1. **Напишите программу вывода графика функции y= tg (x)+5\*cos (x), с помощью точек, а затем с помощью линий.**
   1. **С помощью точек**

**Решение:**

#include <windows.h>

#include <gl\gl.h>

#include <gl\glu.h>

#include <gl\glaux.h>

#include <math.h>

GLint windW, windH;

void CALLBACK Reshape(int width, int height)

{ glViewport(0, 0, width, height);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-15,15,-15,15);

glMatrixMode(GL\_MODELVIEW);

}

void CALLBACK Draw(void)

{

glClearColor(1.0,1.0,1.0,1);

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_LINES);

glColor3ub(190,190,190);

for (int i=-4; i<=4; i++)

{

glVertex2f(float(i), -6);//явное преобразование типа

glVertex2f(float(i), 15);

}

for (i=-6; i<=15; i++)

{

glVertex2f(-4, float(i));

glVertex2f(4, float(i));

}

glEnd();

glColor3ub(0,0,0);

glBegin(GL\_LINES);

glVertex2f (-5, 0);

glVertex2f(5, 0);

glVertex2f(0,16);

glVertex2f(0,-7);

glEnd();

glPointSize(2);

glBegin(GL\_POINTS);

glColor3ub(0,0,255);

int n;

double a,b,dx,x,y;

a=-1.415;

b=1.5;

n=20;

dx=(b-a)/(n-1);

x=a;

y=0;

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

{

y=float(tan(x)+5\*cos(x));

glVertex2d (x, y);

x=x+dx;

}

glEnd();

glFinish();

auxSwapBuffers();

}

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

{

windW = 800;

windH = 800;

auxInitPosition(100, 100, windW, windH);

auxInitDisplayMode(AUX\_RGB | AUX\_DOUBLE);

auxInitWindow("v11\_01\_1");

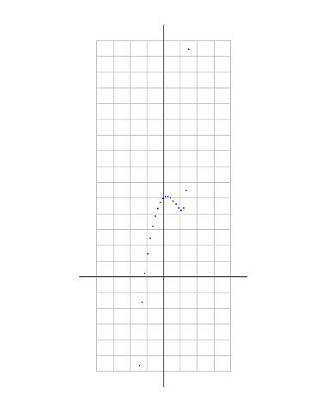
glTranslated(0,-4,0);

auxReshapeFunc(Reshape);

auxMainLoop(Draw);

}

**Результат:**



* 1. **с помощью линий**

**Решение:**

#include <windows.h>

#include <gl\gl.h>

#include <gl\glu.h>

#include <gl\glaux.h>

#include <math.h>

GLint windW, windH;

void CALLBACK Reshape(int width, int height)

{ glViewport(0, 0, width, height);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-15,15,-15,15);

glMatrixMode(GL\_MODELVIEW);

}

void CALLBACK Draw(void)

{

glClearColor(1.0,1.0,1.0,1);

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_LINES);

glColor3ub(190,190,190);

for (int i=-4; i<=4; i++)

{

glVertex2f(float(i), -6);//явное преобразование типа

glVertex2f(float(i), 15);

}

for (i=-6; i<=15; i++)

{

glVertex2f(-4, float(i));

glVertex2f(4, float(i));

}

glEnd();

glColor3ub(0,0,0);

glBegin(GL\_LINES);

glVertex2f (-5, 0);

glVertex2f(5, 0);

glVertex2f(0,16);

glVertex2f(0,-7);

glEnd();

glBegin(GL\_LINE\_STRIP);

glColor3ub(0,0,255);

int n;

double a,b,dx,x,y;

a=-1.415;

b=1.5;

n=100;

dx=(b-a)/(n-1);

x=a;

y=0;

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

{

y=float(tan(x)+5\*cos(x));

glVertex2d (x, y);

x=x+dx;

}

glEnd();

glFinish();

auxSwapBuffers();

}

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

{

windW = 800;

windH = 800;

auxInitPosition(100, 100, windW, windH);

auxInitDisplayMode(AUX\_RGB | AUX\_DOUBLE);

auxInitWindow("v11\_01\_1");

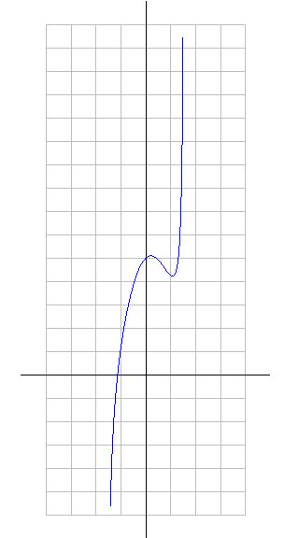
glTranslated(0,-4,0);

auxReshapeFunc(Reshape);

auxMainLoop(Draw);

}

**Результат:**



1. **Нарисуйте каркас призмы, в основании которой лежит правильный 14-угольник**

**Решение:**

//v11\_02

#include <windows.h>

#include <GL/gl.h>

#include <GL/glu.h>

#include <GL/glaux.h>

#include <math.h>

void CALLBACK resize (int width, int height)

{

glViewport (0,0,width, height);

glMatrixMode (GL\_PROJECTION);

glLoadIdentity();

glOrtho(-15,15,-10,10, -10,20);

gluLookAt(1,-1,1, 0,0,0, 0,0,1);

glMatrixMode(GL\_MODELVIEW);

}

void CALLBACK display (void)

{

GLUquadricObj \*quadObj;

quadObj = gluNewQuadric();

glClearColor(1.0,1.0,1.0,1);

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

glColor3d(0,0,1);

// каркас

gluQuadricDrawStyle(quadObj, GLU\_SILHOUETTE);

//призма в основании которой лежит правильный 14-угольник

gluCylinder(quadObj, 2, 2, 5, 14, 14);

glBegin(GL\_LINES);

glColor3ub(0, 0, 0);

glVertex2f(6, 0);

glVertex2f(-6, 0);

glVertex2f(0, 6);

glVertex2f(0, -6);

glVertex3f(0, 0, 0);

glVertex3f(0, 0, 8);

for (int i=-5; i<=5; i++)

{

if (i!=0)

{

glColor3ub(190,190,190);

glVertex2f(float(i), -5);

glVertex2f(float(i), 5);

}

}

for (i=-5; i<=5; i++)

{

if (i!=0)

{

glColor3ub(190,190,190);

glVertex2f(-5, float(i));

glVertex2f(5, float(i));

}

}

glEnd();

glFinish();

auxSwapBuffers();

}

void main ()

{

auxInitPosition (100,100,800,600);

auxInitDisplayMode (AUX\_RGB | AUX\_DEPTH | AUX\_DOUBLE);

auxInitWindow ("v11\_02");

glScaled(1.5, 1.5, 1.5);

glTranslated(0,0,-1);

auxIdleFunc (display);

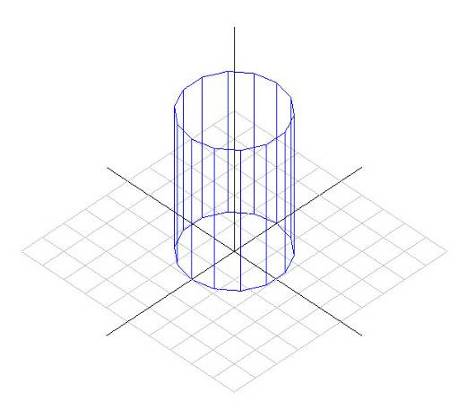
auxReshapeFunc(resize);

glEnable (GL\_DEPTH\_TEST);

auxMainLoop(display);

}

**Результат:**



1. **Напишите программу вывода графика функции x=2\*sin(z)\*cos(y)-3\*tg(y) используя алгоритм плавающего горизонта. Модифицируйте программу таким образом, чтобы поверхность состояла из четырехугольников**

**Решение:**

#include <windows.h>

#include <gl\gl.h>

#include <gl\glu.h>

#include <gl\glaux.h>

#include <math.h>

// запретить вывод предупреждений о преобразовании данных

#pragma warning(disable: 4305) // MIPS

#pragma warning(disable: 4244) // MIPS

GLint windW, windH;

void CALLBACK Reshape(int width, int height)

{

windW = (GLint)width;

windH = (GLint)height;

}

int alpha=0, beta=0;

void CALLBACK Key\_LEFT(void)

{

alpha -= 5;

}

void CALLBACK Key\_RIGHT(void)

{

alpha += 5;

}

void CALLBACK Key\_UP(void)

{

beta += 5;

}

void CALLBACK Key\_DOWN(void)

{

beta -= 5;

}

void CALLBACK InitViewport(int x, int y, int width, int height)

{

glViewport(x, y, width, height);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-windW,windW, -windH,windH, windH/1000,windH\*1000);

gluLookAt(0,0,windH, 0,0,0, 1,0,0);

glMatrixMode(GL\_MODELVIEW);

}

void CALLBACK Draw(void)

{

float x,y,z;

float d=10;

InitViewport(0,0, windW, windH);

glColor3d(0,1,0);

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE);

for (y=-windH/1.3; y<=windH/1.3; y+=10)

{glBegin(GL\_QUADS);

for (z=-windW/1.3; z<=windW/1.3; z+=10)

{

x=2\*sin(z)\*cos(y)-3\*tan(y);

glVertex3f(x,y,z);

glVertex3f(x,y+d,z);

x=2\*sin(z+d)\*cos(y+d)-3\*tan(y+d);

glVertex3f(x,y+d,z+d);

glVertex3f(x,y,z+d);

}

glEnd();

}

glFinish();

}

void CALLBACK Turn(void)

{

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

glPushMatrix();

glRotated(alpha, 0,1,0);

glRotated(beta, -1,0,0);

Draw();

glPopMatrix();

auxSwapBuffers();

glFinish();

}

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

{

windW = 800;

windH = 600;

auxInitPosition(0, 0, windW, windH);

auxInitDisplayMode(AUX\_RGB | AUX\_DEPTH | AUX\_DOUBLE);

if(auxInitWindow("v11\_03") == GL\_FALSE) {

auxQuit();

}

auxExposeFunc((AUXEXPOSEPROC)Reshape);

auxReshapeFunc((AUXRESHAPEPROC)Reshape);

auxIdleFunc(Turn);

auxKeyFunc(AUX\_LEFT, Key\_LEFT);

auxKeyFunc(AUX\_RIGHT, Key\_RIGHT);

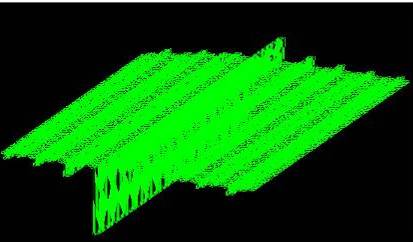
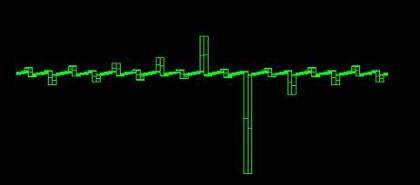
auxKeyFunc(AUX\_UP, Key\_UP);

auxKeyFunc(AUX\_DOWN, Key\_DOWN);

auxMainLoop(Turn);

}

**Результат:**



1. **Напишите программу вывода вращающего тора относительно своей оси x, с одновременным движением по следующей траектории z=sin(y)**

**Решение:**

//v11\_04

#include <windows.h>

#include <GL/gl.h>

#include <GL/glu.h>

#include <GL/glaux.h>

#include <math.h>

int flag=1;

void CALLBACK resize (int width, int height)

{

glViewport (0,0,width, height);

glMatrixMode (GL\_PROJECTION);

glLoadIdentity();

glOrtho(-15,15,-10,10, -10,20);

gluLookAt(1,-1,1, 0,0,0, 0,0,1);

glMatrixMode(GL\_MODELVIEW);

}

void CALLBACK display (void)

{

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

glColor3d(0,1,0);

float pos[4]={0,7.5,7.5,1};

static double time=0;

float y,z;

y=0;

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glLightfv(GL\_LIGHT0, GL\_POSITION, pos);

glPushMatrix();

y=float(time/100);

z=float(sin(y)); // float(); преобразование типа

//движением по следующей траектории z=sin(y)

glTranslated(0,y,z);

//вращающего тора относительно оси x

glRotated(time,1,0,0);

if (y>11 && flag==1)

{

flag=2;

}

if (y<-4 && flag==2)

{

flag=1;

}

if (flag==2)

{

time--;

}

else

{

time++;

}

auxSolidTorus(0.15, 0.5);

glPopMatrix();

// система координат XYZ

glBegin(GL\_LINES);

glColor3ub(255, 0, 0);

glVertex3f(5, 0, 0);

glVertex3f(-5, 0, 0);

// х красный

glColor3ub(0, 255, 0);

glVertex3f(0, 11, 0);

glVertex3f(0, -4, 0);

// y зеленый

glColor3ub(0, 0, 255);

glVertex3f(0, 0, 0);

glVertex3f(0, 0, 5);

// z синий

glEnd();

glFinish();

auxSwapBuffers();

}

void main ()

{

auxInitPosition (100,100,800,600);

auxInitDisplayMode (AUX\_RGB | AUX\_DEPTH | AUX\_DOUBLE);

auxInitWindow ("v11\_04");

glScaled(1.5, 1.5, 1.5);

glTranslated(0,-3,0);

auxIdleFunc (display);

auxReshapeFunc(resize);

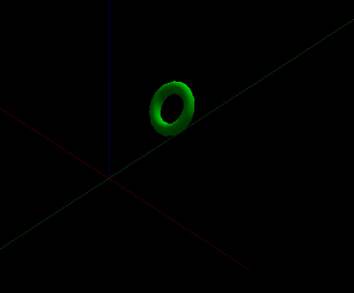
glEnable (GL\_DEPTH\_TEST);

glEnable (GL\_COLOR\_MATERIAL);

auxMainLoop(display);

}

**Результат:**



1. **Напишите программу мерцающего звездного неба**

**Решение:**

#include <windows.h>

#include <GL/gl.h>

#include <GL/glu.h>

#include <GL/glaux.h>

#include<time.h>

#include <stdio.h>

#include <iostream.h>

#include <conio.h>

#include <math.h>

int t\_sec() // считывание секунд с системных часов

{

struct tm \*newtime;

time\_t aclock;

time(&aclock);

newtime = localtime(&aclock);

asctime(newtime);

int s=newtime->tm\_sec;

return s;

}

void CALLBACK resize (int width, int height)

{

glViewport (0,0,width, height);

glMatrixMode (GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-100,100,-100,100);

glMatrixMode(GL\_MODELVIEW);

}

void CALLBACK Draw (void)

{

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

int c, x, y, p;

srand(t\_sec());

//задание стартовой точки

//для генерации случайных чисел

for(int i=0; i<1000; i++)

{

c=rand()%255; //генерация случайных чисел от 0 до 255

x=rand()%200; //генерация случайных чисел от 0 до 200

y=rand()%200; //генерация случайных чисел от 0 до 200

p=rand()%3;

glPointSize(p);

glBegin(GL\_POINTS);

glColor3ub(c,c,c);

glVertex2d(x,y);

glEnd();

}

auxSwapBuffers();

}

void main ()

{

auxInitPosition (100,100,800,800);

auxInitDisplayMode (AUX\_RGB | AUX\_DEPTH | AUX\_DOUBLE);

auxInitWindow ("v11\_06");

glTranslated(-100,-100,0);

auxIdleFunc (Draw);

auxReshapeFunc (resize);

glEnable (GL\_DEPTH\_TEST);

glEnable (GL\_COLOR\_MATERIAL);

auxMainLoop(Draw);

}

**Результат:**



1. **Изобразите радугу состоящую из полигонов, которая освещается прожектором. Направление света прожектора постоянно меняется и скользит вдоль радуги**

**Решение:**

#include <windows.h>

#include <GL/gl.h>

#include <GL/glu.h>

#include <GL/glaux.h>

#include<time.h>

#include <stdio.h>

#include <iostream.h>

#include <conio.h>

#include <math.h>

int flag=1;

void CALLBACK resize (int width, int height)

{

glViewport (0,0,width, height);

glMatrixMode (GL\_PROJECTION);

glLoadIdentity();

//glOrtho(-20,20,-15,15, -10,25);

glOrtho(-6,6,-6,6, -20,20);

gluLookAt(1,-1,1, 0,0,0, 0,0,1);

//gluLookAt(0,-5,0, 0,0,0, 0,0,1);

glMatrixMode(GL\_MODELVIEW);

}

void CALLBACK Draw (void)

{

static double time=0;

GLUquadricObj \*quadObj;

quadObj = gluNewQuadric();

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

glPushMatrix();

float x=time/100-2.2;

float z=pow((pow(2.4,2)-pow(x,2)),0.5);

if (x>2.2 && flag==1)

{

flag=2;

}

if (x<-2.2 && flag==2)

{

flag=1;

}

if (flag==2)

{

time--;

}

else

{

time++;

}

float dir[3]={x,0.1,z};

float pos[4]={0,-1,0,1};

// включение нулевой лампы

GLfloat light\_diffuse[]={3.0,3.0,3.0,1.0};

glLightfv(GL\_LIGHT0,GL\_DIFFUSE, light\_diffuse);

glLightfv(GL\_LIGHT0, GL\_POSITION, pos);

glLightfv(GL\_LIGHT0, GL\_SPOT\_DIRECTION, dir);

glLightf(GL\_LIGHT0, GL\_SPOT\_CUTOFF, 90);

glLightf(GL\_LIGHT0, GL\_SPOT\_EXPONENT, 9);

glEnable(GL\_LIGHTING);

glEnable (GL\_LIGHT0);

glEnable (GL\_DEPTH\_TEST);

glPopMatrix();

glPushMatrix();

glRotated(90,1,0,0);

gluQuadricDrawStyle(quadObj, GLU\_FILL);

// К

glColor3ub(255, 0, 0);

gluPartialDisk(quadObj, 2, 2.2, 30, 30, -90, 180);

// О

glColor3ub(255, 125, 0);

gluPartialDisk(quadObj, 2.2, 2.4, 30, 30, -90, 180);

// Ж

glColor3ub(255, 255, 0);

gluPartialDisk(quadObj, 2.4, 2.6, 30, 30, -90, 180);

// З

glColor3ub(0, 255, 0);

gluPartialDisk(quadObj, 2.6, 2.8, 30, 30, -90, 180);

// Г

glColor3ub(0, 255, 255);

gluPartialDisk(quadObj, 2.8, 3, 30, 30, -90, 180);

// С

glColor3ub(0, 0, 255);

gluPartialDisk(quadObj, 3, 3.2, 30, 30, -90, 180);

// Ф

glColor3ub(255, 0, 255);

gluPartialDisk(quadObj, 3.2, 3.4, 30, 30, -90, 180);

glPopMatrix();

glDisable(GL\_LIGHTING);

auxSwapBuffers();

}

void main ()

{

auxInitPosition (100,100,800,800);

auxInitDisplayMode (AUX\_RGB | AUX\_DEPTH | AUX\_DOUBLE);

auxInitWindow ("v11\_06");

glTranslated(3,-3,2);

glEnable (GL\_COLOR\_MATERIAL); // установка свойств материала

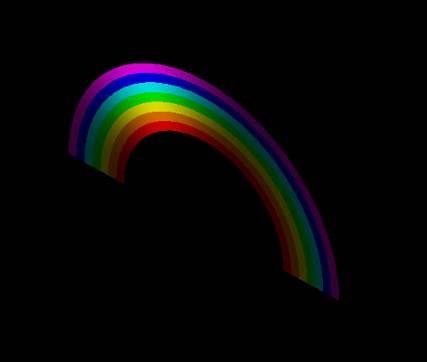
auxIdleFunc (Draw);

auxReshapeFunc (resize);

auxMainLoop(Draw);

}

**Результат:**



1. **Напишите программу вывода цилиндра, на котором наложена текстура. Создайте эффект сползания текстуры с цилиндра. Вокруг цилиндра вращается несколько полупрозрачных небольших сфер по спирали**

**Решение**

#include <windows.h>

#include <GL/gl.h>

#include <GL/glu.h>

#include <GL/glaux.h>

#include<time.h>

#include <stdio.h>

#include <iostream.h>

#include <conio.h>

#include <math.h>

int flag0=1;

int flag1=1;

int flag2=1;

int flag3=1;

unsigned int image1\_ID;

AUX\_RGBImageRec\* image1;

void CALLBACK resize (int width, int height)

{

glViewport (0,0,width, height);

glMatrixMode (GL\_PROJECTION);

glLoadIdentity();

glOrtho(-6,6,-6,6, -20,20);

gluLookAt(1,-1,1, 0,0,0, 0,0,1);

glMatrixMode(GL\_MODELVIEW);

}

void CALLBACK Draw (void)

{

static double time0=0;

static double time1=0;

static double time2=0;

static double time3=0;

GLUquadricObj \*quadObj;

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

glColor3d(1,1,1);

float pos[4]={3,-3,5,1};

glLightfv(GL\_LIGHT0, GL\_POSITION, pos);

glEnable(GL\_LIGHTING);

glEnable (GL\_LIGHT0);

glEnable (GL\_DEPTH\_TEST);

quadObj = gluNewQuadric();

float s=0.9-(time0/800);

if (s<0.01 && flag0==1)

{flag0=2;}

if (s>0.9 && flag0==2)

{flag0=1;}

if (flag0==1)

{time0++;}

else

{time0--;}

const float p[4]={0,0,s,0};

glPushMatrix();

glEnable(GL\_TEXTURE\_2D);

glEnable(GL\_TEXTURE\_GEN\_S);

glEnable(GL\_TEXTURE\_GEN\_T);

glTexGeni(GL\_S, GL\_TEXTURE\_GEN\_MODE, GL\_OBJECT\_LINEAR);

glTexGeni(GL\_T, GL\_TEXTURE\_GEN\_MODE, GL\_OBJECT\_LINEAR);

glTexGenfv(GL\_T,GL\_OBJECT\_PLANE,p);

glRotated(-180,1,0,0);

glTranslated(0,0,-3);

glColor3d(1,1,0);

gluCylinder(quadObj, 1, 1, 3, 50, 50);

glDisable(GL\_TEXTURE\_GEN\_S);

glDisable(GL\_TEXTURE\_2D);

glPopMatrix();

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);

glPushMatrix();

glEnable(GL\_ALPHA\_TEST);

glEnable(GL\_BLEND);

double z1=time1/300;

glRotated(time1,0,0,1);

glTranslated(1.3,0,1);

glTranslated(0,0,z1);

if (z1>2 && flag1==1)

{flag1=2;}

if (z1<-1 && flag1==2)

{flag1=1;}

glColor4d(0,1,0, 0.1);

auxSolidSphere(0.2); //s1

glDisable(GL\_BLEND);

glDisable(GL\_ALPHA\_TEST);

if (flag1==1)

{time1++;}

else

{time1--;}

glPopMatrix();

glPushMatrix();

glEnable(GL\_ALPHA\_TEST);

glEnable(GL\_BLEND);

double z2=time2/600;

glRotated(time2\*2,0,0,15);

glTranslated(1.8,0,1);

glTranslated(0,0,z2);

if (z2>2 && flag2==1)

{flag2=2;}

if (z2<-1 && flag2==2)

{flag2=1;}

glColor4d(1,0,0, 0.1);

auxSolidSphere(0.2); //s2

glDisable(GL\_BLEND);

glDisable(GL\_ALPHA\_TEST);

if (flag2==1)

{time2++;}

else

{time2--;}

glPopMatrix();

glPushMatrix();

glEnable(GL\_ALPHA\_TEST);

glEnable(GL\_BLEND);

double z3=time3/400;

glRotated(time3\*3,0,0,1);

glTranslated(2.3,0,1);

glTranslated(0,0,z3);

if (z3>2 && flag3==1)

{flag3=2;}

if (z3<-1 && flag3==2)

{flag3=1;}

glColor4d(0,0,1, 0.2);

auxSolidSphere(0.2); //s3

glDisable(GL\_BLEND);

glDisable(GL\_ALPHA\_TEST);

if (flag3==1)

{time3++;}

else

{time3--;}

glPopMatrix();

gluDeleteQuadric(quadObj);

auxSwapBuffers();

}

void main ()

{

auxInitPosition (100,100,800,800);

auxInitDisplayMode (AUX\_RGB | AUX\_DEPTH | AUX\_DOUBLE);

auxInitWindow ("v11\_07");

glTranslated(0,0,-1);

glEnable (GL\_COLOR\_MATERIAL); // установка свойств материала

image1 = auxDIBImageLoad("v11\_07.bmp");

glGenTextures(1, &image1\_ID);

glBindTexture(GL\_TEXTURE\_2D, image1\_ID);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glTexImage2D(GL\_TEXTURE\_2D, 0, 3,

image1->sizeX,

image1->sizeY,

0, GL\_RGB, GL\_UNSIGNED\_BYTE,

image1->data);

auxIdleFunc (Draw);

auxReshapeFunc (resize);

auxMainLoop(Draw);

}

**Результат:**

