Effect Pools Sample

Collapse All

This sample demonstrates the use of effect pools on Direct3D 9 and Direct3D 10.

Path

Source	*SDK root*\Samples\C++\Direct3D10\EffectPools
Executable	*SDK root*\Samples\C++\Direct3D10\Bin*x86 or x64*\EffectPools.exe

How the Sample Works

One of the downsides to using multiple effects in a single application is that many effect variables need to be set for each effect. It is often the case that many of these variables are common between effects. Without a method for sharing variables or shaders between effects, many redundant set calls will need to be made setting data that is exactly the same for each effect. Effect pools are meant to solve this problem. Effect pools allow different effects to share variables, textures, and shaders between them. This means that for common variables, the data needs to be set only once per frame, rather than once per effect invocation. An effect pool effectively consists of two parts. There is the effect pool itself, and the effects that are part of the pool. These effects that are part of the same pool are call child effects.

This sample uses 3 effect files to render 3 spheres. The effect files share light direction (g_LightDir), light diffuse color (g_LightDiffuse), the viewproj matrix (g_mViewProj), and texture uv offset (g_TexMove) between them. In addition, they share the same diffuse texture resource, and pixel shader. However, these effect do have their differences, and therefore need their their own vertex shaders as well as world matrices (g_mWorld) and diffuse colors (g_MaterialDiffuseColor). The common variables are defined in an include file, EffectPools.fxh, while the individual child effect files (EffectPools1-3.fx) include EffectPools.fxh, but define their own unique variables and shaders.

Include Files and Shared Variables

The benefit of using effect pools is that you can share data between different effect files. While you can include the same header file in each separate effect, without using effect pools, the variables and shaders are just duplicated between effects. If you change variable A in one effect file, you still need to change the same variable in the other effect file in order for it to have the correct data. Effect pools allow "shared" variables. Shared variables can be used by all effect files in the pool. When you set the value of a shared variable for an effect pool, you don't need to redundantly set the same variable for each effect in the pool. A good example of this is the viewproj matrix. In many cases, the viewproj matrix is constant over an entire frame. However, we may have several hundred effects being used in the scene. Each one needs the viewproj matrix. Without using effect pools, we would have to set this variable once for each effect. However, by using effect pools, we set this variable once, period.

An easy way to use shared variables is to use an include file for your shaders. The include file contains all of the variables and shaders that will be shared by the application effects. These variables and shaders must be marked as "shared" in order for the effect system to know that they are shared variables and can be used by multiple effects in the pool. The shared keyword usage varies a little between Direct3D 9 and Direct3D 10.

On Direct3D 9 each variable needs to be preceded by the shared keyword. Sampleer cannot be shared on Direct3D 9.

```
shared float3 g_LightDir;               // Light's direction in world space
shared float4 g_LightDiffuse;           // Light's diffuse color
shared float4x4 g_mViewProj;            // View * Projection matrix
shared float  g_TexMove;                // random variable that offsets the V texcoord
```

```
shared texture2D g_txDiffuse;
sampler2D MeshTextureSampler = sampler_state
{
    Texture = (g_txDiffuse);
    MinFilter = Linear;
    MagFilter = Linear;
    AddressU = WRAP;
    AddressV = WRAP;
};
```

On Direct3D 10 shared constants must be grouped into constant buffers. Only entire constant buffers can be shared. You are not limited to one constant shared constant buffer. In addition, because samplers are separate objects in shader model 4.0, samplers can be shared.

```
shared cbuffer cbShared
{
float3 g_LightDir;                  // Light's direction in world space
float4 g_LightDiffuse;              // Light's diffuse color
float4x4 g_mViewProj;               // View * Projection matrix
float  g_TexMove;                   // random variable that offsets the V texcoord
};


shared texture2D g_txDiffuse;
shared sampler2D MeshTextureSampler = sampler_state
{
    Texture = (g_txDiffuse);
    MinFilter = Linear;
    MagFilter = Linear;
    AddressU = WRAP;
    AddressV = WRAP;
};
```

In either Direct3D 9 or Direct3D 10 shaders can be shared by placing the shared keyword in front of the shader declaration.

```
shared float4 ScenePS( PS_INPUT input ) : COLOR0
{
    ...
}
```

Child Effects

In order for a child effect to use the shared variables it has to include the header file. Nothing else needs to be done. Variables and shaders are accessed as if they were variables local to the effect.

An include file is not necessary to share variables. Child effects can declare variables as shared inside the effect file. However, care must be taken to ensure that the shared variables names are identical between shaders. It is often easier just to use an include file.

Creating the Effect Pool

The actual creation of the effect pool varies between Direct3D 9 and Direct3D 10. In Direct3D 9, the effect pool is created by the device, but contains no initial information about shared variables. This information is later collected when child effects are added to the effect pool.

```
    V_RETURN( D3DXCreateEffectPool( &g_pEffectPool ) );
```

In Direct3D 10, the effect pool can be created from a file. This will likely be the header file used for all of the child effect files.

```
    V_RETURN( D3DX10CreateEffectPoolFromFile( str,
                                              NULL,
                                              NULL,
                                              "fx_4_0",
                                              D3D10_SHADER_ENABLE_BACKWARDS_COMPATIBILITY,
                                              0,
                                              pd3dDevice,
                                              NULL,
                                              &g_pEffectPool,
                                              NULL,
                                              NULL ) );
```

Creating the Child Effects

The creation of child effects is similar between both APIs. On Direct3D 9, the effect is compiled normally, but the effect pool is passed in as one of the parameters. This makes the effect a child effect.

```
V_RETURN( D3DXCreateEffectFromFile( pd3dDevice,
                                    str,
                                    NULL,
                                    NULL,
                                    dwShaderFlags,
                                    g_pEffectPool,
                                    &g_EffectData[i].pEffect,
                                    NULL ) );
```

LIkewise, in Direct3D 10, the effect pool is passed in as a parameter. However, the childe must also be compiled with the D3D10_EFFECT_COMPILE_CHILD_EFFECT flag set.

```
V_RETURN( D3DX10CreateEffectFromFile( str,
                                      NULL,
                                      NULL,
                                      "fx_4_0",
                                      dwShaderFlags,
                                      D3D10_EFFECT_COMPILE_CHILD_EFFECT,
                                      pd3dDevice,
                                      g_pEffectPool,
                                      NULL,
                                      &g_EffectData[i].pEffect,
                                      NULL,
                                      NULL ) );
```

Pre-Compiled Effects

Effects pools are not limited to effects that are compiled at runtime. Most applications can gain signification performance improvements by compiling their effects offline. Pre-compiled effects can be loaded as child effects in exactly the same way as text effect files. However, the files must be precompiled as child effects for them to work in effect pools. Precompiled effects can be used as child effects alongside text effects.

The sample uses fxc.exe to precompile EffectPools3.fx in a custom build step and saves the compiled output to EffectPools3.fxo9 or EffectPools3.fxo10 depending on API. This is then loaded in place of EffectPools3.fx at runtime. In order to precompile an effect as a child effect using fxc.exe, the following command line options must be used. On Direct3D 9, the effect needs simply to compiled with /Fo to turn it into a binary output file. On Direct3D 10, to remain analogous to the D3D10_EFFECT_COMPILE_CHILD_EFFECT paradigm of the API, the effect also needs to be compiled with /Gch. This flag compiles the effect as a child effect for fx_4_0 targets.