Texturing

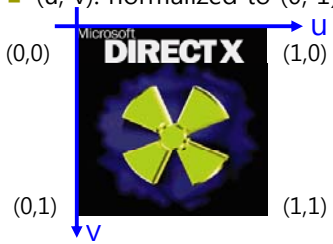
305890
Spring 2010
4/16/2010
Kyoung Shin Park

Overview

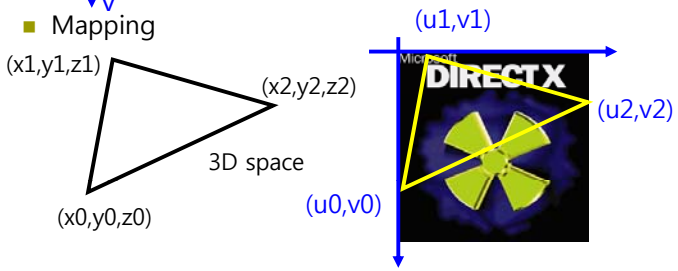
- Texture coordinates
- Create and enable textures
- Texture filters
- Mipmaps
- Address Modes
- Tiled Ground

Texture Coordinates

- Texture Coordinates
  - (u, v): normalized to (0, 1)

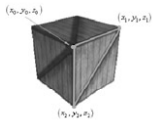


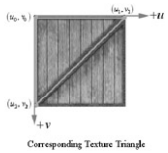
  - Mapping



Texture Coordinates

- Vertex structure include texture coordinates






```
struct VertexPNT {
    float x, y, z;      // XYZ
    float nx, ny, nz;   // normal
    float u, v;         // texture coordinates
    static IDirect3DVertexDeclaration9* Decl;
};

D3DVERTEXELEMENT9 VertexPNTElements[] =        {
    {0, 0,  D3DDECLTYPE_FLOAT3, D3DDECLMETHOD_DEFAULT,
D3DDECLUSAGE_POSITION, 0},
    {0, 12, D3DDECLTYPE_FLOAT3, D3DDECLMETHOD_DEFAULT,
D3DDECLUSAGE_NORMAL, 0},
    {0, 24, D3DDECLTYPE_FLOAT2, D3DDECLMETHOD_DEFAULT,
D3DDECLUSAGE_TEXCOORD, 0},
    D3DDECL_END()
};
```

Creating and Enabling Textures

- Load an image file into IDIRECT3DTexture9 object.
  - D3DXCreateTextureFromFile
  - Return: D3D_OK or D3DERR_xxx
  - This function can load any of the following image formats: BMP, DDS, DIB, HDR, JPG, PFM, PNG, PPM, and TGA.

```
HRESULT WINAPI D3DXCreateTextureFromFile(
    LPDIRECT3DDEVICE9 pDevice,          // point to IDirect3DDevice9
    LPCTSTR pSrcFile,                   // filename
    LPDIRECT3DTEXTURE9 **ppTexture); // out
```

Creating and Enabling Textures

- To create a texture from an image "stonewall.bmp"

```
IDirect3DTexture9* mTex0;
D3DXCreateTextureFromFile(_device, "crate.jpg", &mTex0);
```

- To set a created IDirect3DTexture9 object to texture effect parameter, call ID3DXEffect::SetTexture.

```
// .fx file
uniform extern texture gTex;
// texture mappping program
mFX->SetTexture(mhTex, mTex0);
mFX->CommitChanges();
drawTriangleUsingTex0();
```

Creating and Enabling Textures

- To set no texture

```
mFX->SetTexture(mhTex, 0);
drawTrisUsingTex0();          // draw the same textured cube
```

- To use different textures but are drawn using the same effect:

```
mFX->SetTexture(mhTex, mTex0);
drawTrisUsingTex0();

mFX->SetTexture(mhTex, mTex1);
drawTrisUsingTex1();
```

Filters

- When the texture triangle is smaller than the screen triangle, the texture triangle is magnified to fit.
- When the texture triangle is large than the screen triangle, the texture triangle is minified to fit.
- Mapping filter
  - MAGFILTER
  - MINFILTER

```
uniform extern texture gTex;
sampler TexS = sampler_state  {
      Texture = <gTex>;
      MinFilter = LINEAR;
      MagFilter = LINEAR;
};
```

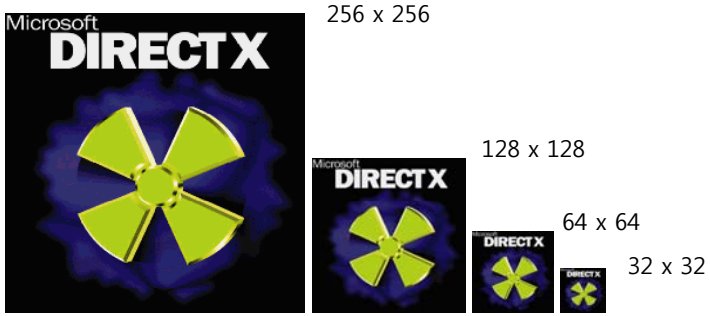
Filters

- 3 types of filters
  - Nearest point sampling: POINT, poor quality, faster (default)
  - Linear filtering: LINEAR, high quality, relatively fast (recommended)
  - Anisotropic filtering: Anisotropic, higher quality, relatively slow.
    - Must also set D3DSAMP_MAXANISOTROPY level (to determine the quality of the anisotropic filtering). (Default is 1)

```
uniform extern texture gTex0;
sampler Tex0S = sampler_state {
    Texture = <gTex0>;
    MinFilter = Anisotropic;
    MagFilter = LINEAR;
    MaxAnisotropy = 4;
};
```

Mipmaps

- We can create a chain of mipmaps for a texture.
  - The idea is to take a texture and create a series of smaller, lower-resolution textures, but customizing the filtering for each of these levels.



Mipmap

- Mipmap filter:
  - NONE: disable mipmapping
  - POINT: Direct3D chooses the mipmap level that is closest in size to the screen triangle. Once that level is chosen, Direct3D filters that level based on the specified min and mag filters.
  - LINEAR: Direct3D takes the two mipmap levels that are closest in size to the screen triangle, filters each level with the min and mag filters, and finally linearly combines these two levels to form the final color values.

  MipFilter = Filter;
- Direct3D mipmap
  - D3DXCreateTextureFromFile generates a mipmap chain for your. In addition, Direct3D automatically selects the mipmap that best matches the screen triangle based on the specified filter.

Address Modes

- The texture coordinates that go outside [0, 1] range is defined by Direct3D address mode:
  - Address mode:
    - Wrap
    - Border color
    - Clamp
    - mirror
  - D3DTEXTUREADDRESS enum type
    - D3DTADDRESS_WRAP: repeat the texture on every integer junction
    - D3DTADDRESS_MIRROR: every other row and column is a mirrors image of the preceding row or column
    - D3DTADDRESS_CLAMP: smear the color of edge pixels
    - D3DADDRESS_BORDER: use the border color, for any texture coordinates outside the range

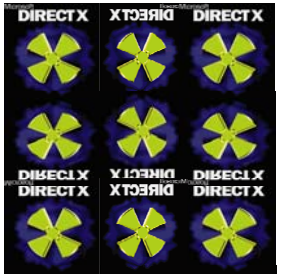
Address Modes

```
// wrap
sampler TexS = sampler_state {
    Texture = <gTex>;
    MinFilter = LINEAR; MagFilter = LINEAR; MipFilter = LINEAR;
    AddressU = WRAP; AddressV = WRAP; };
// border color
sampler TexS = sampler_state {
    Texture = <gTex>;
    MinFilter = LINEAR;     MagFilter = LINEAR; MipFilter = LINEAR;
    AddressU = BORDER; AddressV = BORDER; BorderColor = 0xff0000ff; };
// clamp
sampler TexS = sampler_state {
    Texture = <gTex>;
    MinFilter = LINEAR; MagFilter = LINEAR; MipFilter = LINEAR;
    AddressU = CLAMP; AddressV = CLAMP; };
// mirror
sampler TexS = sampler_state {
    Texture = <gTex>;
    MinFilter = LINEAR; MagFilter = LINEAR; MipFilter = LINEAR;
    AddressU = MIRROR; AddressV = MIRROR; };
```

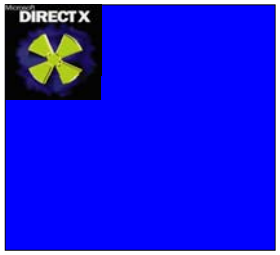
Address Modes

- Quad (0,0),(0,3),(3,0),(3,3):

wrap



mirror

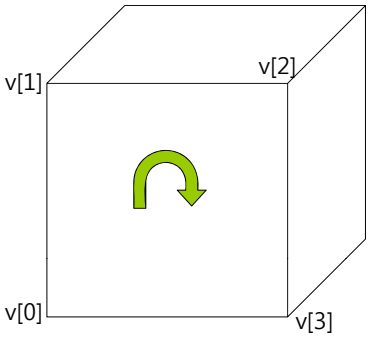


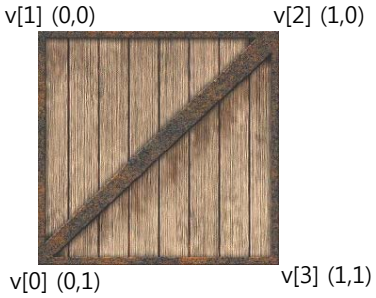
border color

clamp

Crate Demo

- Add a texture to a cube.
  1. Specifying the texture coordinates.
  2. Creating the texture using D3DXCreateTextureFromFile.
  3. Sampling the Texture.
  4. Setting the Texture using SetTexture.



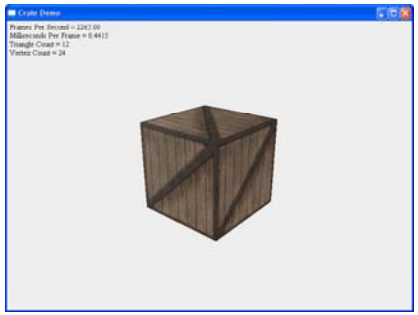


Crate Demo

```
// Write box vertices to the vertex buffer.
VertexPNT* v = 0;
HR(mBoxVB->Lock(0, 0, (void**)&v, 0));

// Fill in the front face vertex data. (CW winding order) (normal=0,0,-1)
v[0] = VertexPNT(-1.0f, -1.0f, -1.0f, 0.0f, 0.0f, -1.0f, 0.0f, 1.0f);
v[1] = VertexPNT(-1.0f,  1.0f, -1.0f, 0.0f, 0.0f, -1.0f, 0.0f, 0.0f);
v[2] = VertexPNT( 1.0f,  1.0f, -1.0f, 0.0f, 0.0f, -1.0f, 1.0f, 0.0f);
v[3] = VertexPNT( 1.0f, -1.0f, -1.0f, 0.0f, 0.0f, -1.0f, 1.0f, 1.0f);

// Fill in the back face vertex data. (CW winding order) (normal=0,0,1)
v[4] = VertexPNT(-1.0f, -1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, 1.0f);
v[5] = VertexPNT( 1.0f, -1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 0.0f, 1.0f);
v[6] = VertexPNT( 1.0f,  1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 0.0f, 0.0f);
v[7] = VertexPNT(-1.0f,  1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, 0.0f);

// 중간생략..
HR(mBoxVB->Unlock());
```

Tiled Ground Demo

```
sampler TexS = sampler_state
{
    Texture = <gTex>;
    MinFilter = Anisotropic;
    MagFilter = LINEAR;
    MipFilter = LINEAR;
    MaxAnisotropy = 8;
    AddressU = WRAP;
    AddressV = WRAP;
};
```



Tiled Ground Demo

- Tile a ground texture repeatedly over a grid to provide us with a "ground" plane.

```
// tile a texture over a grid mesh
VertexPNT* v = 0;
mGridVB->Lock(0, 0, (void**)&v, 0);
float texScale = 0.2f;
for (int i = 0; i < numVertRows; ++i) {
    for (int j = 0; j < numVertCols; ++j) {
        DWORD index = i* numVertCols + j;
        v[index].pos = verts[index];                          // position
        v[index].normal = D3DXVECTOR3(0.0f, 1.0f, 0.0f);      // normal
        v[index].tex = D3DXVECTOR2((float)j, (float)i) * texScale;// texture
    }
}
mGridVB->Unlock();
```

Tiled Ground Demo

```
sampler TexS = sampler_state
{
    Texture = <gTex>;
    MinFilter = Anisotropic;
    MagFilter = LINEAR;
    MipFilter = LINEAR;
    MaxAnisotropy = 8;
    AddressU = WRAP;
    AddressV = WRAP;
};
```