# 8.1 纹理和资源概述

我们的演示程序正在变得越来越有趣，但是顶点颜色无法体现真实世界物体所具有的细节。纹理贴图映射（texture mapping）是一种将图像数据映射到三角形表面的技术，它可以显著提高场景的细节和真实感。例如，我们可以创建一个立方体，然后将一个板条箱纹理映射到立方体的每个面上，使它看上去更像是一个板条箱（参见图8.1）。

****

**图8.1：板条箱演示程序创建了一个带有板条箱纹理的立方体。**

**学习目标**

1．学习如何将纹理片段映射到一个三角形上。

2．了解如何创建和启用纹理。

3．学习如何通过纹理过滤来消除图像中的锯齿，使整个图像看上去更平滑。

4．了解如何使用寻址模式实现纹理平铺。

5．了解如何将多重纹理合并为一幅新的纹理，实现一些特殊效果。

6．学习如何通过纹理动画来创建一些简单的效果。

回顾我们自从第4章之后学习和使用过的一些纹理；尤其是深度缓冲区和后台缓冲区，它们都是通过**ID3D11Texture2D**接口描述的2D纹理对象。在本节中，我们将回顾第4章讲过的一些有关纹理和材质的知识。

2D纹理是一种数据元素矩阵。2D纹理的用途之一是存储2D图像数据，在纹理的每个元素中保存一个像素颜色。但这不是纹理的唯一用途；例如，在一种称为法线贴图映射（normal mapping）的高级技术中，纹理存储的不是颜色，而是3D向量。因此，从通常意义上讲，纹理用来存储图像数据，但是在实际应用中纹理可以有更广泛的用途。1D纹理（**ID3D11Texture1D**）类似于一个1D数据元素数组，3D纹理（**ID3D11Texture3D**）类似一个3D数据元素数组。1D、2D、3D纹理接口都继承自**ID3D11Resource**接口。

本章稍后会讨论纹理的更多特性。纹理比数据数组要复杂的多；纹理可以带有多级渐近纹理层（mipmap level），GPU可以在纹理上执行特殊运算，比如使用过滤器（filter）和多重采样（multisampling）。但是并非所有数据块都能存入纹理；纹理只支持特定格式的数据存储，这些格式由**DXGI\_FORMAT**枚举类型描述。其中一些常用的格式如下：

**1．DXGI\_FORMAT\_R32G32B32\_FLOAT**：每个元素包含3个32位浮点分量。

**2．DXGI\_FORMAT\_R16G16B16A16\_UNORM**：每个元素包含4个16位分量，分量的取值范围在[0,1]区间内。

**3．DXGI\_FORMAT\_R32G32\_UINT**：每个元素包含2个32位无符号整数分量。

**4．DXGI\_FORMAT\_R8G8B8A8\_UNORM**：每个元素包含4个8位无符号整数分量，分量的取值范围在[0,1]区间内。

**5．DXGI\_FORMAT\_R8G8B8A8\_SNORM**：每个元素包含4个8 位有符号整数分量，分量的取值范围在[−1,1] 区间内。

**6．DXGI\_FORMAT\_R8G8B8A8\_SINT**：每个元素包含4个8位有符号整数分量，分量的取值范围在[−128, 127]区间内。

**7．DXGI\_FORMAT\_R8G8B8A8\_UINT**：每个元素包含4个8位无符号整数分量，分量的取值范围在[0, 255]区间内。

注意，字母R、G、B、A分别表示red（红）、green（绿）、blue（蓝）和alpha（透明度）。不过，正如我们之前所述，纹理存储的不一定是颜色信息；例如，格式

**DXGI\_FORMAT\_R32G32B32\_FLOAT**

包含3个浮点分量，可以存储一个使用浮点坐标的3D向量（而不一定是颜色向量）。另外，还有一种弱类型（typeless）格式，可以预先分配内存空间，然后在纹理绑定到管线时再指定如何重新解释数据内容（这一过程与数据类型转换颇为相似）。例如，下面的弱类型格式会为每个元素预留4个8位分量，且不指定数据类型（例如：整数、浮点数、无符号整数）：

**DXGI\_FORMAT\_R8G8B8A8\_TYPELESS**

纹理可以被绑定到渲染管线的不同阶段；例如，比较常见的情况是将纹理作为渲染目标（即，Direct3D渲染到纹理）和着色器资源（即，在着色器中对纹理进行采样）。当创建用于这两种目的的纹理资源时，应使用绑定标志值：

**D3D11\_BIND\_RENDER\_TARGET | D3D11\_BIND\_SHADER\_RESOURCE**

指定纹理所要绑定的两个管线阶段。其实，资源不能被直接绑定到一个管线阶段；我们只能把与资源关联的资源视图绑定到不同的管线阶段。无论以哪种方式使用纹理，Direct3D始终要求我们在初始化时为纹理创建相关的资源视图。这样有助于提高运行效率，正如SDK文档指出的那样：“运行时环境与驱动程序可以在视图创建执行相应的验证和映射，减少绑定时的类型检查”。所以，当把纹理作为一个渲染目标和着色器资源时，我们要为它创建两种视图：渲染目标视图（**ID3D11RenderTargetView**）和着色器资源视图（**ID3D11ShaderResourceView**）。资源视图主要有两个功能：（1）告诉Direct3D如何使用资源（即，指定资源所要绑定的管线阶段）；（2）如果在创建资源时指定的是弱类型（typeless）格式，那么在为它创建资源视图时就必须指定明确的资源类型。对于弱类型格式，纹理元素可能会在一个管线阶段中视为浮点数，而在另一个管线阶段中视为整数。

**注意**：2008年8月的SDK文档指出：“当创建资源时，为资源指定强类型（fully-typed）格式，把资源的用途限制在格式规定的范围内，有利于提高运行时环境对资源的访问速度……”。所以，你只应该在真正需要弱类型资源时，才创建弱类型资源；否则，应尽量创建强类型资源。

为了给一个资源创建特定的视图，该资源在创建时必须带有特定的绑定标志值。例如，在创建资源时如果没有使用**D3D11\_BIND\_SHADER\_RESOURCE**绑定标志值（该标志值表示纹理将作为一个着色器资源绑定到管线上），那么我们就无法为这个资源创建**ID3D11ShaderResourceView**视图。如果你尝试一下就会发现Direct3D 会报告如下调试

错误：

D3D11: ERROR: ID3D11Device::CreateShaderResourceView:

A ShaderResourceView cannot be created of a Resource that

did not specify the D3D11\_BIND\_SHADER\_RESOURCE BindFlag.

在本章中，我们主要讲解如何将纹理绑定为着色器资源，以使我们的像素着色器可以对纹理进行采样，并用纹理来实现几何体着色。