**9 纹理贴图**

**9.1 纹理与资源的回顾**

纹理贴图是一种将图像数据映射到网格三角形上的技术。

2D纹理是一种由特定数据元素所构成的矩阵，它的用处之一即是存储2D图像数据，每个元素存储着对应像素的颜色。但这并不是它唯一的用途，比如说，在法线贴图中，每个纹理元素都存储的是一个3D向量。

1D纹理与3D纹理就像由数据元素构成的1D，3D数组，1D, 2D以及3D纹理都用泛型接口ID3D12Resource来表示。

纹理所支持的数据格式由枚举DXGI\_FORMAT表示。

**9.2 纹理坐标**

Direct3D所采用的纹理坐标系，是由指向图像水平正方向的u轴与指向图像垂直正方向的v轴所组成的。取值范围为0<=u,v<=1的坐标(u,v)标定的是一种称为纹素的纹理元素。左上角的坐标是(0,0)，v轴的正方向指向图像的正下方。

针对3D三角形上任意一点(x,y,z)处的纹理坐标(u,v)，我们通过对顶点坐标纹理进行线性插值计算而求得。

我们能够将几个并无关联的图像合为一个大的纹理图，称为纹理图集。

**9.3 纹理数据源**

对于实时图形应用程序来说，DDS图像文件格式是一种上佳的选择：除了支持GPU可原生处理的各种图像格式，它还支持一些GPU自身就可以解压缩的压缩图像格式。

经压缩后的DDS纹理只能用于输入到渲染流水线中的着色器阶段，而不能作为渲染目标。

由于块压缩算法要以4x4的像素块为基础进行处理，所以纹理的尺寸必须是4的倍数。

有两种可以将传统图像格式转换为DDS格式的方法：

1，NVIDIA公司为PS提供了一款可以将图像导出为DDS格式的插件；

2，微软公式提供了一个名为textconv的命令行工具。

**9.4 创建以及启用纹理**

**9.4.1 加载DDS文件**

加载DDS文件：DirectX::CreateDDSTextureFromFile12。

通过填写**D3D12\_SHADER\_RESOURCE\_VIEW\_DESC**对象来描述SRV描述符。

通过根签名把着色器资源视图构成的描述符表绑定到流水线槽位上。

由于纹理图集可以在一次绘制调用中渲染出多个几何体，因此可以将它用于优化性能。

**9.4.2 着色器资源视图堆**

创建了纹理资源后，我们还需要为它再创建一个SRV描述符，并将其设置到一个根签名参数槽，以供着色器程序使用。为此，首先要创建描述符堆。

**9.4.3 创建着色器资源视图描述符**

我们通过填写**D3D12\_SHADER\_RESOURCE\_VIEW\_DESC**对象来描述SRV描述符。

**9.4.4 将纹理绑定到流水线**

从纹理中提取此像素的漫反射反照率：

gDiffuseMap.Sample(gsamAnisotropicWrap, pin.TexC);

**9.5 过滤器**

**9.5.1 放大**

纹理放大时，图形硬件往往会支持常数插值和线性插值两种插值方法。在实践中，线性插值使用更为普遍。

在1D情况下：假设我们有一个内含256个样本的1D纹理，并且某个插值纹理坐标为u=0.126484375，所以归一化纹理坐标就对应于0.126484375x256=32.38处的纹素。

2D线性插值又称为双线性插值，其处理流程：给出四个纹素之间的一个纹理坐标，先在水平方向u上进行两次线性插值，后在垂直方向v上再进行一次插值计算。

纹理放大是个无法回避的问题，在与目标保持特定距离时，纹理可能看上去还不错，但是随着观察点逐渐接近目标，其效果开始惨不忍睹。

在纹理这一语境中，通过常数插值来求得纹素之间纹理坐标处的纹理数据也称为点过滤。为了求取纹素之间纹理坐标处的纹理数据而使用线性插值的计算方法，也称为线性过滤。

**9.5.2 缩小**

在初始化期间，通过对图像下采样来创建mipmap链便可制作出缩小版的纹理。

在运行时，图形硬件将根据程序员的设定，从以下两种不同的执行方案中择一而行：

1，选择与待投影到屏幕上的几何体分辨率最为匹配的mipmap层级，并根据具体需求选用常数插值或线性插值。这便是针对mipmap的点过滤。

2，选取与待投影到屏幕上的结合体分辨率最为匹配的两个相邻的mipmap层级。对这两种mipmap层级分别应用常量过滤和线性过滤，最后，在这两种插值纹理之间再次进行颜色的插值计算。这个过程称为mipmap的线性过滤。

**9.5.3 各向异性过滤**

各向异性过滤有助于缓解当多边形法向量与摄像机观察向量之间夹角过大(比如当多边形正交于观察窗口时)所导致的失真现象。这种过滤器的开销最大，但是其校正失真的效果的确对得起它所消耗的资源。

**9.6 寻址模式**

1，重复寻址模式：通过在坐标的每个整数点处重复绘制图像来扩充纹理函数。

2，边框颜色寻址模式：通过将每个不在范围[0,1]2内的坐标(u,v)都映射为程序员指定的颜色而扩充纹理函数。

3，钳位寻址模式：通过将范围[0,1]外的每个坐标(u,v)都映射为颜色T(u0,v0)来扩充纹理函数，其中(u0,v0)为范围[0,1]2内距离(u,v)最近的点。

4，镜像寻址模式：通过在坐标的每个整数点处绘制图像的镜像来扩充纹理函数。

在程序中总是要指定一种寻址模式的，默认为重复寻址模式。

**9.7 采样器对象**

在运用纹理的过程中，除了纹理数据本身之外，还有另外两个相关的重要概念，即纹理过滤以及寻址模式。这些都是由采样器对象来定义的。一个应用程序通常需要采用若干个采样器对象以不同的方式来采集纹理。

**9.7.1 创建采样器**

采样器会被着色器所用。为了将采样器绑定到着色器上供其使用，我们就需要为采样器对象绑定描述符。可以通过根签名来使用采样器描述符。

如果要设置采样器描述符，还需一个采样器堆。而要创建采样器堆，就应通过填写D3D12\_DESCRIPTOR\_HEAP\_DESC结构体实例并将其堆类型指定为D3D12\_DESCRIPTOR\_HEAP\_TYPE\_SAMPLER。

有了采样器堆，就能创建采样器描述符了。此时，我们再通过填写**D3D12\_SAMPLER\_DESC**对象来指定寻址模式，过滤器类型以及其他一些参数。

**9.7.2 静态采样器**

事实证明，图形应用程序通常不会使用过多的采样器。为此，Direct3D专门提供了一种特殊的方式来定义采样器数组，使用户可以在不创建采样器堆的情况下也能对它们进行配置。

CD3DX12\_ROOT\_SIGNATURE\_DESC类有两种参数不同的Init函数，用户可以借此为应用程序定义所用的静态采样器数组。我们通过结构体D3D12\_STATIC\_SAMPLE\_DESC来描述静态采样器，它与D3D12\_SAMPLER\_DESC比较相似，但有一些区别：

1. 边框颜色存在一些限制；
2. 含有额外的字段用来指定着色器寄存器，寄存器空间以及着色器的可见性，这些其实都是配置采样器堆的相关参数。

静态采样器数组通过CD3DX12\_ROOT\_SIGNATURE\_DESC进行绑定。

**9.8 在着色器中对纹理进行采样**

通过下列HLSL语法来定义纹理对象，并将其分配给特定的纹理寄存器：

Texture2D gDiffuseMap : register(t0);

纹理寄存器由tn来标定，其中，整数n表示的是纹理寄存器的槽号。

定义采样器对象，并将它们分别分配到特定的采样器寄存器。

SamplerState gsamPointWrap : register(s0);

采样器寄存器由sn来指定，其中整数n表示的是采样器寄存器的槽号。

通过Texture2D::Sample方法进行纹理采样。

**9.9 实践：板条箱演示程序**

9.9.1 指定纹理坐标

9.9.2 创建纹理

9.9.3 设置纹理

9.9.4 更新HLSL部分代码

**9.10 纹理变换**

常量缓冲区变量gTexTransform与gMatTransform用于在顶点着色器中对输入的纹理坐标进行变换。

**9.11 实践：附有纹理的山川演示程序**

1，通过陆地网格重复铺设草地纹理来获取更高的分辨率；

2，根据时间函数令流水线纹理沿波浪几何体滚动起来。