Лабораторная работа №2

Уткин А.В 4210 вариант №2

Задание:  
  
1. Используя операции преобразования модели, создайте программу:

Прямоугольник, произвольно заданный в МСК, масштабируется относительно своего центра и пульсирует

2. Для вариантов, приведенных ниже, создайте программу, которая рисует сцену, состоящую из фиг.0 и нескольких фиг.1. Для фиг.1 создайте отдельную функцию, описывающую ее в СК0, и затем перемещайте фиг.1 из позиции в позицию. В отчете для каждой позиции фиг.1 выпишите последовательность движений фиг.1, в результате которых она попадает в заданную позицию. Кроме того, считайте, что функция для фиг.1 описывает ее в локальной системе координат СКL, а полученные преобразования понимайте как преобразования координат из СКL в СК0. Для каждой позиции фиг1. изобразите цепочку преобразований систем координат и все координатные фреймы как это сделано на рис.2.

Примечание. Fig.0 и Fig.1 во всех позициях должны быть нарисованы с положительными координатными полуосями (стрелки не рисовать).

Листинг программы №1:

#include <stdlib.h>

#include <math.h>

#include <iostream>

#define GL\_SILENCE\_DEPRECATION

#include <GLUT/glut.h>

#include <unistd.h>

GLfloat R = 640.0 / 480; //Форматное соотношение

GLfloat w = 40; //Ширина мирового окна

GLfloat h; //Высота мирового окна

GLfloat l, r, b, t; //Параметры мирового окна

GLfloat f = 0;

**void** init(**void**)

{

h = w / R; l = -w / 2; r = w / 2; b = -h / 2; t = h / 2; //Расчет параметров миро-вого окна

glClearColor(1.0, 1.0, 1.0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(l, r, b, t);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

**void** reshape(GLsizei W, GLsizei H)

{

**if** (R > W / H) glViewport(0, 0, W, W / R);

**else** glViewport(0, 0, H \* R, H);

}

**void** fig0(**void**)

{

glColor3f(1.0, 0.0, 1.0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(-3.0, -3.0);

glVertex2f(-3.0, 3.0);

glVertex2f(5.0, 3.0);

glVertex2f(5.0, -3.0);

glEnd();

}

**void** scene(**void**)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glPushMatrix();

glScalef(f, -f, f);

fig0();

glPopMatrix( );

glFlush();

glutSwapBuffers();

f+=0.5; **if**(f==2.0) f=0;

usleep(100000);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640, 480);

glutInitWindowPosition(20, 20);

glutCreateWindow("Myprog");

glutReshapeFunc(reshape);

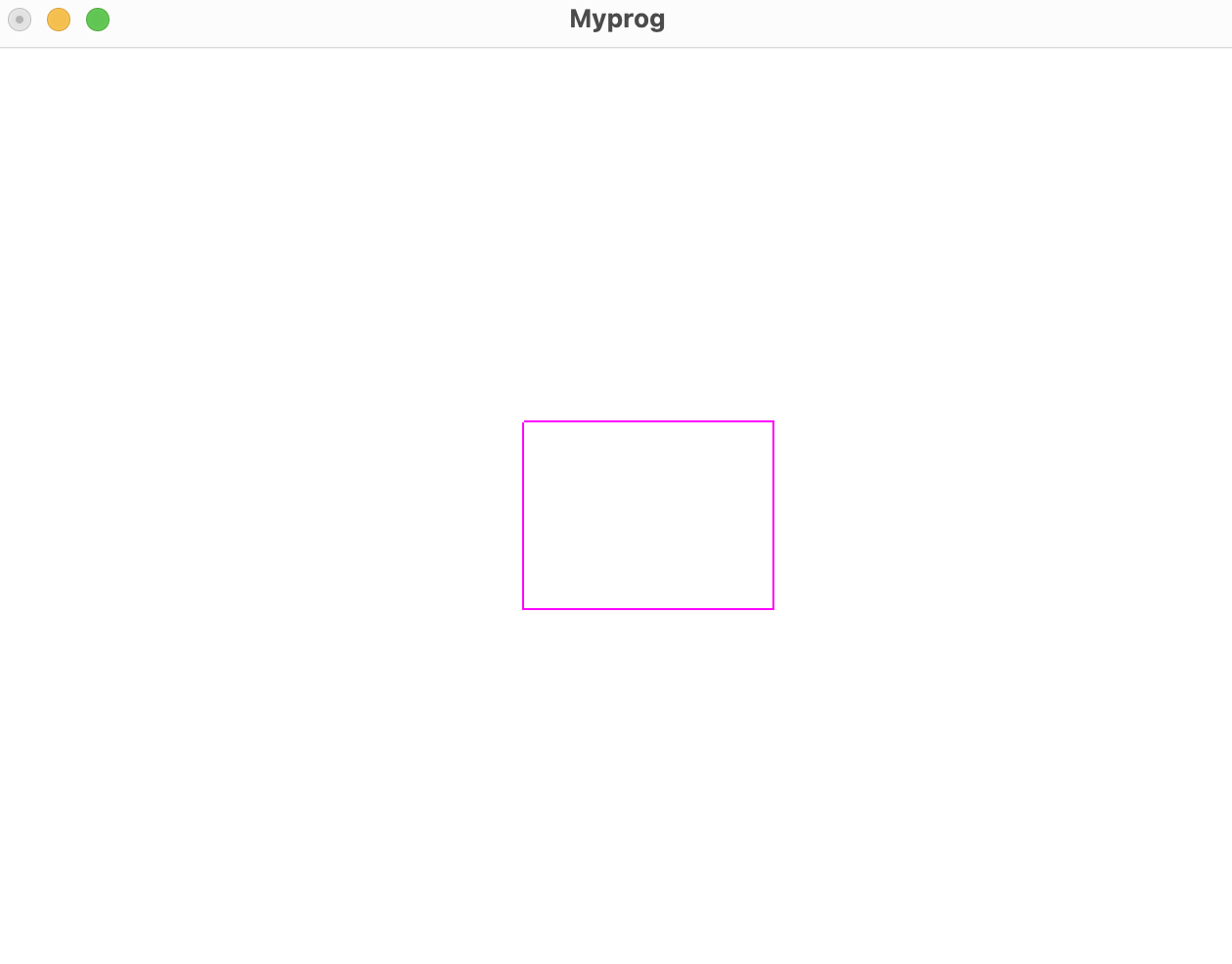
glutDisplayFunc(scene);

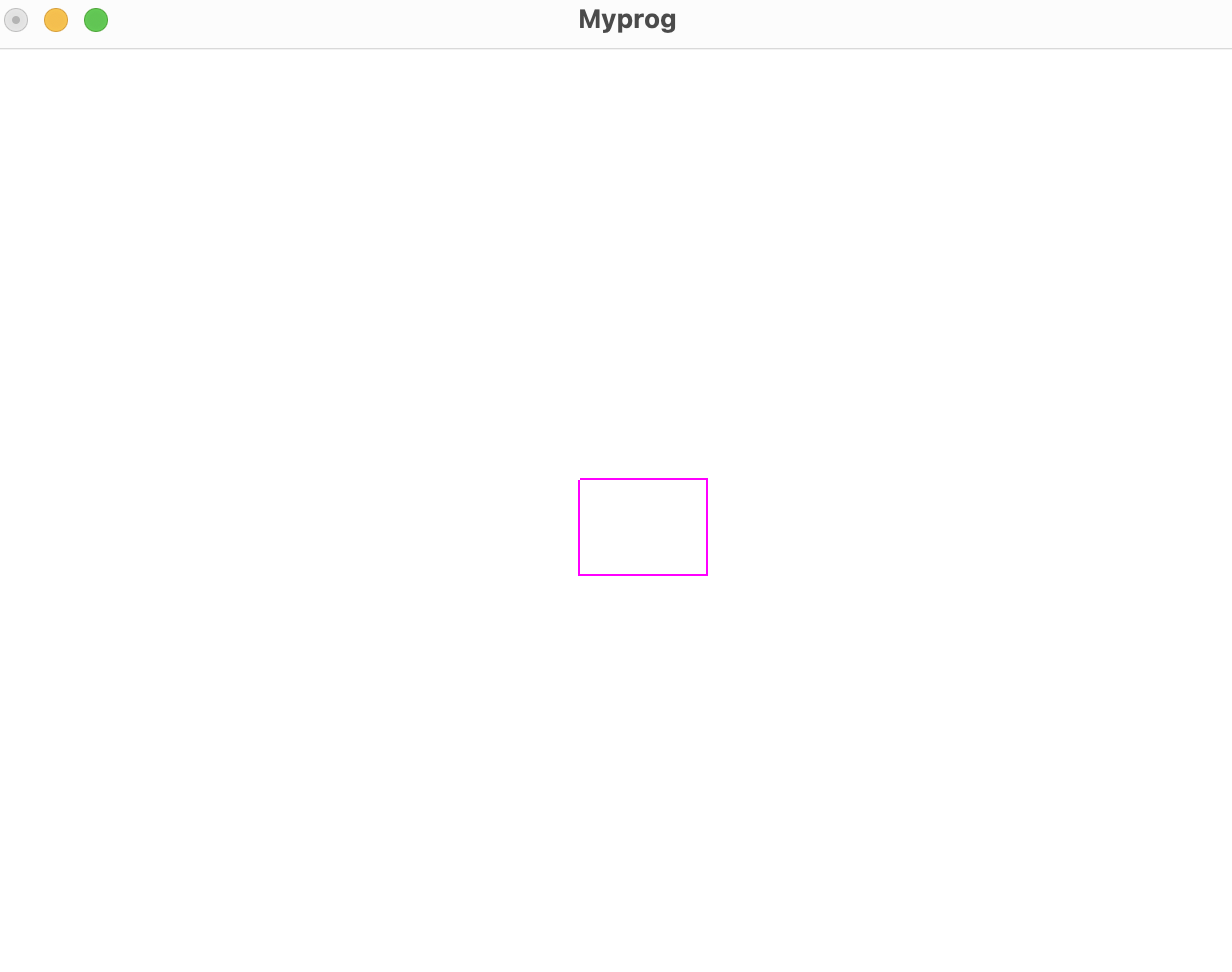
glutIdleFunc(scene);

init();

glutMainLoop();

}





Листинг программы №2

#include <stdlib.h>

#include <math.h>

#include <iostream>

#define GL\_SILENCE\_DEPRECATION

#include <GLUT/glut.h>

#include <unistd.h>

GLfloat R = 640.0 / 480;

GLfloat w = 40, h;

GLfloat l, r, b, t;

GLfloat f = 0.0;

GLfloat angle = 0;

**void** init(**void**)

{

h = w / R; l = -w / 2; r = w / 2; b = -h / 2; t = h / 2;

glClearColor(1.0, 1.0, 1.0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(l, r, b, t);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

**void** reshape(GLsizei W, GLsizei H)

{

**if** (R > W / H) glViewport(0, 0, W, W / R);

**else** glViewport(0, 0, H \* R, H);

}

**void** showAxis(**void**)

{

glColor3f(0.0, 1.0, 0.0);

glBegin(GL\_LINES);

glVertex2f(0, 0);

glVertex2f(0, t);

glVertex2f(0, 0);

glVertex2f(r, 0);

glEnd();

}

**void** fig0(**float** x, **float** y, **float** r, **int** amountSegments)

{

glColor3f(0.0, 0.0, 0.0);

glBegin(GL\_LINE\_LOOP);

**for** (**int** i = 0; i < amountSegments; i++)

{

**float** angle = 2.0 \* 3.1415926 \* **float**(i) / **float**(amountSegments);

**float** dx = r \* cosf(angle);

**float** dy = 0.5 \* r \* sinf(angle); // divide the y-coordinate by 2 to center the oval shape

glVertex2f(x + dx, y + dy);

}

glEnd();

glBegin(GL\_LINES);

// glVertex2f(-1.0, 0.0);

// glVertex2f(10.0, 0.0);

glVertex2f(-1.0, 0.0);

glVertex2f(10.0, 0.0);

glEnd();

glBegin(GL\_LINES);

glVertex2f(2.5, -1.5);

glVertex2f(2.5, 3.0);

glEnd();

}

**void** drawCircle(**float** x, **float** y, **float** r, **int** amountSegments)

{

glColor3f(1.0, 0.0, 0.0);

glBegin(GL\_LINE\_LOOP);

**for** (**int** i = 0; i < amountSegments; i++)

{

**float** angle = 2.0 \* 3.1415926 \* **float**(i) / **float**(amountSegments);

**float** dx = r \* cosf(angle);

**float** dy = r \* sinf(angle);

glVertex2f(x + dx, y + dy);

}

glEnd();

}

**void** scene(**void**)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

drawCircle(0, 0, 5, 100);

glPushMatrix();

glTranslatef(0.0, 0.0, 0.0);

glRotatef(f, 0.0, 0.0, 1.0);

glTranslatef(-10.0, 0.0, 0.0);

glScalef(1.0, -1.0, 1.0);

fig0(2.5, 0, 2.5, 200);

glPopMatrix();

glFlush();

glutSwapBuffers();

usleep(100000);

}

**void** timer(**int** value)

{

f += 90;

**if** (f >= 360) { f = 0; }

glutPostRedisplay();

glutTimerFunc(500, timer, 0);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(640, 480);

glutInitWindowPosition(20, 20);

glutCreateWindow("Myprog");

glutReshapeFunc(reshape);

glutDisplayFunc(scene);

glutTimerFunc(500, timer, 0);

glutIdleFunc(scene);

init();

glutMainLoop();

**return** 0;

}

