

Tutorial 10: Color Key Transparency

by Victor Saar

Content

In the seventh tutorial we already had the topic of alpha blending with the alpha value of the vertex color. This time we will cover a new way of using transparency. We will load a texture, which shows a leaf. As you can see there is a black border around the leaf, but we don't want the black area to be displayed. Here we come to the color key. The color key defines one specific color, in this case black, which shouldn't be drawn to the render surface. The Direct3DX utility library includes a function, which can be used for that. On the image below you can see the leaf.

For more information on alpha blending, please take a look at tutorial 7.

application.cpp

We start with the appropriate render and texture stage states. The first render states are clear. Then we activate alpha blending and define that the source blend factor is taken from our source. Then we define that the destination blend factor is one minus the source blend factor. At the end we define the filters and that the alpha value is taken from the texture.

```
//z-buffer disabled
m_pDirect3DDevice->SetRenderState(D3DRS_ZENABLE, D3DZB_FALSE);

//alpha blending enabled
m_pDirect3DDevice->SetRenderState(D3DRS_ALPHABLENDENABLE, true);
//source blend factor
m_pDirect3DDevice->SetRenderState(D3DRS_SRCBLEND, D3DBLEND_SRCALPHA);
//dest blend factor
m_pDirect3DDevice->SetRenderState(D3DRS_DESTBLEND, D3DBLEND_INVSRCALPHA);

//alpha from texture
m_pDirect3DDevice->SetTextureStageState(0, D3DTSS_ALPHAARG1, D3DTA_TEXTURE);
```

main.cpp

Now comes the interesting stuff. At the beginning we have the needed objects and variables. We create four triangles for two quads. We only have vertices with coordinates for the point and the texture.

```
LPDIRECT3DVERTEXBUFFER9 pLeafVB = NULL;
LPDIRECT3DTEXTURE9 pLeafTexture = NULL;
VOID* pData;
D3DVERTEX aLeaf[] = {
    {-1.0f, -1.0f, 5.0f, 0.0f, 1.0f},
    { 1.0f, -1.0f, 5.0f, 1.0f, 1.0f},
    {-1.0f,  1.0f, 5.0f, 0.0f, 0.0f},
    { 1.0f, -1.0f, 5.0f, 1.0f, 1.0f},
    {-1.0f,  1.0f, 5.0f, 0.0f, 0.0f},
    { 1.0f,  1.0f, 5.0f, 1.0f, 0.0f},
    {-2.0f, -1.0f, 4.0f, 0.0f, 1.0f},
    { 0.0f, -1.0f, 4.0f, 1.0f, 1.0f},
    {-2.0f,  1.0f, 4.0f, 0.0f, 0.0f},
    { 0.0f, -1.0f, 4.0f, 1.0f, 1.0f},
    {-2.0f,  1.0f, 4.0f, 0.0f, 0.0f},
    { 0.0f,  1.0f, 4.0f, 1.0f, 0.0f},
    { 0.0f,  1.0f, 4.0f, 1.0f, 0.0f}};
```

Now we create the vertex buffer and fill it with the vertices.

```

g_App.GetDevice()->CreateVertexBuffer(sizeof(aLeaf),
                                     D3DUSAGE_WRITEONLY,
                                     D3DFVF_CUSTOMVERTEX,
                                     D3DPOOL_MANAGED,
                                     &pLeafVB,
                                     NULL);

pLeafVB->Lock(0, sizeof(pData), (void**)&pData, 0);
memcpy(pData, aLeaf, sizeof(aLeaf));
pLeafVB->Unlock();

```

Now we have to load the texture. We use the function `D3DXCreateTextureFromFileExA()`. The parameter `D3DX_DEFAULT` define that the value is taken from the texture. For more information about the parameters please take a look in the SDK documentation. The important thing is the eleventh parameter, which defines the color key. We set it to black, because we don't want the black border around the leaf to be drawn.

```

D3DXCreateTextureFromFileExA(g_App.GetDevice(), //device
                             "leaf.png",      //file name
                             D3DX_DEFAULT,    //width
                             D3DX_DEFAULT,    //height
                             D3DX_DEFAULT,    //mip levels
                             NULL,           //usage
                             D3DFMT_UNKNOWN,  //texture color format
                             D3DPOOL_MANAGED, //memory class
                             D3DX_DEFAULT,    //filter
                             D3DX_DEFAULT,    //mip filter
                             0xff000000,     //color key
                             NULL,           //source info
                             NULL,           //palette
                             &pLeafTexture); //texture object

```

The main loop needn't to be changed.

```

//set the texture
g_App.GetDevice()->SetTexture(0, pLeafTexture);
//stream source
g_App.GetDevice()->SetStreamSource(0, pLeafVB, 0, sizeof(D3DVERTEX));
//drawing
g_App.GetDevice()->DrawPrimitive(D3DPT_TRIANGLELIST, 0, 4);

```

That's all we need for color key transparency.