

# Tutorial 7: Vertex Alpha

by Victor Saar

## Content

This tutorial covers a nice effect. We want to make the polygons of the pyramide transparent with alpha blending. We will use the alpha value of the vertex color. To do this we have to add a diffuse color value to the vertex structure again. Then we set the alpha value to 50%, so that the texture will be half transparent. Before we can use alpha blending we have to change the render states. We enable alpha blending and deactivate the depth buffer, because the hidden surfaces are now also visible and they wont be drawn with an activated depth buffer.

Before we can start we have to clear how alpha blending in Direct3D works. When you use alpha blending you can see a part of the texture color and also a part of the background color on the screen. Now we have to define how much of each color we want to take for the final color. This is done with the source blend factor and the destination blend factor. The source blend factor defines how much of the texture color we use for the final color. The destination blend factor does the same with the background color. The final color on the screen is calculated as follows:

$$\text{FinalColor} = \text{TexelColor} * \text{SourceBlendFactor} + \text{PixelColor} * \text{DestBlendFactor}$$

## d3ddefs.h

First we have to add the diffuse color value to the vertex structure. We change the definition of the vertex format by adding `D3DFVF_DIFFUSE`. Then we add a `DWORD` variable to the structure, which will hold the color.

```
//diffuse color added (NEW)
#define D3DFVF_CUSTOMVERTEX (D3DFVF_XYZ |
                             D3DFVF_NORMAL |
                             D3DFVF_DIFFUSE |
                             D3DFVF_TEX1)

//structures
struct D3DVERTEX
{
    float    fX,
            fY,
            fZ;
    D3DVECTOR Normal;
    DWORD    dwColor; //DWORD for color (NEW)
    float    fU,
            fV;
};
```

## application.cpp

Here we have to change a little bit more. First we deactivate the depth buffer.

```
//depth buffer deactivated (NEW)
m_pDirect3DDevice->SetRenderState(D3DRS_ZENABLE,D3DZB_FALSE);
```

Now we add some render states. The first enables alpha blending. The next two functions define how we want the alpha blending to be done. We set the source blend render state to source alpha

(D3DBLEND\_SRCALPHA), which means the alpha value is taken from the source, which is in our case the vertex color. Then we specify that the destination blend factor is one minus the source blend factor (D3DBLEND\_INVSRCALPHA), so that in the end the source and destination blend factor are both 0.5.

```
//alpha blending enabled (NEW)
m_pDirect3DDevice->SetRenderState(D3DRS_ALPHABLENDENABLE, true);
//source alpha (NEW)
m_pDirect3DDevice->SetRenderState(D3DRS_SRCBLEND, D3DBLEND_SRCALPHA);
//destination alpha (NEW)
m_pDirect3DDevice->SetRenderState(D3DRS_DESTBLEND, D3DBLEND_INVSRCALPHA);
```

The last thing is a texture stage state, which specifies that the alpha value is taken from the vertex color.

```
m_pDirect3DDevice->SetTextureStageState(0, D3DTSS_ALPHAARG1, D3DTA_DIFFUSE);
```

## main.cpp

In this file we only have to add the color information to the vertices. The seventh value is the diffuse color. We set the color to white with an alpha value of `0x7f`, which is 50%. The first two bits in a color value specify the alpha value in hexadecimal.

```
D3DVERTEX aTriangle[] = {
    { 0.0f, 1.0f, 0.0f, 0.0f, 1.0f, -1.0f, 0x7fffffff, 0.5f, 0.0f },
    { -1.0f, -1.0f, -1.0f, 0.0f, 1.0f, -1.0f, 0x7fffffff, 0.0f, 1.0f },
    { 1.0f, -1.0f, -1.0f, 0.0f, 1.0f, -1.0f, 0x7fffffff, 1.0f, 1.0f },
    { 0.0f, 1.0f, 0.0f, -1.0f, 1.0f, 0.0f, 0x7fffffff, 0.5f, 0.0f },
    { -1.0f, -1.0f, 1.0f, -1.0f, 1.0f, 0.0f, 0x7fffffff, 0.0f, 1.0f },
    { -1.0f, -1.0f, -1.0f, -1.0f, 1.0f, 0.0f, 0x7fffffff, 1.0f, 1.0f },
    { 0.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, 0x7fffffff, 0.5f, 0.0f },
    { 1.0f, -1.0f, 1.0f, 0.0f, 1.0f, 1.0f, 0x7fffffff, 0.0f, 1.0f },
    { -1.0f, -1.0f, 1.0f, 0.0f, 1.0f, 1.0f, 0x7fffffff, 1.0f, 1.0f },
    { 0.0f, 1.0f, 0.0f, 1.0f, 1.0f, 0.0f, 0x7fffffff, 0.5f, 0.0f },
    { 1.0f, -1.0f, -1.0f, 1.0f, 1.0f, 0.0f, 0x7fffffff, 0.0f, 1.0f },
    { 1.0f, -1.0f, 1.0f, 1.0f, 1.0f, 0.0f, 0x7fffffff, 1.0f, 1.0f } };

D3DVERTEX aQuad[] = {
    { -1.0f, -1.0f, -1.0f, 0.0f, -1.0f, 0.0f, 0x7fffffff, 0.0f, 0.0f },
    { 1.0f, -1.0f, -1.0f, 0.0f, -1.0f, 0.0f, 0x7fffffff, 1.0f, 0.0f },
    { -1.0f, -1.0f, 1.0f, 0.0f, -1.0f, 0.0f, 0x7fffffff, 0.0f, 1.0f },
    { 1.0f, -1.0f, 1.0f, 0.0f, -1.0f, 0.0f, 0x7fffffff, 1.0f, 1.0f } };
```

That is all. The alpha blending is done now. We don't have to change more in this project.