# 17.3 对天空进行纹理映射

在本节中， 我们使用一个环境贴图对天空进行纹理映射。 我们创建了一个包围整个场景的椭圆体（用椭圆体来模拟一个较为平坦的天空表面）。我们通过图 17.6 所示的方法，使用一个环境贴图对椭圆体进行纹理映射，将环境贴图投影到椭圆体表面上，模拟天地相接处的远山假象。

****

**图17.6：为简单起见，我们在2D空间中描述一概念；读者可以把个正方形想像为一个3D空间中的立方体，把个椭圆形想像为一个3D空间中的椭圆体。我们假设天空和环境贴图的中心均与坐标系原点对齐。然后，我们对椭圆体表面上的点进行纹理映射，使用从原点指向表面点的向量作为立方体贴图的查找向量，将立方体贴图投影到椭圆体上。**

我们假设天空椭圆体无限遥远（即，它的中心与世界空间的原点对齐，但是有无限大的半径），无论摄像机如何在世界空间中移动，我们永远都无法靠近或远离天空椭圆体表面。要实现这个无限遥远的天空，我们只需要在世界空间中将天空椭圆体的中心和摄像机的位置对齐，让它跟着摄像机移动。这样无论摄像机移动到哪，我们都不会靠近天空椭圆体表面。如果不这样做，那么当摄像机与天空椭圆体表面越来越近时，整个假象就会被拆穿，我们用来模拟天空的这个小把戏就没什么意思了。

用来实现天空的effect文件如下：

//=============================================================================

// Sky.fx by Frank Luna (C) 2011 All Rights Reserved.

//

// Effect used to shade sky dome.

//=============================================================================

cbuffer cbPerFrame

{

 float4x4 gWorldViewProj;

};

// Nonnumeric values cannot be added to a cbuffer.

TextureCube gCubeMap;

SamplerState samTriLinearSam

{

 Filter = MIN\_MAG\_MIP\_LINEAR;

 AddressU = Wrap;

 AddressV = Wrap;

};

struct VertexIn

{

 float3 PosL : POSITION;

};

struct VertexOut

{

 float4 PosH : SV\_POSITION;

 float3 PosL : POSITION;

};

VertexOut VS(VertexIn vin)

{

 VertexOut vout;

 // 将z设置为w，这样z/w = 1 (即天空球总是在位于平面).

 vout.PosH = mul(float4(vin.PosL, 1.0f), gWorldViewProj).xyww;

 // 使用局部顶点位置作为立方体贴图的查找向量

 vout.PosL = vin.PosL;

 return vout;

}

float4 PS(VertexOut pin) : SV\_Target

{

 return gCubeMap.Sample(samTriLinearSam, pin.PosL);

}

RasterizerState NoCull

{

 CullMode = None;

};

DepthStencilState LessEqualDSS

{

 // 将深度测试函数设置为LESS\_EQUAL而不是LESS.

 // 否则，当深度缓冲清除为1时，z=1(NDC)处规范化深度值将通不过深度测试。

 DepthFunc = LESS\_EQUAL;

};

technique11 SkyTech

{

 pass P0

 {

 SetVertexShader( CompileShader( vs\_5\_0, VS() ) );

 SetGeometryShader( NULL );

 SetPixelShader( CompileShader( ps\_5\_0, PS() ) );

 SetRasterizerState(NoCull);

 SetDepthStencilState(LessEqualDSS, 0);

 }

}

**注意**：在以前，应用程序会先绘制天空，把它作为一个替代物来清空渲染目标和深度/模板缓冲区。不过，现在出于以下原因《ATI Radeon HD 2000编程指南》（<http://ati.amd.com/developer/SDK/AMD_SDK_Samples_May2007/Documentations/ATI_Radeon_HD_2000_programming_guide.pdf>）建议我们不要再使用这种方式。首先，深度/模板缓冲区应该被直接清空，这样可以获得硬件内部的深度优化，有利于提高性能。这一点与渲染目标的情形相似。其次，天空通常会被其他物体阻挡，比如建筑物和地形。当我们先绘制天空时，天空图像会被与摄像机距离更近的其他物体覆盖，是对资源的浪费。所以，现在的首选方式是始终调用Clear\*\*\*方法清空各个缓冲区，最后绘制天空。