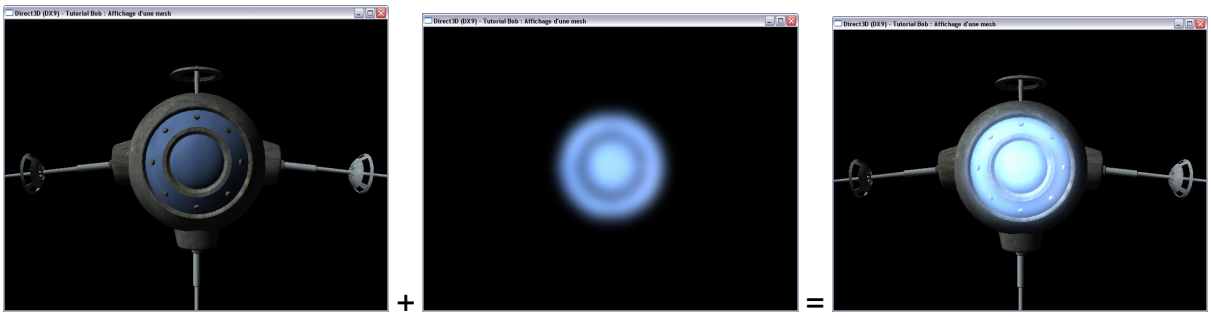
Tutorial 2 : Glow

Description : L'effet de **GLOW** est pour rendre le halo de lumière entourant une source.

Nous allons voir comment réaliser cet effet en shader avec l'aide du langage Cg. Tout d'abord, il vous faut avoir des connaissances en DirectX et quelques notions sur les shader pour comprendre ce tutorial.

Explication : Pour rendre cet effet il nous faut comprendre la technique… Afin de rendre un halo sur un objet de la scène, il faut tout d'abord le rendre dans une texture ensuite créer un flou sur cette texture et enfin l'appliquer sur un polygone qui va prendre tout l'écran.



Un .fx est composés de vertex et pixel shaders. On peut en avoir autant que l'on souhaite et il permet de mettre plusieurs techniques ainsi que de faire des multi-passes pour le rendu.

Dans ce fichier nous pouvons déclarer des variables globales ainsi que définir les propriétés des différents attributs du shader.

Exemple pour les attributs d'une texture :

```
sampler GlowSampler = sampler_state
{
    Texture   = <glowTexture>;
    MipFilter = LINEAR;
    MinFilter = LINEAR;
    MagFilter = LINEAR;
};
```

Ici nous avons 4 vertexshader pour rendre les blurs dans des sens différents :

```
GLOW_OUTPUT glowVSHorizontal1(GLOW_INPUT IN)
{
      GLOW_OUTPUT OUT;

      OUT.pos =  mul(IN.pos, texTrans);
      OUT.texCoord0 = IN.texCoord + float2(-pSize*3, 0);
      OUT.texCoord1 = IN.texCoord + float2(-pSize*2, 0);
      OUT.texCoord2 = IN.texCoord + float2(-pSize*1, 0);
      OUT.texCoord3 = IN.texCoord;
```

```
	return OUT;
}
GLOW_OUTPUT glowVSHorizontal2(GLOW_INPUT IN)
{
	GLOW_OUTPUT OUT;

	OUT.pos =  mul(IN.pos, texTrans);
	OUT.texCoord0 = IN.texCoord + float2(pSize*3,0);
	OUT.texCoord1 = IN.texCoord + float2(pSize*2, 0);
	OUT.texCoord2 = IN.texCoord + float2(pSize*1, 0);
	OUT.texCoord3 = IN.texCoord;

	return OUT;
}
GLOW_OUTPUT glowVSVertical1(GLOW_INPUT IN)
{
	GLOW_OUTPUT OUT;

	OUT.pos =  mul(IN.pos, texTrans);
	OUT.texCoord0 = IN.texCoord + float2(0,-pSize*3);
	OUT.texCoord1 = IN.texCoord + float2(0,-pSize*2);
	OUT.texCoord2 = IN.texCoord + float2(0,-pSize*1);
	OUT.texCoord3 = IN.texCoord;

	return OUT;
}
GLOW_OUTPUT glowVSVertical2(GLOW_INPUT IN)
{
	GLOW_OUTPUT OUT;

	OUT.pos =  mul(IN.pos, texTrans);
	OUT.texCoord0 = IN.texCoord + float2(0,pSize*3);
	OUT.texCoord1 = IN.texCoord + float2(0,pSize*2);
	OUT.texCoord2 = IN.texCoord + float2(0,pSize*1);
	OUT.texCoord3 = IN.texCoord;

	return OUT;
}
```

Et enfin il nous faut le pixelshader :

```
pixel glowPS(GLOW_OUTPUT IN)
{
	pixel OUT;
	float4 color = tex2D( GlowSampler, IN.texCoord0 ) * 0.1;
	color += tex2D( GlowSampler, IN.texCoord1 ) * 0.3;
	color += tex2D( GlowSampler, IN.texCoord2 ) * 0.4;
	color += tex2D( GlowSampler, IN.texCoord3 ) * 0.25;

	OUT.color = color;
	OUT.color.a = 1.0f;

	return OUT;
}
```

Il nous faut maintenant déterminer les techniques ainsi que les passes pour rendre l'effet :

```
technique T0
```

```
{
	pass P0 <string renderStage="texture";>
	{
		Sampler[0] = (GlowSampler);
		vertexshader = compile vs_1_1 glowVSHorizontal1();
		pixelshader  = compile ps_1_1 glowPS();
		fvf = XYZ | Tex1;
	}
	pass P1 <string renderStage="texture";>
	{
		Sampler[0] = (GlowSampler);
		vertexshader = compile vs_1_1 glowVSHorizontal2();
		pixelshader  = compile ps_1_1 glowPS();
		fvf = XYZ | Tex1;
	}
	pass P2 <string renderStage="texture";>
	{
		Sampler[0] = (GlowSampler);
		vertexshader = compile vs_1_1 glowVSVertical1();
		pixelshader  = compile ps_1_1 glowPS();
		fvf = XYZ | Tex1;
	}
	pass P3 <string renderStage="texture";>
	{
		Sampler[0] = (GlowSampler);
		vertexshader = compile vs_1_1 glowVSVertical2();
		pixelshader  = compile ps_1_1 glowPS();
		fvf = XYZ | Tex1;
	}
}
```

Et voici le rendu final :