

Tutorial 4: Solid Objects, Rotation, Translation

by Victor Saar

Content

The contents of this tutorial are solid objects and the rotation and translation of them. We do these operations using the Direct3DX utility library. Solid objects are nothing more than many single polygons, which lie next to each other in 3D space. We will build a pyramide out of four triangles and one quad for the basis. Then we rotate it and translate it into the screen so that we can see it. Using the Direct3DX utility library for that makes it easy. For each rotation and translation we need one matrix. We create these matrices with the Direct3DX functions. Then we have to multiply all matrices together and send it to the device. That's all what we do in this tutorial. We need no more files or functions.

main.cpp

Here we have some new variables and change the two arrays, which hold the vertices for the polygons. The new variables are three matrices for the rotation and translation. Each matrix will hold the information for one transformation. At the end we multiply all three and set the resulting matrix as world matrix. The float variable fRotation holds the current rotation angle of the pyramide. After that we have an array with four triangles for the top area. The second array is a quad for the basis. This time we set the vertices around the origin in model space, because we translate them into the world space later. The rest outside the main loop is not changed.

```
D3DXMATRIX matRotationX,
            matRotationY,
            matTranslation; //transformation matrices (NEW)
float fRotation = 0.0f; //rotation angle (NEW)
D3DVERTEX aTriangle[] = {{ 0.0f, 1.0f, 0.0f,0xffff0000}, //1. tri (NEW)
                          {-1.0f,-1.0f,-1.0f,0xff00ff00},
                          { 1.0f,-1.0f,-1.0f,0xff0000ff},
                          { 0.0f, 1.0f, 0.0f,0xffff0000}, //2. tri (NEW)
                          {-1.0f,-1.0f, 1.0f,0xff0000ff},
                          {-1.0f,-1.0f,-1.0f,0xff00ff00},
                          { 0.0f, 1.0f, 0.0f,0xffff0000}, //3. tri (NEW)
                          { 1.0f,-1.0f, 1.0f,0xff00ff00},
                          {-1.0f,-1.0f, 1.0f,0xff0000ff},
                          { 0.0f, 1.0f, 0.0f,0xffff0000}, //4. tri (NEW)
                          { 1.0f,-1.0f,-1.0f,0xff0000ff},
                          { 1.0f,-1.0f, 1.0f,0xff00ff00}};

D3DVERTEX aQuad[] = {{-1.0f,-1.0f,-1.0f,0xffffffff00}, //1 quad (NEW)
                    { 1.0f,-1.0f,-1.0f,0xffffffff00},
                    {-1.0f,-1.0f, 1.0f,0xffffffff00},
                    { 1.0f,-1.0f, 1.0f,0xffffffff00}};
```

Now we have to change the main loop. First we create the matrices for the transformation by using the Direct3DX matrix functions. We need two rotation matrices and one translation matrix. We multiply all three matrices and set the result as world matrix. We have to take care of the order, in which we multiply. First the rotation and then the translation.

```
if(g_App.CheckDevice())
{
    g_App.GetDevice()->Clear(0,
                            NULL,
                            D3DCLEAR_TARGET | D3DCLEAR_ZBUFFER,
                            D3DCOLOR_XRGB(0,0,0),
```

```
                1.0f,  
                0);  
g_App.GetDevice()->BeginScene();  
  
//x rotation (NEW)  
D3DXMatrixRotationX(&matRotationX,fRotation);  
//y rotation (NEW)  
D3DXMatrixRotationY(&matRotationY,fRotation * 0.75f);  
//translation (NEW)  
D3DXMatrixTranslation(&matTranslation,0.0f,0.0f,10.0f);  
  
//result as world matrix (NEW)  
g_App.GetDevice()->SetTransform(D3DTS_WORLD,  
                                &(matRotationX *  
                                matRotationY *  
                                matTranslation));
```

Here we just set the buffers and draw the pyramid. At the end of the main loop we increase the rotation angle.

```
g_App.GetDevice()->SetStreamSource(0,pTriangleVB,0,sizeof(D3DVERTEX));  
g_App.GetDevice()->DrawPrimitive(D3DPT_TRIANGLELIST,0,4);  
  
g_App.GetDevice()->SetStreamSource(0,pQuadVB,0,sizeof(D3DVERTEX));  
g_App.GetDevice()->DrawPrimitive(D3DPT_TRIANGLESTRIP,0,2);  
  
g_App.GetDevice()->EndScene();  
g_App.GetDevice()->Present(NULL,NULL,NULL,NULL);  
  
fRotation += 0.001f; //increase angle (NEW)  
}
```