Министерство образования и науки Российской Федерации Федеральное агентство по образованию Федеральное государственное бюджетное образовательное учреждение высшего образования «Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Лабораторная работа №7

по курсу «Компьютерная графика»

# «Аффинные преобразования в пространстве»

Выполнил студент группы ИВТ-21\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Птахова А.М/

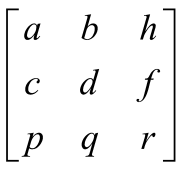
Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Коржавина А.С./

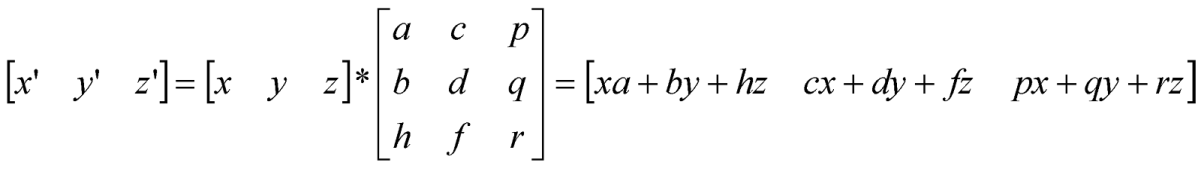
Киров 2021

**Цель работы:** закрепить лекционный материал по изучению материала одноименной темы, реализовав матрицы переноса, масштабирования, отражения и вращения применительно к координатам описанной в программе объемной фигуры (многогранника) с целью демонстрации движения и преобразования формы этой фигуры в пространстве.

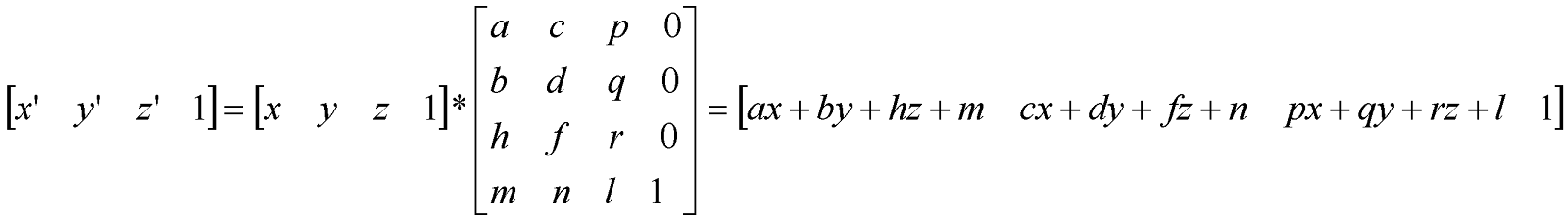
**Краткие теоретические сведения.**

Пусть дана тройка векторов a,b,c с общим началом в точке О (начале координат), не лежащих в одной плоскости. Относительно точки О положение любой точки M в пространстве характеризуется ее радиус - вектором r. Если принять векторы a,b,c за базис и представить r=l\*a+w\*b+v\*c, то положение точки М характеризуется набором чисел (l,w,v), называемых аффинными координатами точки М, т.е.координатами ее радиус-вектора r. Если базис (a,b,c) - единичный и взаимно перпендикулярный, то система координат - декартова. Тогда основные векторы (a,b,c) принято обозначать как i,j,k- декартов базис, координаты - (x,y,z), r=x\*i+y\*j+z\*k.

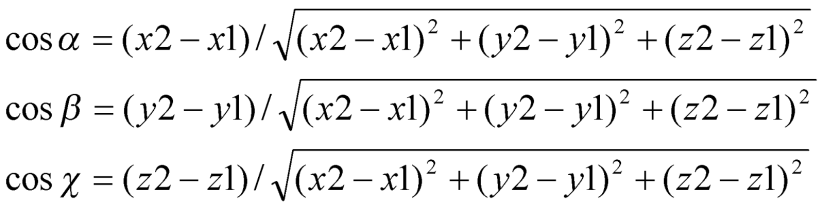
Как и на плоскости следует применять геометрические преобразования к точкам изображения. Точка в пространстве задается либо радиус-вектором r; либо матрицами координат (x y z). Аффинные преобразования пространства определяются системой координат Oxyz, матрицей коэффициентов:  и числами m,n,l ,осуществляющими перенос изображения.

Если новая и старая системы координат имеют одно начало, преобразования в матричной форме выглядят так: 

В противном случае целесообразно прибегнуть к однородным координатам:

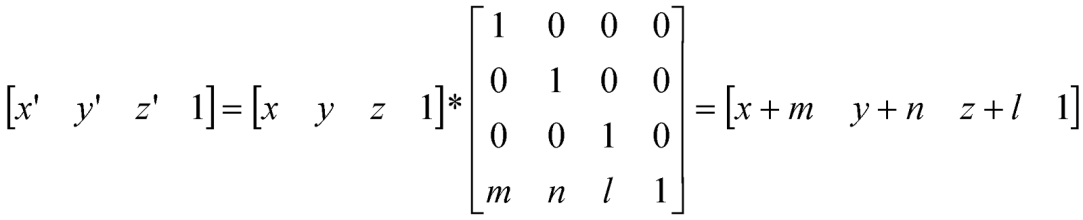


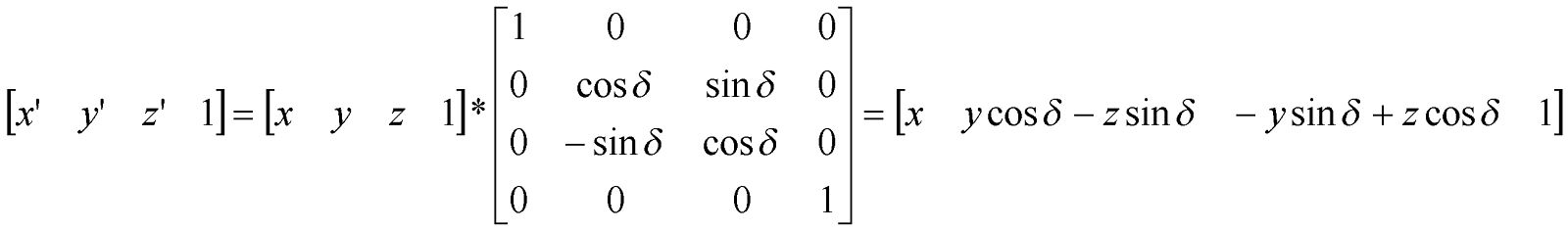
Направляющие косинусы вектора - это косинусы углов, которые он образует с осями координат. Они полностью определяют направление вектора. Если вектор а=ax\*i+ay\*j+az\*k, то ax -проекция вектора a на ось x, умноженная на косинус угла между векторами и осью x. Аналогично для ay и az. Если вектор а задан координатами начальной и конечной точек (x1,y1) и (x2,y2), направляющие косинусы определяются уравнениями:

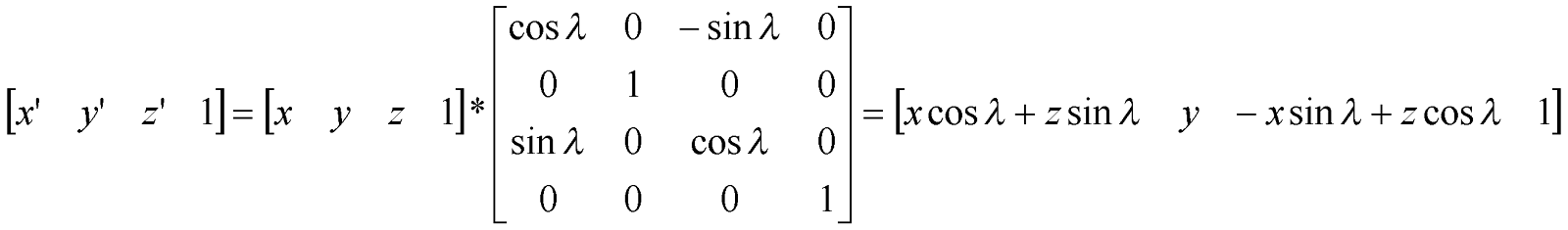


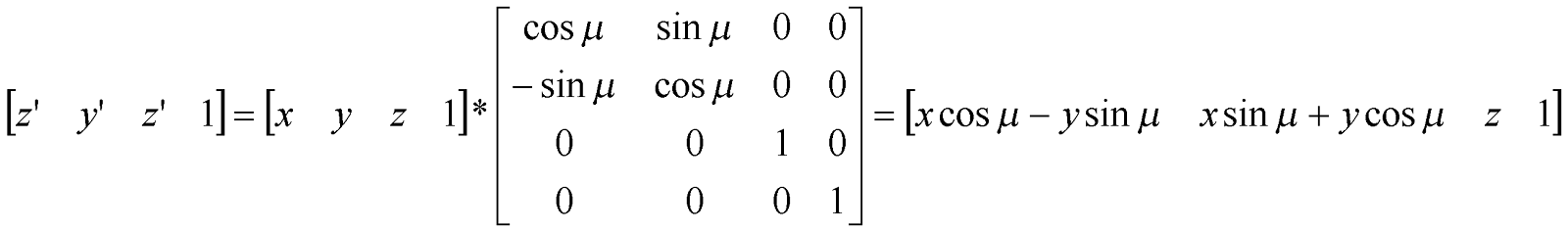
где α,β,γ - углы, которые вектор а образует с осями Ох, Оy, Oz, соответственно.

Частные случаи аффинных преобразований:

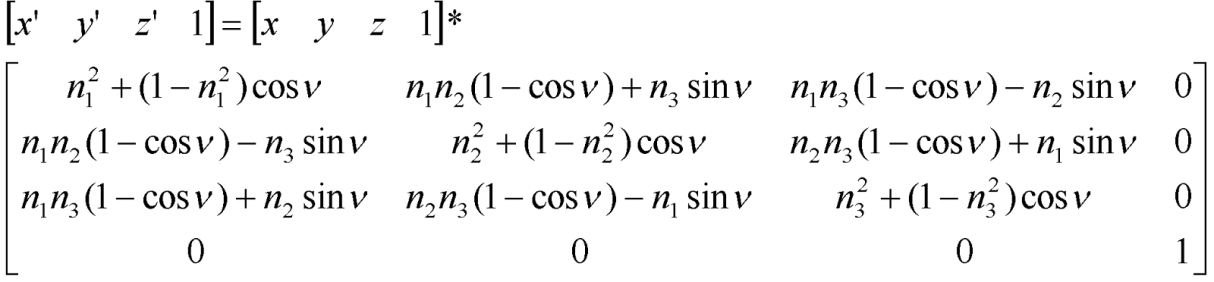
1.Сдвиг - перенос точки (x,y,z) на m единиц по координате x, на n - по y, на l единиц- по z.: 

2.Повоpот точки (x,y,z) вокpуг оси абсцисс на угол δ: 

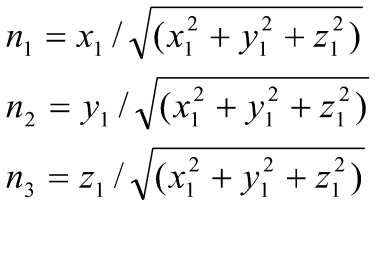
вокруг оси ординат на угол λ: 

вокруг оси аппликат на угол μ: 

вокруг оси, проходящей через начало координат на угол ν

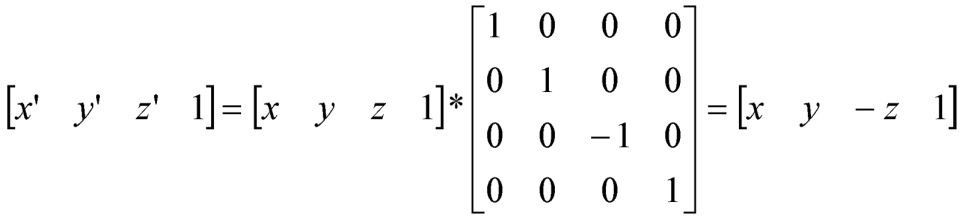
.

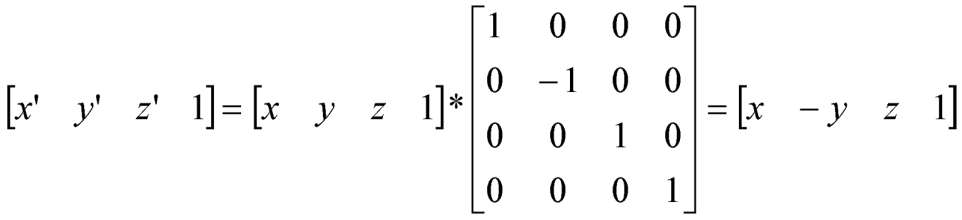
 Здесь n1,n2,n3 - направляющие косинусы вращения. Если ось вращения проходит через начало координат и точку (x1,y1, z1) то

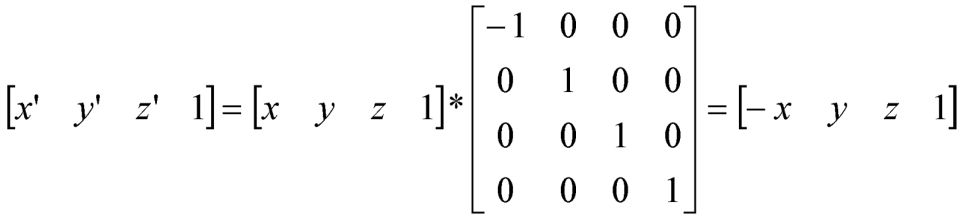


Вращение вокруг произвольной оси осуществляется переносом ее вместе с изображением в начало координат, вращением вокруг перенесенной оси и обратным переносом изображения в исходное положение. Итоговая матрица будет определена умножением соответствующих простых матриц.

3. Симметрия относительно координатных плоскостей осуществляет зеркальное отображение 3D-изображения.

Относительно плоскости XOY 

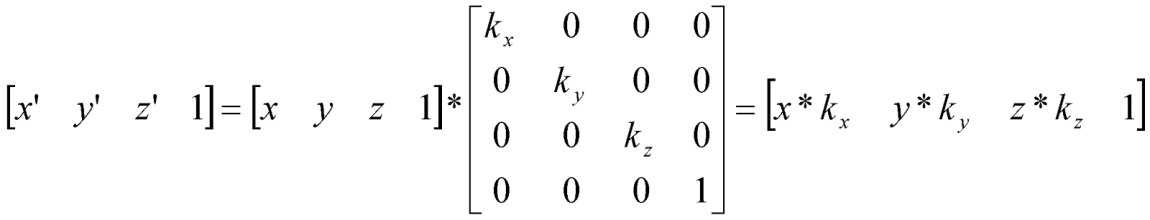
Плоскости XOZ - -

Плоскости YOZ :  -

Симметрия относительно произвольной плоскости - это сложное  преобразование, осуществляемое из простых поэтапно:

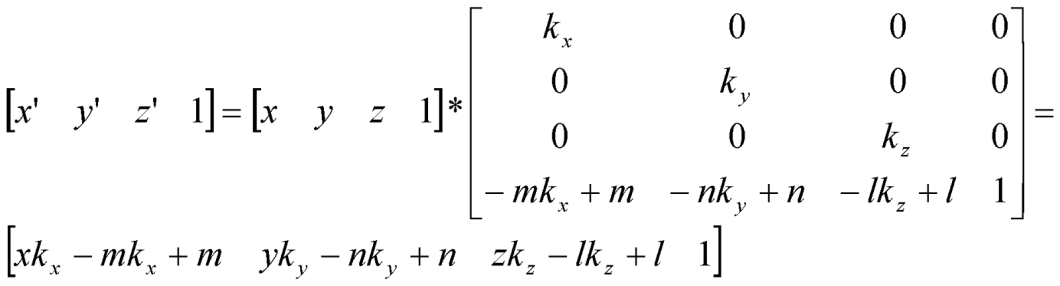
* совмещение плоскости симметрии с одной из координатных
* отражение относительно этой координатной плоскости
* обратное преобразование, возвращающее плоскость симметрии в исходное состояние.

4.Масштабирование осуществляется диагональными элементами матрицы преобразования.

Относительно начала координат: ,

где kx, ky, kz - коэффициенты искажения вдоль осей ox,oy,oz, соответственно.

Относительно произвольного центра с координатами (m,n,l):

 -

Любое другое преобразование может быть представлено суперпозицией вращений, переносов, отражений, растяжений (сжатий).

**Разработка программы**

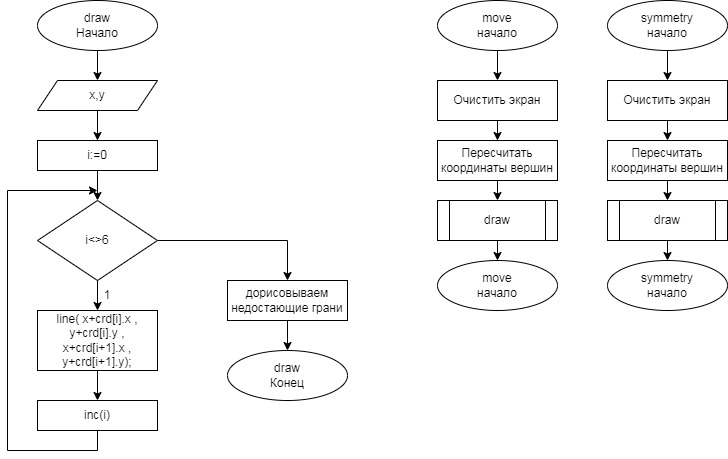
В программе реализованы возможности передвижения объекта при помощи стрелок и кнопок Page Up и Page Down, отражение относительно разных осей при помощи кнопок 1-3, вращения с использованием различных неподвижных точек (сторон) при помощи кнопок 4-9, изменения размеров, возвращения к первоначальному виду( ctrl+R), анимации вращения (Ctrl+F), анимации изменения размеров (Ctrl+G).

**Вывод:**

В ходе лабораторной работы были получены навыки работы с аффинными преобразованиями в пространстве для выполнения различных действий над 3х-мерным объектом.

Приложение А

Схемы алгоритмов



Приложение Б

Листинг

**Procedure** DrawAxis(x,y:integer;p1,f1:real);

**var** a:**array**[0..3] **of** Tcrd;

p,f:real;

i:integer;

**begin**

setcolor(5);

p:=p1\*pi/180;

f:=f1\*pi/180;

**for** i:=0 **to** 3 **do**

**begin**

a[i].x:=round( Axis[i].x\*cos(p)+Axis[i].z\*sin(p) );

a[i].y:=round(

Axis[i].x\*sin(p)\*sin(f)+

Axis[i].y\*cos(f)-

Axis[i].z\*sin(f)\*cos(p)

);

**end**;

line(x+a[0].x , y+a[0].y , x+a[1].x , y+a[1].y);

line(x+a[0].x , y+a[0].y , x+a[2].x , y+a[2].y);

line(x+a[0].x , y+a[0].y , x+a[3].x , y+a[3].y);

setcolor(11);

**end**;

**Procedure** draw(x,y:integer);

**var** i:integer;

**begin**

**for** i:=0 **to** 6 **do**

**begin**

line( x+crd[i].x , y+crd[i].y , x+crd[i+1].x , y+crd[i+1].y);

**end**;

line( x+crd[0].x , y+crd[0].y , x+crd[3].x , y+crd[3].y );

line( x+crd[0].x , y+crd[0].y , x+crd[7].x , y+crd[7].y );

line( x+crd[1].x , y+crd[1].y , x+crd[6].x , y+crd[6].y );

line( x+crd[2].x , y+crd[2].y , x+crd[5].x , y+crd[5].y );

line( x+crd[4].x , y+crd[4].y , x+crd[7].x , y+crd[7].y );

DrawAxis(x,y,q,w);

**end**;

**procedure** Axonometric(p1,f1:real);

**var** p,f:real;

i:integer;

**begin**

ClearDevice;

p:=p1\*pi/180;

f:=f1\*pi/180;

**for** i:=0 **to** 7 **do**

**begin**

crd[i].x:=round( v[i].x\*cos(p)+v[i].z\*sin(p) );

crd[i].y:=round(

v[i].x\*sin(p)\*sin(f)+

v[i].y\*cos(f)-

v[i].z\*sin(f)\*cos(p)

);

**end**;

draw(500,500);

**end**;

**procedure** Move(m,n,l:integer);

**var** i:integer;

**begin**

**for** i:=0 **to** 7 **do**

**begin**

v[i].x:=v[i].x+m;

v[i].y:=v[i].y+n;

v[i].z:=v[i].z+l;

**end**;

**end**;

**procedure** Symmetry(m,n,l:integer);

**var** i:integer;

**begin**

**for** i:=0 **to** 7 **do**

**begin**

v[i].x:=m\*v[i].x;

v[i].y:=n\*v[i].y;

v[i].z:=l\*v[i].z;

**end**;

**end**;

**procedure** Scaling(k1,k2,k3:real;p:TXYZ);

**var** i:integer;

m,n,l:integer;

**begin**

m:=p.x;

n:=p.y;

l:=p.z;

**for** i:=0 **to** 7 **do**

**begin**

v[i].x:=round(k1\*v[i].x -m\*k1+m);

v[i].y:=round(k2\*v[i].y -n\*k2+n);

v[i].z:=round(k3\*v[i].z -l\*k3+l);

**end**;

**end**;

**procedure** rotationAxisX(angle:integer);

**var** i:integer;

r:real;

**begin**

b:=v;

r:=angle\*Pi/180;

**for** i:=0 **to** 7 **do**

**begin**

v[i].y:=round(b[i].y\*cos(r)-b[i].z\*sin(r));

v[i].z:=round(b[i].y\*sin(r)+b[i].z\*cos(r));

**end**;

**end**;

**procedure** rotationAxisY(angle:integer);

**var** i:integer;

r:real;

**begin**

b:=v;

r:=angle\*Pi/180;

**for** i:=0 **to** 7 **do**

**begin**

v[i].x:=round(b[i].x\*cos(r)+b[i].z\*sin(r));

v[i].z:=round(-b[i].x\*sin(r)+b[i].z\*cos(r));

**end**;

**end**;

**procedure** rotationAxisZ(angle:integer);

**var** i:integer;

r:real;

**begin**

b:=v;

r:=angle\*Pi/180;

**for** i:=0 **to** 7 **do**

**begin**

v[i].x:=round(b[i].x\*cos(r)-b[i].y\*sin(r));

v[i].y:=round(b[i].x\*sin(r)+b[i].y\*cos(r));

**end**;

**end**;

**procedure** rotation(n1,n2,n3:real;angle:integer);

**var** i:integer;

r:real;

**begin**

b:=v;

r:=angle\*Pi/180;

**for** i:=0 **to** 7 **do**

**begin**

v[i].x:=round(

b[i].x\*(n1\*n1+(1-n1\*n1)\*cos(r))+

b[i].y\*( n1\*n2\*(1-cos(r))-n3\*sin(r)) +

b[i].z\*( n1\*n3\*(1-cos(r))+n2\*sin(r))

);

v[i].y:=round(

b[i].x\*(n1\*n2\*(1-cos(r))+n3\*sin(r))+

b[i].y\*( n2\*n2+(1-n2\*n2)\*cos(r) ) +

b[i].z\*( n2\*n3\*(1-cos(r))-n1\*sin(r))

);

v[i].z:=round(

b[i].x\*( n1\*n3\*(1-cos(r))-n2\*sin(r)) +

b[i].y\*( n2\*n3\*(1-cos(r))+n1\*sin(r)) +

b[i].z\*( n3\*n3+(1-n3\*n3)\*cos(r) )

);

**end**;

**end**;

**procedure** rotationDiagonal(n:integer;angle:integer);

**begin**

rotation(

v[n].x/sqrt(sqr(v[n].x)+sqr(v[n].y)+sqr(v[n].z)),

v[n].y/sqrt(sqr(v[n].x)+sqr(v[n].y)+sqr(v[n].z)),

v[n].z/sqrt(sqr(v[n].x)+sqr(v[n].y)+sqr(v[n].z)),

angle

);

**end**;

**procedure** animation();

**var** i:integer;

t:**array**[0..7] **of** TXYZ;

**begin**

t:=v;

**for** i:=1 **to** 90 **do**

**begin**

rotationDiagonal(4,4);

Axonometric(q,w);

delay(1);

**end**;

v:=t;

Axonometric(q,w);

**end**;

**procedure** animationScaling();

**var** i:integer;

t:**array**[0..7] **of** TXYZ;

**begin**

t:=v;

**for** i:=1 **to** 90 **do**

**begin**

**if**(i<45)**then**

**begin**

Scaling(0.95,0.95,0.95,v[1]);

**end**

**else**

**begin**

Scaling(1.05,1.05,1.05,v[1]);

**end**;

Axonometric(q,w);

delay(1);

**end**;

v:=t;

Axonometric(q,w);

**end**;

Приложение В

Экранные формы

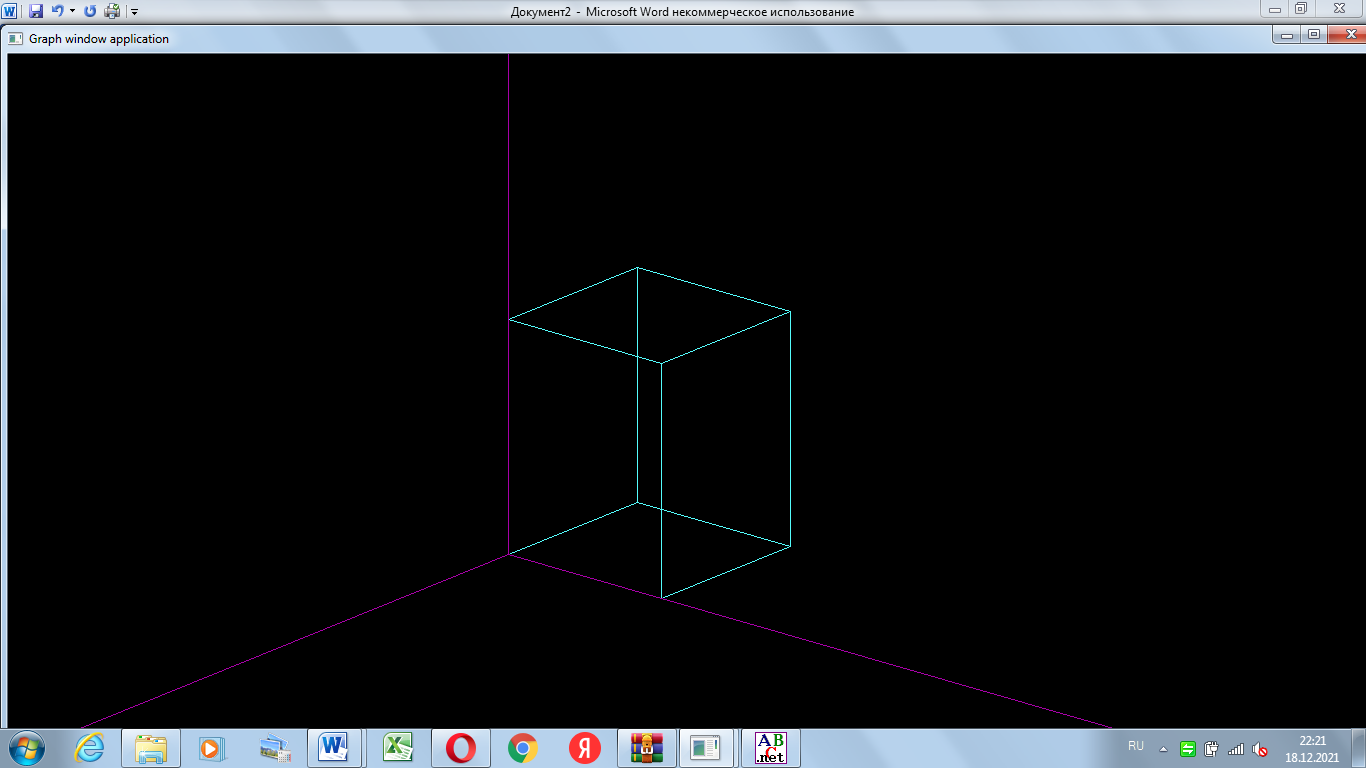


Рисунок 1 – начальное расположение объекта

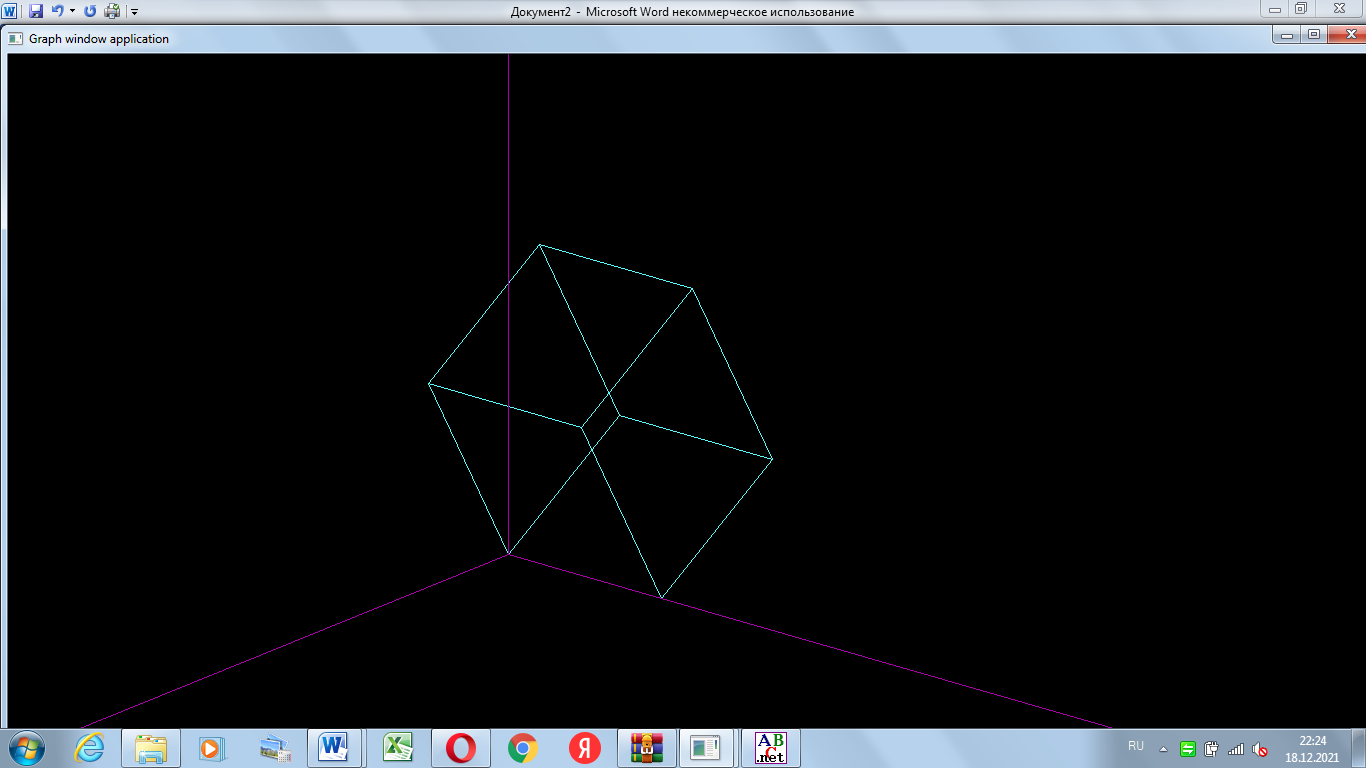


Рисунок 2 – поворот объекта