Translation:

$$\begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & v \\ 0 & 1 & 0 & w \\ 0 & 0 & 1 & u \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Scaling:

$$\begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation around the X axis:

$$\begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation around the Y axis::

$$\begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation around the Z axis::

$$\begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

3D TRANSFORMATIONS

(x, y, z) ⟶ (a, b,c)

matrix formulas

---

1) Translation by vector (v, w, u)
2) Scaling on a scale (sx, sy, sz)
3) Angle rotation`α

4-line matrices in the so-called homogeneous recording are used to determine images of transformations.

https://pl.wikipedia.org/wiki/Wsp%C3%B3%C5%82rz%C4%99dne_jednorodne

The point which has coordinates (x, y, z) is represented in a homogeneous form (x, y, z, 1).

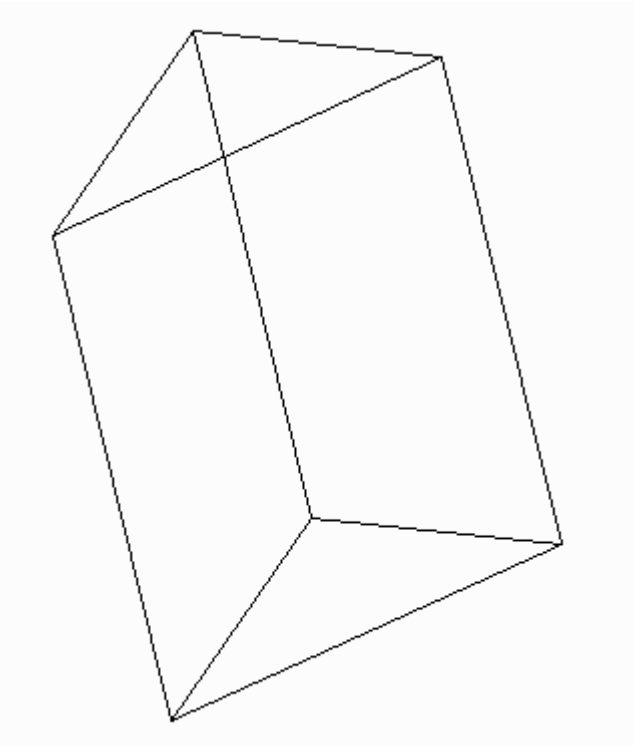
Application example: 3D rotation (Pascal code in graphic mode)

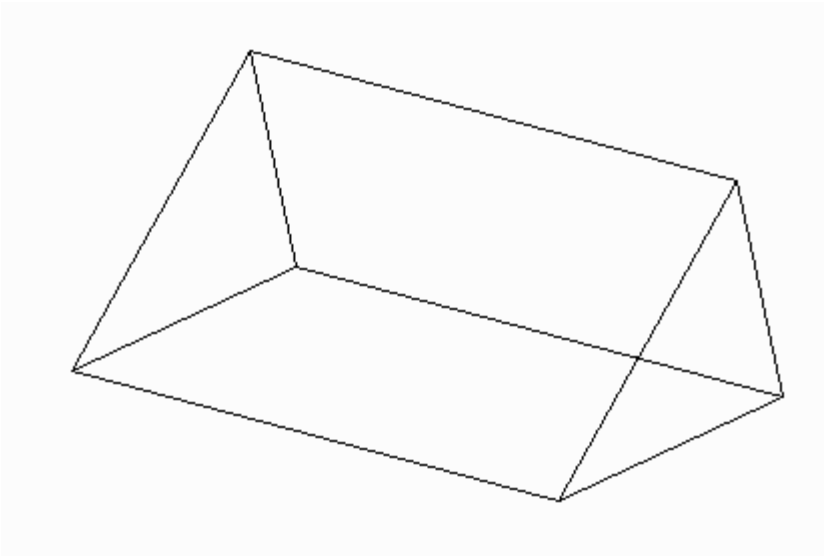
Instead of the **Crt** module, the **WinCrt** library was used to ensure proper operation of the **KeyPressed** condition. It is necessary to use the standard **Graph** module.

The program allows you to rotate the triangular prism using the arrow keys.

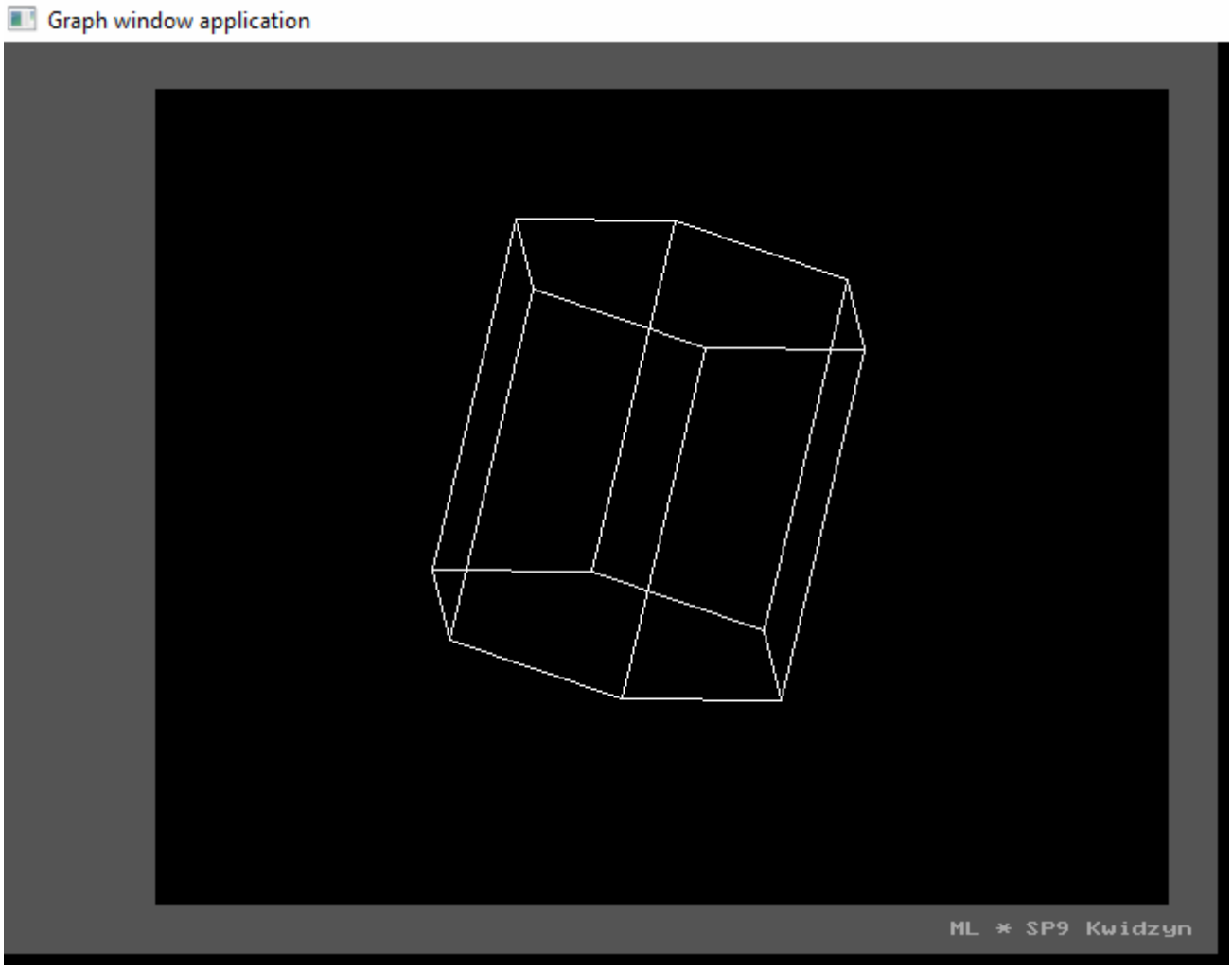
The display starts after pressing the <ENTER > or <SPACE> key.

Exit the program after pressing the <ESC> key.





Minor modifications to the program regarding the number of vertices and starting coordinates of the XYZ system allow to obtain drawings of other polyhedra with the possibility of their rotation.




```
program solid;
uses WinCrt,Graph;
type
matrix=array[1..4,1..4] of real;
typ=array [0..6] of matrix;
var
character :char;
code :integer;
rotatex,rotatey,rotatez :integer;
xp,yp,zp :real;
figure :typ;
T,Rx,Ry,Rz,Skala,A,B :matrix;


{***************** BUFFER *****************}
procedure buffer;
begin
while keypressed do character:=readkey;
end;
```

```
{********** PRODUCT OF THE MATRIX ************}
procedure product(m,n,p :integer;A,B :matrix;var C :matrix);
var i,j,k :integer;
suma :real;
begin
for i:=1 to m do
for j:=1 to p do
begin
suma:=0;
for k:=1 to n do
suma:=suma+A[i,k]*B[k,j];
C[i,j]:=suma;
end;
end;

{********** EDGE ************}
procedure edge(var fig3d :typ; n1,n2 :byte;x,y,kolor :integer);
var x1,y1,x2,y2 :integer;
begin
setcolor(kolor);
x1:=round(fig3d[n1][1,1]);
y1:=round(fig3d[n1][2,1]);
x2:=round(fig3d[n2][1,1]);
y2:=round(fig3d[n2][2,1]);
line(x1+x,y1+y,x2+x,y2+y);
end;

{********** ROTATE X *************}
procedure mac_Rx(alfa: real);
var i,j :integer;
angle :real;
begin
for i:=1 to 4 do
begin
for j:=1 to 4 do Rx[i,j]:=0;
Rx[i,i]:=1;
end;
angle:=alfa*pi/180;
Rx[2,2]:=cos(angle);
Rx[3,3]:=Rx[2,2];
Rx[2,3]:=sin(angle);
Rx[3,2]:=-Rx[2,3];
end;

{********** ROTATE Y *************}
procedure mac_Ry(alfa: real);
var i,j :integer;
angle :real;
begin
for i:=1 to 4 do
begin
for j:=1 to 4 do Ry[i,j]:=0;
Ry[i,i]:=1;
end;
```

```
angle:=alfa*pi/180;
Ry[1,1]:=cos(angle);
Ry[3,3]:=Ry[1,1];
Ry[3,1]:=sin(angle);
Ry[1,3]:=-Ry[3,1];
end;

{********** ROTATE Z *************}
procedure mac_Rz(alfa: real);
var i,j :integer;
angle :real;
begin
for i:=1 to 4 do
begin
for j:=1 to 4 do Rz[i,j]:=0;
Rz[i,i]:=1;
end;
angle:=alfa*pi/180;
Rz[1,1]:=cos(angle);
Rz[2,2]:=Rz[1,1];
Rz[1,2]:=sin(angle);
Rz[2,1]:=-Rz[1,2];
end;

{********** COORDINATES *************}
procedure coord(var fig3d :typ; n :byte; x,y,z :real);
begin
fig3d[n][1,1]:=x;
fig3d[n][2,1]:=y;
fig3d[n][3,1]:=z;
fig3d[n][4,1]:=1;
end;

{********** STARTING COORDINATES ********}
procedure coord_start;
begin
coord(figure,0,-40,-140,-100);
coord(figure,1,110,-140,-100);
coord(figure,2,0,-140,100);
coord(figure,3,-40,140,-100);
coord(figure,4,110,140,-100);
coord(figure,5,0,140,100);
coord(figure,6,0,-140,-100);
end;

{*************** DRAW **********}
procedure draw(x,y :integer);
var i,j :integer;
begin
edge(figure,0,1,x,y,0);
edge(figure,1,2,x,y,0);
edge(figure,2,0,x,y,0);
edge(figure,3,4,x,y,0);
edge(figure,4,5,x,y,0);
edge(figure,5,3,x,y,0);
```

```
edge(figure,0,3,x,y,0);
edge(figure,1,4,x,y,0);
edge(figure,2,5,x,y,0);
end;

{************** GRAPHIC *************}
procedure graphic;
var karta,tryb :integer;
begin
karta:=detect;
InitGraph(karta,tryb,'');
end;

{************* IMAGE ***********}
procedure image;
var k: integer;
begin
bar(80,25,614,454);
draw(340,240);
delay(10);
for k:=0 to 6 do
begin
product(4,4,1,B,figure[k],A);figure[k]:=A;
end;
delay(100);
end;

{************** MAIN ************}
begin
coord_start;
graphic;
cleardevice;
setfillstyle(1,15);
REPEAT
buffer;
repeat
mac_Rz(0);
mac_Ry(0);
mac_Rx(0);
product(4,4,4,Ry,Rx,B);
repeat
until keypressed;
code:=ord(readkey);
until ((code=75) or (code=72) or (code=77) or (code=80)
or (code=13) or (code=32) or (code=27));
case code of
72: begin
rotatex:=5;
mac_Rx(rotatex);
product(4,4,4,Ry,Rx,B);
image;
end;
```

```
80: begin
rotatex:=-5;
mac_Rx(rotatex);
product(4,4,4,Ry,Rx,B);
image;
end;
77: begin
rotatey:=5;
mac_Ry(rotatey);
product(4,4,4,Ry,Rx,B);
image;
end;
75: begin
rotatey:=-5;
mac_Ry(rotatey);
product(4,4,4,Ry,Rx,B);
image;
end;
27: begin
cleardevice;
halt;
end;
end; {case}
image;
UNTIL 1=2;
READLN;
CloseGraph;
End.
```