

Tutorial 13: Multi Pass Multitexturing

by Victor Saar

Content

There are often people, who have graphic cards that don't support single pass multitexturing or want to use more than the supported textures. These people can use a technic called multi pass multitexturing. In this tutorial we try to replicate the result of the last tutorial with multi pass multitexturing. Multi pass multitexturing is done as follows. We set a texture in the first texture stage and define all arguments needed for rendering. Then we simply render the primitive. Now we set the second texture in the first texture stage and define color or alpha blending render states to adjust the blending factors as needed. Now we render the same primitive again with the new texture and blending factors. The system now blends the already rendered colors and the new colors of the second texture according to the blending factors.

The disadvantage of this technic is that each primitive must be rendered as many times as you want textures on it. Besides there aren't as many operations possible as with single pass multitexturing. Nevertheless we try to reproduce the last tutorial, but as you will see you will not get the same result.

Since we draw the primitive only with one texture, we only need one set of texture coordinates.

main.cpp

Before we can draw anything we have to create the two texture objects and to load the textures, of course.

```
//texture pointers
LPDIRECT3DTEXTURE9 pBaseMap = NULL,pDarkMap = NULL;

//load base texture
D3DXCreateTextureFromFileA(g_App.GetDevice(), "texture.png", &pColorMap);
//load dark map
D3DXCreateTextureFromFileA(g_App.GetDevice(), "darkmap.png", &pDarkMap);
```

In the main loop we first define the standard render states to draw the first texture. We also have to enable alpha blending. This also enables the color blending operations we need. Then we set the first texture in stage 0 and render the primitive.

```
//enable blending operations
g_App.GetDevice()->SetRenderState(D3DRS_ALPHABLENDENABLE, true);
//standard blend factors
g_App.GetDevice()->SetRenderState(D3DRS_SRCBLEND, D3DBLEND_ONE);
g_App.GetDevice()->SetRenderState(D3DRS_DESTBLEND, D3DBLEND_ZERO);

//set first texture
g_App.GetDevice()->SetTexture(0, pBaseMap);
g_App.GetDevice()->SetStreamSource(0, pQuadVB, 0, sizeof(D3DVERTEX));
g_App.GetDevice()->DrawPrimitive(D3DPT_TRIANGLESTRIP, 0, 2);
```

Now comes the important part. We define the blending factors, so that we get the expected result. Then we set the dark map in the first texture stage and render the same primitive again. Try out some of the blending factors to get different results.

```
//blend factors
g_App.GetDevice()->SetRenderState(D3DRS_SRCBLEND, D3DBLEND_ZERO);
```

```
g_App.GetDevice()->SetRenderState(D3DRS_DESTBLEND,D3DBLEND_SRCOLOR);  
  
//dark map  
g_App.GetDevice()->SetTexture(0,pDarkMap);  
g_App.GetDevice()->SetStreamSource(0,pQuadVB,0,sizeof(D3DVERTEX));  
g_App.GetDevice()->DrawPrimitive(D3DPT_TRIANGLESTRIP,0,2);
```