać chmurę punktów w przestrzeni 3-D, należy pokryć dziedzinę parametryczną (kwadrat jednostkowy w przestrzeni u, v) równomierną siatką punktów i przeliczyć współrzędne poszczególnych punktów siatki z dziedziny na współrzędne punktów w przestrzeni 3-D za pomocą opisanych powyżej równań. Algorytm zostanie przedstawiony i opisany w części praktycznej.

Biblioteka OpenGL umożliwia również komunikację człowiek-komputer za pośrednictwem funkcji zwrotnej (*callback function*). Funkcja glutKeyboardFunc(keys) pozwala na rejestrowanie funkcji zwrotnej, co z odrobiną kodu pozwala użytkownikowi np. na modyfikowanie obrazu po wciśnięciu danego klawisza lub napisaniu jakiegoś wyrażenia.

Wprawienie obiektu w ruch jest możliwe poprzez odpowiednio szybkie: modyfikację położenia obiektu oraz wyświetlenie go w nowej pozycji na ekranie. Tutaj również wykorzystywana jest funkcja zwrotna, jednak tym razem wykorzystana zostanie funkcja glutIdleFunc(). Funkcja ta jest wywoływana, gdy program jest bezczynny – za jej pomocą możemy więc poddać animacji pożądany obiekt, jeżeli w

funkcji zwrotnej odpowiednio będziemy modyfikować wartości kątów. Inne możliwe zastosowanie funkcji glutIdleFunc() to np. płynna zmiana kolorów obiektu wraz z przebiegiem czasu.

2. Realizacja ćwiczenia

Jako zadanie na laboratorium należało zmodyfikować zaprezentowany w instrukcji laboratoryjnej szkielet kodu w taki sposób, by wyświetlał on model jajka. Po wciśnięciu klawiszy: p, w lub s model ten ma ulegać zmianie – oryginalnie program wyświetla jajko zaimplementowane w postaci chmury pojedynczych punktów (do tego modelu można wrócić po kliknięciu klawisza p), po wciśnięciu klawisza w na ekranie pojawia się jajko wykonane z siatki odcinków, natomiast wciśnięcie klawisza s powoduje wyświetlenie jajka w postaci siatki kolorowych trójkątów. Dodatkowo, jajko się obraca.

Aby narysować jajko, należało opracować odpowiedni algorytm.

1. Zadać liczbę **N**, która określi, na ile przedziałów należy podzielić bok kwadratu jednostkowego dziedziny parametrycznej. W programie liczba N ma zdefiniowaną wartość równa 20 i jest zmienną lokalną, ale jest możliwa jest zmiana jej wartości przez programistę.

Warto wspomnieć, że aby podzielić bok kwadratu np. na 3 części, muszą znajdować się na nim 4 punkty. Punkt 1 i 2 tworzą 1-szą część, punkt 2 i 3 2-gą część, a punkt 3 i 4 3-cią część. Fakt ten został wykorzystany podczas pisania kodu podczas pisania pętli tworzących tablice lub łączących ze sobą punkty. Nieuwzględnienie go sprawiało, że przy późniejszym rysowaniu siatki odcinków jajko nie było poprawnie połączone.

2. Zadeklarować tablicę o rozmiarze **NxN**, służącej do zapisywania współrzędnych punktów w przestrzeni 3-D. Zostało to zrealizowane za pomocą dwuwymiarowej tablicy dynamicznej typu trzyelementowego *point3*, która przechowuje [N+1][N+1] wartości (z powodu opisanego powyżej) o składowych X, Y i Z.

3. Nałożyć na kwadrat jednostkowy dziedziny parametrycznej równomierną siatkę **NxN** punktów. Następnie każdemu elementowi tablicy przypisano wartość któregoś z punktów kwadratu jednostkowego za pomocą dwóch pętli w taki sposób, że punkty zostały równomiernie rozłożone na powierzchni kwadratu.

4. Dla każdego punktu u, v nałożonej w poprzednim kroku siatki, obliczono przy pomocy podanych we wstępie teoretycznym równań współrzędne x(u, v), y(u, v) oraz z(u, v), zapisując je do tablicy. Za pomocą dwóch pętli przypisano wszystkim punktom siatki odpowiednie wartości, dzięki czemu otrzymano kształt jajka.

5. Wyświetlić na ekranie elementy tablicy współrzędnych punktów posługując się konstrukcją glBegin() oraz glEnd(), służącą do rysowania obiektów pierwotnych na ekranie. Rysowanie punktów jest dość proste – wystarczą dwie pętle, by dla każdego elementu w tablicy narysować pojedynczy punkt na ekranie. Powstaje w ten sposób jajko wykonane z punktów.

Aby otrzymać jajko wykonane z odcinków siatki, dla każdego elementu tablicy należy narysować dwa odcinki – jeden poziomy (połączyć obecny punkt z punktem kolejnym wzdłuż linii poziomu) i jeden pionowy (łącząc obecny punkt z punktem kolejnym wzdłuż linii pionowej).

Aby otrzymać jajko wykonane z trójkątów, należy narysować dla każdego elementu tablicy dwa trójkąty – oba będą miały wspólny obecny punkt oraz punkt oddalony o w pionie i poziomie od obecnego, jednak jeden trójkąt będzie zawierał punkt wyłącznie oddalony w poziomie od obecnego, a drugi będzie posiadał punkt oddalony tylko w pionie od obecnego.

Aby uzyskać kolorowe trójkąty, zadeklarowano na początku dodatkową tablicę dwuwymiarową *Colours* o rozmiarze [N+1][N+1] i typu point3, reprezentującą składowe R, G i B punktu w układzie współrzędnych. Następnie wylosowano wartości RGB dla każdego punktu za pomocą funkcji losującej liczby pseudolosowe rand(). Po pierwszym wylosowaniu kolor punktów już się nie zmienia. Po wywołaniu funkcji rysującej trójkąty narysowane zostanie różnokolorowe jajko. Będzie ono posiadało płynne przejścia między kolorami, co wynika z faktu, że przyległe trójkąty mają przypisane ten sam kolor.

```
colours = new point3[n_points * n_points];
srand(time(NULL));
for (int i = 0; i < n_points * n_points; i++)
{
    colours[i][0] = (rand() % 101) * 0.01;
    colours[i][1] = (rand() % 101) * 0.01;
    colours[i][2] = (rand() % 101) * 0.01;
}
for (int i = 0; i < n_points; i++)
{
    colours[i][0] = colours[0][0];
    colours[i][1] = colours[0][1];
    colours[i][2] = colours[0][2];
    colours[n_points * n_points - 1 - i][0] = colours[n_points * n_points - 1][0];
    colours[n_points * n_points - 1 - i][1] = colours[n_points * n_points - 1][1];
    colours[n_points * n_points - 1 - i][2] = colours[n_points * n_points - 1][2];
}
```

Aby jajko się obracało, dokonano następujących kroków:

- zadeklarowano zmienną statyczną typu GLfloat theta[] = (0.0, 0.0, 0.0), w której przetrzymywane są kąty obecnego obrotu jajka wokół osi [x, y, z];

- dodano kolejną funkcję zwrotną (nazwaną spinEgg()), która modyfikuje wartość kąta obrotu zdefiniowaną w poprzednim punkcie i odświeży rysowany obraz za pomocą funkcji wbudowanej glutPostRedisplay()

```
void spinEgg()
{
    theta[0] = theta[0] - 0.5;
    if (theta[0] > 360)
        theta[0] = theta[0] - 360;

    theta[1] = theta[1] - 0.5;
    if (theta[1] > 360)
        theta[1] = theta[1] - 360;

    theta[2] = theta[2] - 0.5;
    if (theta[2] > 360)
        theta[2] = theta[2] - 360;

    glutPostRedisplay();
}
```

- dodanie funkcji glutIdleFunc(spinEgg), która jest wywoływana nieustannie, gdy obraz jest w stanie bezczynnym (pod względem programistycznym, czyli nie jest obsługiwany przez żadną inną funkcję), w połączeniu z argumentem spinEgg oznacza to wywołanie obrotu jajka i wrażenie ruchu;

- dodanie linijek kodu powodujących obrót modelu jajka wokół osi x, y i z o kąty wyliczone w funkcji spinEgg():

```
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);
```

Wskutek zaimplementowania algorytmu program rysuje jajko zbudowane z punktów, siatki odcinków bądź kolorowych trójkątów, które się nieustannie obraca. Możliwa jest zmiana dokładności jajka poprzez zmianę wartości parametru N przez programistę. Wciśnięcie klawiszy p, w i s zmienia obecny model jajka. Okno konsolowe przekazuje użytkownikowi informację o tym, w jaki sposób może zmienić model jajka.

```
void keys(unsigned char key, int x, int y)
{
    if (key == 'p')
        model = 1;
    if (key == 'w')
        model = 2;
    if (key == 's')
        model = 3;
    if (key == 't')
        model = 4;
    RenderScene(); // przerysowywanie obrazu sceny
}
```

Aby jajko było w pełni widoczne na ekranie (a nie odcięte u góry), przesunięto je tak, by jego środek mniej więcej znajdował się na środku układu współrzędnych zdefiniowanym w instrukcji.

3. Podsumowanie i wnioski

Poziom realizowanego laboratorium stanowił duży przeskok względem wcześniejszych zajęć. Instrukcja nie nakierowywała tak dobrze jak poprzednio. Podczas zajęć wystąpił problem z niecałkowitym rysowaniem jajka (brak części siatki i punktów, szczelina w jajku z trójkątów) który udało rozwiązać się już po zajęciach poprzez ponowne sprawdzenie zakresów wykonywania się pętli.