Blending

305890
Spring 2010
4/23/2010
Kyoung Shin Park

Overview

- Blending Equation
- Blend Factors & Mode
- Transparency
- Creating an Alpha Channel Using DX Tex Tool
- Transparent Teapot Demo

Blending Equation

- Blending
  - Allows us to blend (combine) the **original pixels** that we are currently rasterizing with the **destination pixels** that were previously rasterized to the back buffer.
  - If P_{ij} is the source pixel we are currently rasterizing, then P_{ij} is blended with the previous ij-th destination pixel on the back buffer b_{ij}.
- Blending Rule
  1. Draw an object with no blending
  2. Sort blending objects according to the depth from the camera (In viewing space, sort objects by z)
  3. Draw objects in the back to front order.

Blending Equation

- Direct3D Blending Equation

  outputPixel = srcPixel⊗srcBlendFactor + dstPixel⊗dstBlendFactor
  - Each of the above variables is a 4D color vector (r, g, b, a), ⊗ denotes component-wise multiplication
  - outputPixel: the resulting blended pixel
  - srcPixel: the pixel currently being computed that is to be blended with pixel on the back buffer
  - dstPixel: the pixel currently on the back buffer
  - srcBlendFactor & dstBlendFactor: [0,1]

Blending Equation

- Enable/disable blending
  Device->setRenderState(D3DRS_ALPHABLENDENABLE, true);
  - Blending is disabled, by default
  - You can enable blending by setting D3DRS_ALPHAENABLED render state to true, and disable blending by setting it to false.
  - Blending is not a cheap operation and should only be enabled for the geometry that needs it.
  - When you are done rendering that geometry, you should disable blending.
  - Also, try to batch triangles that use blending and render them at once, so that you can avoid turning blending on and off multiple times per frame.

Blend Factors

- Source and destination blend factors are specified by
  Device->SetRenderState(D3DRS_SRCBLEND, Source);
  Device->SetRenderState(D3DRS_DESTBLEND, Destination);

D3DBLEND_ZERO	(0, 0, 0, 0)
D3DBLEND_ONE	(1, 1, 1, 1)
D3DBLEND_SRCCOLOR	(r_s, g_s, b_s, a_s)
D3DBLEND_INVSRCCOLOR	$(1-r_s, 1-g_s, 1-b_s, 10a_s)$
D3DBLEND_SRCALPHA	(a_s, a_s, a_s, a_s) srcFactor의 default
D3DBLEND_INVSRCALPHA	$(1-a_s, 1-a_s, 1-a_s, 1-a_s)$ dstFactor의 default
D3DBLEND_DESTALPHA	(a_d, a_d, a_d, a_d)
D3DBLEND_INVDESTALPHA	$(1-a_d, 1-a_d, 1-a_d, 1-a_d)$
D3DBLEND_DESTCOLOR	(r_d, g_d, b_d, a_d)
D3DBLEND_INVDESTCOLOR	$(1-r_d, 1-g_d, 1-b_d, 1-a_d)$
D3DBLEND_SRCALPHASAT	$(f, f, f, 1)$, $f = \min(a_s, 1-a_d)$
D3DBLEND_BOTHINVSRCALPHA NOTE:D3DRS_SRCBLEND에서만 이용가능	srcFactor = $(1-a_s, 1-a_s, 1-a_s, 1-a_s)$ dstFactor = (a_s, a_s, a_s, a_s)

Blend Factors

```
// set the source & destination blend factors directly in an effect file
pass P0
{
    vertexShader = compile vs_2_0 VS();
    pixelShader = compile vs_2_0 PS();

    AlphaBlendEnable = true;
    SrcBlend = One;
    DestBlend = One;
    BlendOp = RevSubtract;
};
```

Blend Factors

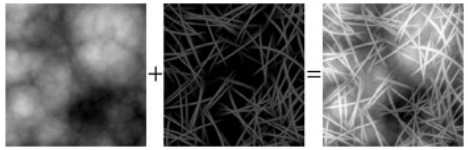
```
// alpha blending
pass P0
{
    vertexShader = compile vs_2_0 VS();
    pixelShader = compile vs_2_0 PS();

    // Use these states to blend RGB components
    AlphaBlendEnable = true;
    SrcBlend = One;
    DestBlend = One;
    BlendOp = RevSubtract;

    // Use these states to blend the alpha components
    SeparateAlphaBlendEnable = true;
    SrcBlendAlpha = SrcAlpha;
    DestBlendAlpha = InvSrcAlpha;
    BlendOpAlpha = Add;
};
```

Blend Factors Example 1

- No blending
  - Want to keep the original destination pixel exactly as it is and not overwrite or blend it with the source pixel currently being rasterized
  - Set the source pixel blend factor to D3DBLEND_ZERO and the destination pixel blend factor to D3DBLEND_ONE.
  - *OutputPixel* = *SourcePixel* ⊗ *SourceBlendFactor* + *DestPixel* ⊗ *DestBlendFactor*
  - *OutputPixel* = *SourcePixel* ⊗ (0, 0, 0, 0) + *DestPixel* ⊗ (1, 1, 1, 1)
  - *OutputPixel* = (0, 0, 0, 0) + *DestPixel*
  - *OutputPixel* = *DestPixel*

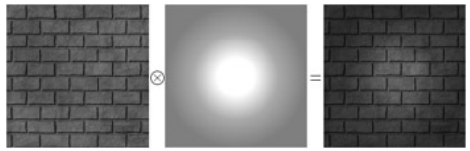
Blend Factors Example 2

- Add blending
  - Want to directly add them together to form a new image
  - Set the source pixel blend factor to D3DBLEND_ONE and the destination pixel blend factor to D3DBLEND_ONE.
  - *OutputPixel* = *SourcePixel* ⊗ *SourceBlendFactor* + *DestPixel* ⊗ *DestBlendFactor*
  - *OutputPixel* = *SourcePixel* ⊗ (1, 1, 1, 1) + *DestPixel* ⊗ (1, 1, 1, 1)
  - *OutputPixel* = *SourcePixel* + *DestPixel*



Blend Factors Example 3

- Multiply blending
  - Want to multiply a source pixel with its corresponding destination pixel
  - Set the source pixel blend factor to D3DBLEND_ZERO and the destination pixel blend factor to D3DBLEND_SRCCOLOR.
  - *OutputPixel* = *SourcePixel* ⊗ *SourceBlendFactor* + *DestPixel* ⊗ *DestBlendFactor*
  - *OutputPixel* = *SourcePixel* ⊗ (0, 0, 0, 0) + *DestPixel* ⊗ *SourcePixel*
  - *OutputPixel* = *DestPixel* ⊗ *SourcePixel*



Blend Factors Example 4

- Alpha blending
  - Want to blend the source and destination pixels based on the transparency percent of the source pixel
  - set the source pixel blend factor to D3DBLEND_SRCALPHA and the destination pixel blend factor to D3DBLEND_INVSRCALPHA.
  - *OutputPixel* = *SourcePixel* ⊗ *SourceBlendFactor* + *DestPixel* ⊗ *DestBlendFactor*
  - *OutputPixel* = *SourcePixel* ⊗ (s_a, s_a, s_a, s_a) + *DestPixel* ⊗ $(1 - s_a, 1 - s_a, 1 - s_a, 1 - s_a)$
  - *OutputPixel* = $s_a \cdot$ *SourcePixel* + $(1 - s_a) \cdot$ *DestPixel*

Transparency

- Transparency
  - To add alpha information to a texture, we create a fourth channel called the alpha channel (transparency).
  - Alpha channel [0, 255]:

    opacity: alpha 0 - 0% transparent

    alpha 128 - 50% half transparent

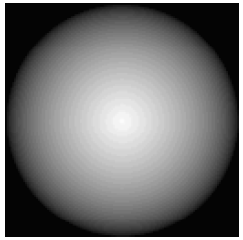
    alpha 255 - 100% opaque
  - If we want to use the texture alpha to set transparency of object, set srcBlendFactor & dstBlendFactor to be D3DBLEND_SRCALPHA & D3DBLEND_INVSRCALPHA (default)

Alpha Channel

- Alpha channel
  - Alpha value: (1) calculate it during shading process, (2) obtain it from the texture alpha channel.
  - Alpha channel (RGBA): By using a texture we can control the transparency of an object at the pixel level, and we can have very complicated patterns of alpha values.

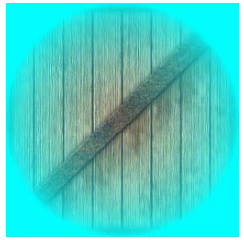
RGB channel

+



Alpha channel

=



Result

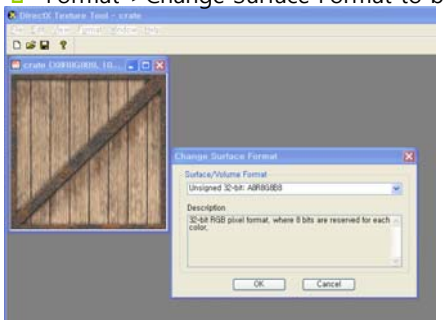
Set Transparency

- 현재 texture가 alpha channel을 가지고 있으면 alpha channel에서 가져오고, 없으면 vertex color에서 가져 옴.
- 어떤 소스를 이용할 것인지를 지정하는 예:

```
// shading과정에서 diffuse color로부터 alpha를 얻도록 지정함
Device->SetTextureStageState(0, D3DTSS_ALPHAARG1, D3DTA_DIFFUSE);
Device->SetTextureStageState(0, D3DTSS_ALPHAOP, D3DTOP_SELECTARG1);

// alpha channel에서 alpha를 얻도록 지정함
Device->SetTextureStageState(0, D3DTSS_ALPHAARG1, D3DTA_TEXTURE);
Device->SetTextureStageState(0, D3DTSS_ALPHAOP, D3DTOP_SELECTARG1);
```

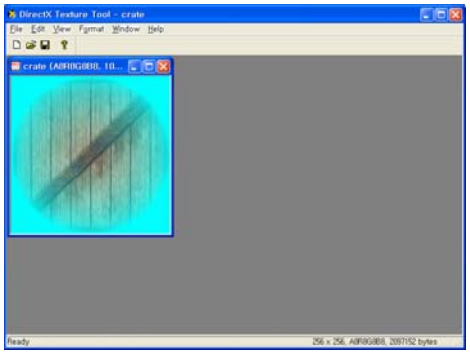
Creating Alpha Channel Using DX Tex Tool

- Create a DDS image file (containing alpha channel)
  - Run DirectX9 SDK Texture Tool
  - Run Program-> Microsoft DirectX 9.0 SDK Update (Feb 2010)-> DirectX Utilities-> DirectX Texture Tool
  - Add the alpha channel
    - File->Open <crate.jpg> (originally, 24-bit RGB)
    - Format->Change Surface Format to be 32-bit A8R8G8B8로 변경



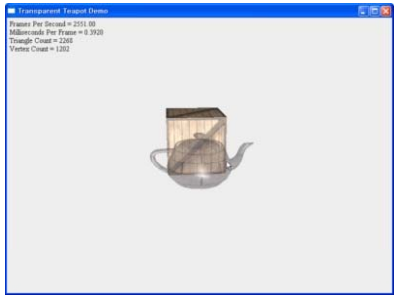
Creating Alpha Channel Using DX Tex Tool

- Create a DDS image file (containing alpha channel)
  - Add the alpha channel in the image file
    - Prepare for 8 bit grey-scale image (e.g., alphachannel.bmp)
    - Load alphachannel.bmp (8-bit grey-scale) by File->Open Onto Alpha Channel Of This Texture
    - Save 'createwalpha.dds' by File->Save As



TeapotDemo

- Render a semi-transparent teapot



TeapotDemo

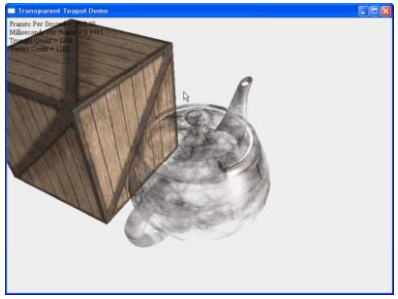
```
mTeapotMtrl.ambient = D3DXCOLOR(1.0f, 1.0f, 1.0f, 1.0f);
mTeapotMtrl.diffuse = D3DXCOLOR(1.0f, 1.0f, 1.0f, 0.5f);
mTeapotMtrl.spec = D3DXCOLOR(0.8f, 0.8f, 0.8f, 1.0f);
mTeapotMtrl.specPower = 16.0f;

// In vertex shader we just pass along the alpha component to
// the pixel shader.
outVS.diffuse.a = gDiffuseMtrl.a;

// In pixel shader we just set the pixel's alpha value to the
// interpolated color's alpha component.
float4 DirLightTexPS(float4 c : COLOR0, float4 spec : COLOR1, float2 tex0 :
    TEXCOORD0) : COLOR
{
    float3 texColor = tex2D(TexS, tex0).rgb;
    float3 diffuse = c.rgb * texColor;
    return float4(diffuse + spec.rgb, c.a);
}
```

TeapotDemo

```
void TeapotDemo::drawTeapot() {
    // Cylindrically interpolate texture coordinates.
    HR(gd3dDevice->SetRenderState(D3DRS_WRAP0, D3DWRAPCOORD_0));

    // Enable alpha blending.
    HR(gd3dDevice->SetRenderState(D3DRS_ALPHABLENDENABLE, true));
    HR(gd3dDevice->SetRenderState(D3DRS_SRCBLEND,
                                  D3DBLEND_SRCALPHA));
    HR(gd3dDevice->SetRenderState(D3DRS_DESTBLEND,
                                  D3DBLEND_IHVSRCALPHA));

    // [...] Set effect parameters/draw teapot

    // Disable alpha blending.
    HR(gd3dDevice->SetRenderState(D3DRS_ALPHABLENDENABLE, false));
}
```

Teapot with Texture Alpha Demo

- Render a semi-transparent teapot, by using the textures alpha channel information.

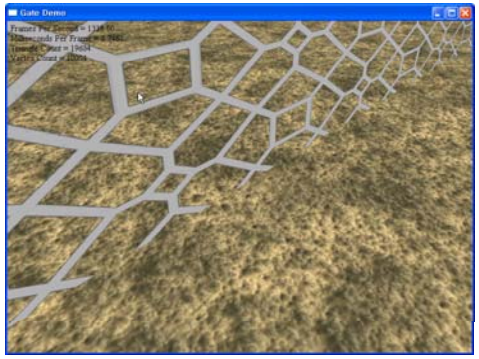


Teapot with Texture Alpha Demo

```
float4 DirLightTexPS(float4 c : COLOR0, float4 spec : COLOR1, float2 tex0 :
    TEXCOORD0) : COLOR
{
    float4 texColor = tex2D(TexS, tex0);
    float3 diffuse = c.rgb * texColor.rgb;
    return float4(diffuse + spec.rgb, texColor.a*c.a);
}
```

Gate Demo

- The alpha test allows us to discard a pixel (i.e., not render it) based on its alpha value in comparison to some reference value.
  - IF $a_s \odot$ *ref == true* THEN *accept pixel*
  - ELSE *reject pixel*



Teapot with Texture Alpha Demo

```
HR(gd3dDevice->SetRenderState(D3DRS_ALPHATESTENABLE, true));
HR(gd3dDevice->SetRenderState(D3DRS_ALPHAFUNC,
                                                D3DCMP_GREATEREQUAL));
HR(gd3dDevice->SetRenderState(D3DRS_ALPHAREF, 100));

// effect file..
pass P0 {
    vertexShader = compile vs_2_0 VS();
    pixelShader = compile ps_2_0 PS();

    // alpha test
    AlphaTestEnable = true;
    AlphaFunc = GreaterEqual;
    AlphaRef = 220;
}
```