## WOJSKOWA AKADEMIA TECHNICZNA

# *LABORATORIUM*

# *GRAFIKA KOMPUTEROWA*

Stopień, imię i nazwisko słuchacza

Grupa szkoleniowa

Stopień, imię i nazwisko prowadzącego

***Grzegorz Pol***

***I7X3S1***

***dr inż. Marek Salamon***

Data wykonania ćwiczenia

***10.12.2008 r.***

***SPRAWOZDANIE***

***Z***

***PRACY LABORATORYJNEJ***

***NR 3***

|  |  |
| --- | --- |
| **Temat:** | Modelowanie prostych obiektów przy pomocy prymitywów. |

* 1. **Zadanie**

Podczas zajęć laboratoryjnych należało wykonać zadanie o poniższej treści:

Napisać program przedstawiający obiekt zbudowany z prymitywów przestrzennych udostępnionych przez biblioteki GLU i GLUT. Użytkownik za pomocą klawiatury powinien mieć możliwość wprowadzania zmian następujących parametrów:

1. Prędkości kątowej obrotu śmigła w zakresie [0-10] stopni/klatkę animacji z krokiem 1 w kierunku CCW
2. Prędkości kątowej lotu samolotu po okręgu o promieniu 30 wokół osi Y (y=4.0) w zakresie [0-5] stopni/klatkę animacji z krokiem 0.25 w kierunku CW.

W programie uwzględnić możliwość interakcyjnej zmiany położenia obserwatora poprzez podanie następujących parametrów:

1. Odległości obserwatora od obiektu
2. Orientacji obserwatora w zakresie [0,360] stopni względem osi 0X, 0Y, 0Z
	1. **Użyte funkcje biblioteki GLUT i GLU**

Funkcje podstawowe:

**PushMatrix();** - Odkłada ona na stos macierz przekształcenia. Wywołujemy ją na początku każdego przekształcenia układu współrzędnych. Zapamiętuje ona kolejne transformacje w zadanej kolejności. Używanie przekształceń jest potrzebne ponieważ każda z gotowych brył jest rysowana w środku układu współrzędnych. W takim razie aby umieścić ją w odpowiednim dla nas miejscu musimy przenieść układ.

**PopMatrix();** - Pobiera odłożoną na stos macierz przekształcenia, a następnie wykonuje odpowiednie transformacje.

Funkcje przekształcające układ współrzędny:

**glTranslatef(x,y,z);** - przenosi ona środek układu o wektor = [x,y,z] gdzie x,y,z są typu float.

**glRotatef(a,x,y,z);** - obraca układ o kąt a podany przez nas w stopniach w kierunku przeciwnym do ruchu wskazówek zegara wokół wektora [x,y,z]

**glScalef(sx,sy,sz);** - nakłada skalowanie wzdłuż osi X - współczynnik skali sx, osi Y - współczynnik skali sy, osi Z - współczynnik skali sz.

Funkcje wstawiające bryły:

**glutWireSphere(r,pol,row);** - Wstawia sferę ze środkiem w początku układu współrzędnych o promieniu r i ilości południków pol oraz ilości równoleżników row.

**glutSolidCube(a);** - Wstawia sześcian ze środkiem w początku układu współrzędnych o boku a. Ściany jego są wypełnione kolorem w przeciwieństwie do funkcji glutWireCube(a);.

Funkcje wstawiające bryły wymagające wczesniejszego inicjowania:

**gluCylinder(obj,r1,r2,h,pol,row);** - Wstawia cylinder ze środkiem w początku układu współrzędnych o promieniach dolnym r1 i górnym r2, wysokości h oraz liczbie południków pol i liczbie równoleżników row.

**gluDisk(obj,r1,r2,pol,row);** - Wstawia dysk ze środkiem w początku układu współrzędnych o promieniu środkowym r1, promieniu zewnętrznym r2 oraz liczbie południków pol i liczbie równoleżników row.

* 1. **Cel zadania**

Celem zajęć laboratoryjnych było zbudowanie samolotu widocznego na poniższym rysunku. Samolot podzieliłem na 7 kształtów:



1. Śmigło

Kod kształtu reprezentujący śmigło jest w naszym przypadku najbardziej złożony, ze względu na to, że użytkownik w końcowym programie ma mieć możliwość wpływu na prędkość obrotu. W rzeczywistości obrót śmigła polega na zmianie jednego parametru rotSmiglaK. Kształt śmigła stworzyłem za pomocą funkcji glutSolidCube() i odpowiednich przekształceń skalowania oraz przesunięć.

 //rysowanie smigla

 glPushMatrix();

 glRotatef(rotSmiglaK, 0, 0, 1);

 glRotatef(90, 0, 0, 1);

 glTranslatef(0, 0, -5-.75-.25/2);

 glScalef(4, .25, .25);

 glutSolidCube(1);

 glPopMatrix();

1. Przednia część kadłuba samolotu

W tym wypadku kod kształtu imitującego przednią część kadłuba samolotu to sfera złożona z zarówno 6 równoleżników jak i południków. Użyta została funkcja glutWireSphere() przy odpowiednim przesunięciu.

//rysowanie przedniej czesci kadluba samolotu

 glPushMatrix();

 glTranslatef(0.0, 0.0, -5.0);

 glutWireSphere(.75, 6, 6);

 glPopMatrix();

1. Kadłub

Kod kształtu reprezentującego kadłub wymaga wcześniejszego zainicjowania w naszym programie:

 kadlub = gluNewQuadric();

 gluQuadricDrawStyle(kadlub, GLU\_LINE);

Teraz już śmiało możemy użyć funkcji gluCylinder(); która to narysuje nam walec (promień początkowy i końcowy jest taki sam). Pamiętamy jednak aby wcześniej przesunąć odpowiednio układ by nasz kadłub ukazał nam się w oczekiwanym przez nas miejscu

 //rysowanie kadluba

 glPushMatrix();

 glRotatef(0, 1, 0, 0);

 glTranslatef(0.0, 0.0, -5.0);

 gluCylinder(kadlub, .75, .75, 10.0, 20, 4);

 glPopMatrix();

1. Skrzydła

Kod kształtu imitującego skrzydła jest bliźniaczo podobny do kodu śmigła, ze względu na użucie tej samej funkcji glutSolidCube(). Różnice oczywiście dostrzegamy w skalowaniu układu współrzędnych przed jej wywołaniem.

//rysowanie skrzydel

 glPushMatrix();

 glTranslatef(0.0, .75-.25/2, 0.0);

 glScalef(8.0, .25, 3);

 glutWireCube(1);

 glPopMatrix();

1. Statecznik pionowy

Kod kształtu reprezentujący statecznik pionowy jest także podobny do kodu śmigła czy skrzydeł, także ze względu na użucie tej samej funkcji glutSolidCube(). Różnice oczywiście dostrzegamy w skalowaniu układu współrzędnych przed jej wywołaniem. Ponadto tutaj musimy także ją obrócić.

 //rysowanie statecznika pionowego

 glPushMatrix();

 glRotatef(90, 0, 0, 1);

 glTranslatef(1.5, 0, 5-1.5/2);

 glScalef(2, .25, 1.5);

 glutWireCube(1);

 glPopMatrix();

1. Tylnia część kadłuba samolotu

Tylnia część kadłubu samolotu to taka jakby podstawa walca, który udaje w naszym projekcie kadłub. Tutaj tak samo jak w przypadku właśnie kadłuba musimy ją wcześniej zainicjować:

 tylnia\_czesc\_kadluba = gluNewQuadric();

 gluQuadricDrawStyle(tylnia\_czesc\_kadluba, GLU\_LINE);

W przeciwieństwie jednak do kadłuba tutaj używamy funkcje gluDisk();, odpowiednio najpierw przesuwając środek układu współrzędnych.

 //rysowanie tylniej czesci kadluba samolotu

 glPushMatrix();

 glTranslatef(0.0, 0.0, 5.0);

 gluDisk(tylnia\_czesc\_kadluba, 0.0, .75, 20, 4);

 glPopMatrix();

1. Statecznika poziomego

Ostatnim naszym kształtem jest odpowiednio wyskalowany kwadrat reprezentujący statecznik poziomy. Jest on wykonany identycznie jak np. statecznik pionowy oczywiście różniąc się przesunięciem i wyskalowaniem układu współrzędnych przed użyciem funkcji gluSolidCube();

 //rysowanie statecznika poziomego

 glPushMatrix();

 glTranslatef(0.0, 0, 5-1.5/2);

 glScalef(4+.75\*2, .25, 1.5);

 glutWireCube(1);

 glPopMatrix();

* 1. **Sterowanie**

|  |  |
| --- | --- |
| C:\Users\Grzesio!\Desktop\q.bmp | - przyśpieszanie lotu samolotu |
|  |  |
| C:\Users\Grzesio!\Desktop\w.bmp | - spowalnianie lotu samolotu |
|  |  |
| C:\Users\Grzesio!\Desktop\e.bmp | - zwiększanie obrotów śmigłem |
|  |  |
| C:\Users\Grzesio!\Desktop\r.bmp | - zmniejszanie obrotów śmigłem |
|  |  |
| C:\Users\Grzesio!\Desktop\up.bmp | - przybliżanie obserwatora |
|  |  |
| C:\Users\Grzesio!\Desktop\down.bmp | - oddalanie obserwatora |
|  |  |
| C:\Users\Grzesio!\Desktop\left.bmp | - obracanie w lewo względem osi Y |
|  |  |
| C:\Users\Grzesio!\Desktop\right.bmp | - obracanie w prawo względem osi Y |
|  |  |
| C:\Users\Grzesio!\Desktop\home.bmp | - obracanie w lewo względem osi X |
|  |  |
| C:\Users\Grzesio!\Desktop\pgup.bmp | - obracanie w prawo względem osi X |
|  |  |
| C:\Users\Grzesio!\Desktop\pgdn.bmp | - obracanie w lewo względem osi Z |
|  |  |
| C:\Users\Grzesio!\Desktop\end.bmp | - obracanie w prawo względem osi Z |
|  |  |
| C:\Users\Grzesio!\Desktop\esc.bmp | - wyjście z programu |

* 1. **Wizualny efekt końcowy**

****

****

* 1. **Wnioski**

Podczas tego laboratorium zapoznaliśmy się z modelowanie bardziej skomplikowanych obiektów niż miało to miejsce na poprzednich laboratoriach. Tym razem modelowanie kształtów było o wiele prostsze ze względu na możliwość wykorzystania prymitywów przestrzennych dostępnych w bibliotekach GLU i GLUT. Ponadto urozmaiciliśmy nasz obiekt przedstawiający samolot zmuszając go do latania oraz obracania śmigłem. Aby bez przeszkód dobrze wykonać postawione na laboratoriach zadanie niezbędne jest obycie z funkcjami oraz pamiętanie o czyszczeniu macierzy przed narysowaniem następnego kształtu.

* 1. **Kod źródłowy**

#include "stdafx.h"

#include <GL/glut.h>

/////////////////////////////////////////

//Wskazniki do wykorzystywanych kwadryk//

/////////////////////////////////////////

GLUquadricObj \*kadlub;

GLUquadricObj \*tylnia\_czesc\_kadluba;

//////////////////////

//Ustawienia wstepne//

//////////////////////

GLfloat lotV = .250;

GLfloat lotK = 0.0;

GLfloat rotSmiglaV = 1.0;

GLfloat rotSmiglaK = 0.0;

GLfloat odlObs = -70.0;

GLfloat rotObsY = 40.0;

GLfloat rotObsX = 40.0;

GLfloat rotObsZ = 0.0;

GLfloat rotPodstawy = 0.0;

GLfloat rotRamienia1= 40.0;

GLfloat rotRamienia2= -40.0;

GLfloat rotGlowicy = 20.0;

GLfloat rozUchwytow = 0.5;

///////////////////////////////////////////////////////////////////

//Funkcja inicjujaca elementy skladowe zamodelowane jako kwadryki//

///////////////////////////////////////////////////////////////////

void InicjujRamieRobota(void)

{

 // Zainicjowanie kwadryki kadluba

 kadlub = gluNewQuadric();

 gluQuadricDrawStyle(kadlub, GLU\_LINE);

 // Zainicjowanie kwadryki tylniej czesci kadluba

 tylnia\_czesc\_kadluba = gluNewQuadric();

 gluQuadricDrawStyle(tylnia\_czesc\_kadluba, GLU\_LINE);

}

/////////////////////////////////////////////////////////////////////////

//Funkcja rysujaca obraz sceny widzianej z biezacej pozycji obserwatora//

/////////////////////////////////////////////////////////////////////////

void Rysuj\_samolocik(GLfloat obrotPodstawy, GLfloat obrotRamienia1, GLfloat obrotRamienia2, GLfloat obrotGlowicy, GLfloat rozstawUchwytow)

{

 // Pocztaek tworzenia ukladu wspolrzednych

 glBegin(GL\_LINES);

 // Os X

 glColor3f(1.0, 0.0, 0.0);

 glVertex3f(-20.0, 0.0, 0.0);

 glVertex3f(20.0, 0.0, 0.0);

 // Os Y

 glColor3f(0.0,1.0,0.0);

 glVertex3f(0.0, -20.0, 0.0);

 glVertex3f(0.0, 20.0, 0.0);

 // Os Z

 glColor3f(0.0,0.0,1.0);

 glVertex3f(0.0, 0.0, -20.0);

 glVertex3f(0.0, 0.0, 20.0);

 // Koniec tworzenia ukladu wspolrzednych

 glEnd();

 glColor3f(1.0,1.0,1.0);

////////////////////////////////////////////

//Rysowanie poszczegolnych czesci samolotu//

////////////////////////////////////////////

 glPushMatrix(); // Przygotowanie stosu macierzy modelowania

 glRotatef(lotK, 0, 1, 0);

 glTranslatef(30.0, 4.0, 0.0);

 //rysowanie kadluba

 glPushMatrix();

 glRotatef(0, 1, 0, 0);

 glTranslatef(0.0, 0.0, -5.0);

 gluCylinder(kadlub, .75, .75, 10.0, 20, 4);

 glPopMatrix();

 //rysowanie tylniej czesci kadluba samolotu

 glPushMatrix();

 glTranslatef(0.0, 0.0, 5.0);

 gluDisk(tylnia\_czesc\_kadluba, 0.0, .75, 20, 4);

 glPopMatrix();

 //rysowanie przedniej czesci kadluba samolotu

 glPushMatrix();

 glTranslatef(0.0, 0.0, -5.0);

 glutWireSphere(.75, 6, 6);

 glPopMatrix();

 //rysowanie skrzydel

 glPushMatrix();

 glTranslatef(0.0, .75-.25/2, 0.0);

 glScalef(8.0, .25, 3);

 glutSolidCube(1);

 glPopMatrix();

 //rysowanie statecznika poziomego

 glPushMatrix();

 glTranslatef(0.0, 0, 5-1.5/2);

 glScalef(4+.75\*2, .25, 1.5);

 glutSolidCube(1);

 glPopMatrix();

 //rysowanie statecznika pionowego

 glPushMatrix();

 glRotatef(90, 0, 0, 1);

 glTranslatef(1.5, 0, 5-1.5/2);

 glScalef(2, .25, 1.5);

 glutSolidCube(1);

 glPopMatrix();

 //rysowanie smigla

 glPushMatrix();

 glRotatef(rotSmiglaK, 0, 0, 1);

 glRotatef(90, 0, 0, 1);

 glTranslatef(0, 0, -5-.75-.25/2);

 glScalef(4, .25, .25);

 glutSolidCube(1);

 glPopMatrix();

 glPopMatrix(); // Posprzatanie na stosie macierzy modelowania

}

/////////////////////////////////////////////////

//Funkcja generujaca pojedyncza klatke animacji//

/////////////////////////////////////////////////

void WyswietlObraz(void)

{

 glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT); // Wyczyszczenie bufora ramki i bufora glebokosci

 glPushMatrix(); // Powielenie macierzy na wierzcholku stosu

 // Wyznaczenie polozenia obserwatora (przeksztalcenie uladu wspolrzednych sceny do ukladu wspolrzednych obserwatora).

 glTranslatef(0, 0, odlObs);

 glRotatef(rotObsX, 1, 0, 0);

 glRotatef(rotObsY,0,1,0);

 glRotatef(rotObsZ,0,0,1);

 rotSmiglaK = (rotSmiglaK < 360) ? rotSmiglaK + rotSmiglaV : 0; //obrot smigla

 lotK = (lotK < 360) ? lotK + lotV : 0; //lot samolotu

 Rysuj\_samolocik(rotPodstawy, rotRamienia1, rotRamienia2, rotGlowicy, rozUchwytow); // Generacja obrazu sceny w niewidocznym buforze ramki

 glPopMatrix(); // Usuniecie macierzy lezacej na wierzcholku stosu (powrot do stanu sprzed wywolania funkcji)

 glutSwapBuffers(); // Przelaczenie buforow ramki

}

//////////////////////////////////////////////////////////////////////////////

//Funkcja ustawiajaca parametry rzutu perspektywicznego i rozmiary viewportu//

//////////////////////////////////////////////////////////////////////////////

void UstawParametryWidoku(int szerokosc, int wysokosc)

{

 glViewport(0, 0, szerokosc, wysokosc); // Ustawienie parametrow viewportu

 // Przejscie w tryb modyfikacji macierzy rzutowania

 glMatrixMode(GL\_PROJECTION);

 glLoadIdentity();

 gluPerspective(40.0, (float)szerokosc/(float)wysokosc, 1.0, 1000.0);

 glMatrixMode(GL\_MODELVIEW); // Przejscie w tryb modyfikacji macierzy przeksztalcen geometrycznych

 glLoadIdentity(); // Zmiana macierzy znajdujacej sie na wierzcholku stosu na macierz jednostkowa

}

////////////////////////////////////////

//Funkcja obsługi klawiszy specjalnych//

////////////////////////////////////////

void ObslugaKlawiszySpecjalnych(int klawisz, int x, int y)

{

 switch(klawisz)

 {

 // ustawianie odleglosci obserwatora

 case GLUT\_KEY\_UP:

 odlObs = (odlObs < 0.0) ? odlObs + 1.0 : odlObs;

 break;

 case GLUT\_KEY\_DOWN:

 odlObs = (odlObs > -200.0) ? odlObs - 1.0 : odlObs;

 break;

 // obracanie osia Y

 case GLUT\_KEY\_LEFT:

 rotObsY = (rotObsY > 0) ? rotObsY - 1.0 : rotObsY;

 break;

 case GLUT\_KEY\_RIGHT:

 rotObsY = (rotObsY < 360.0) ? rotObsY + 1.0 : rotObsY;

 break;

 // obracanie osia X

 case GLUT\_KEY\_HOME:

 rotObsX = (rotObsX > 0) ? rotObsX - 1.0 : rotObsX;

 break;

 case GLUT\_KEY\_PAGE\_UP:

 rotObsX = (rotObsX < 360.0) ? rotObsX + 1.0 : rotObsX;

 break;

 // obracanie osia Z

 case GLUT\_KEY\_END:

 rotObsZ = (rotObsZ > 0) ? rotObsZ - 1.0 : rotObsZ;

 break;

 case GLUT\_KEY\_PAGE\_DOWN:

 rotObsZ = (rotObsZ < 360.0) ? rotObsZ + 1.0 : rotObsZ;

 break;

 }

}

//////////////////////////////

//Funkcja obsługi klawiatury//

//////////////////////////////

void ObslugaKlawiatury(unsigned char klawisz, int x, int y)

{

 switch(klawisz)

 {

 // sterowanie predkoscia samolotu

 case 'q':

 lotV = (lotV < 5 ) ? lotV + .25 : 5;

 break;

 case 'w':

 lotV = (lotV > 0) ? lotV - .25 : 0;

 break;

 // sterowanie predkoscia smigla

 case 'e':

 rotSmiglaV = (rotSmiglaV < 10.0) ? rotSmiglaV + 1.0 : 10;

 break;

 case 'r':

 rotSmiglaV = (rotSmiglaV > 0.0) ? rotSmiglaV - 1.0 : 0;

 break;

 }

 if(klawisz == 27)

 exit(0);

}

///////////////////////////

//Glowna funkcja programu//

///////////////////////////

int main(int argc, char \*\*argv)

{

 glutInit(&argc, argv); // Zainicjowanie biblioteki GLUT

 glutInitDisplayMode (GLUT\_DOUBLE|GLUT\_RGB|GLUT\_DEPTH); // Ustawienie trybu wyswietlania

 glutInitWindowPosition(100, 100); // Ustawienie polozenia dolenego lewego rogu okna

 glutInitWindowSize(600, 600); // Ustawienie rozmiarow okna

 glutCreateWindow("Samolocik"); // Utworzenie okna

 glEnable(GL\_DEPTH\_TEST); // Odblokowanie bufora glebokosci

 glDepthFunc(GL\_LEQUAL); // Ustawienie funkcji wykonywanej na danych w buforze glebokosci

 glClearDepth(1000.0); // Ustawienie wartosci czyszczacej zawartosc bufora glebokosci

 glClearColor (0.3, 0.3, 0.3, 0.3); // Ustawienie koloru czyszczenia bufora ramki

 glutDisplayFunc(WyswietlObraz); // Zarejestrowanie funkcji (callback) wyswietlajacej

 glutReshapeFunc(UstawParametryWidoku); // Zarejestrowanie funkcji (callback) wywolywanej za kazdym razem kiedy zmieniane sa rozmiary okna

 glutIdleFunc(WyswietlObraz); // Zarejestrowanie funkcji wykonywanej gdy okno nie obsluguje zadnych zadan

 glutKeyboardFunc(ObslugaKlawiatury); // Zarejestrowanie funkcji obslugi klawiatury

 glutSpecialFunc(ObslugaKlawiszySpecjalnych); // Zarejestrowanie funkcji obslugi klawiszy specjalnych

 InicjujRamieRobota(); // Zainicjowanie kwadryk tworzacych ramie robota

 glutMainLoop(); // Obsluga glownej petli programu (wywolywanie zarejestrowanych callbackow w odpowiedzi na odbierane zdarzenia lub obsluga stanu bezczynnosci)

 return 0;

}