备忘录_纹理贴图

笔记本:	DirectX 12		
创建时间:	2022/8/16 11:24	**更新时间:**	2022/8/18 20:08
作者:	handsome小赞		
URL:	https://github.com/microsoft/DirectXTK12/blob/main/Src/DDSTextureLoader.cpp		

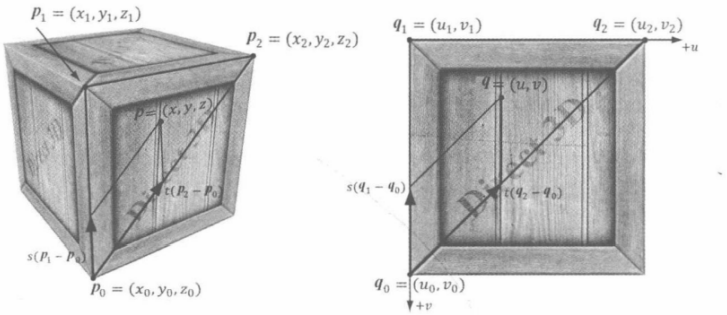
- 纹理也以ID3D12Resource来表示
- 纹理不同于缓冲区资源，因为缓冲区资源仅存储数据数组，而纹理可以有多个mipmap层级、
- 纹理可以绑定到渲染流水线的不同阶段，如作为渲染目标和作为着色器资源。
- **渲染到纹理**：将数据渲染到一个纹理后，再用它作为着色器资源
- 渲染目标：

> 需要为纹理资源创建描述符。并存于渲染目标堆中（D3D12_DESCRIPTOR_HEAP_TYPE_RTV）
>
> ```
> // 绑定为渲染目标
> CD3DX12_CPU_DESCRIPTOR_HANDLE rtv = ...;
> CD3DX12_CPU_DESCRIPTOR_HANDLE dsv = ...;
> cmdList->OMSetRenderTargets(1, &rtv, true, &dsv);
> ```

- 着色器资源：

> 需要为纹理资源创建描述符。并存于着色器资源堆（D3D12_DESCRIPTOR_HEAP_TYPE_CBV_SRV_UAV）
>
> ```
> // 以着色器输入的名义绑定到根参数
> CD3DX12_GPU_DESCRIPTOR_HANDLE tex = ...;
> cmdList->SetGraphicsRootDescriptorTable(rootParamIndex, tex);
> ```

- 纹理坐标



3D三角形　　对应的纹理三角形

图 9.3　左侧的三角形位于 3D 空间，我们将把右侧纹理上的 2D 三角形映射到左侧的 3D 三角形上

$$(x, y, z) = \boldsymbol{p} = \boldsymbol{p}_0 + s(\boldsymbol{p}_1 - \boldsymbol{p}_0) + t(\boldsymbol{p}_2 - \boldsymbol{p}_0)$$

当 $s \geqslant 0, t \geqslant 0, s + t \leqslant 1$ 时，那么，

$$(u, v) = \boldsymbol{q} = \boldsymbol{q}_0 + s(\boldsymbol{q}_1 - \boldsymbol{q}_0) + t(\boldsymbol{q}_2 - \boldsymbol{q}_0)$$

依此方法便可求出三角形上每个点处的对应纹理坐标。

```
struct Vertex
{
  DirectX::XMFLOAT3 Pos;
  DirectX::XMFLOAT3 Normal;
  DirectX::XMFLOAT2 TexC;
};

std::vector<D3D12_INPUT_ELEMENT_DESC> mInputLayout =
{
  { "POSITION", 0, DXGI_FORMAT_R32G32B32_FLOAT, 0, 0,
   D3D12_INPUT_CLASSIFICATION_PER_VERTEX_DATA, 0 },
  { "NORMAL", 0, DXGI_FORMAT_R32G32B32_FLOAT, 0, 12,
   D3D12_INPUT_CLASSIFICATION_PER_VERTEX_DATA, 0 },
  { "TEXCOORD", 0, DXGI_FORMAT_R32G32_FLOAT, 0, 24,
   D3D12_INPUT_CLASSIFICATION_PER_VERTEX_DATA, 0 },
};
```

- 纹理图集
  将几个无关联的图像合并为一个大的纹理图，再将它应用于若干不同的物体，此时纹理坐标用于确定纹理的哪一部分将被映射到目标三角形上。

- 纹理数据源

DDS概述:

- 它是专门为GPU设计的图像格式
- DDS纹理满足用于3D图形开发的以下特征
  1. mipmap
  2. GPU能自行解压的压缩格式
  3. 纹理数组
  4. 立方体图
  5. 体纹理（体积纹理，立体纹理）

DDS 格式能够支援不同的像素格式。像素格式由枚举类型 DXGI_FORMAT 中的成员来表示，但是并非所有的格式都适用于 DDS 纹理。非压缩图像数据一般会采用下列格式。

1. DXGI_FORMAT_B8G8R8A8_UNORM 或 DXGI_FORMAT_B8G8R8X8_UNORM：适用于低动态范围（low-dynamic-range）图像。
2. DXGI_FORMAT_R16G16B16A16_FLOAT：适用于高动态范围（high-dynamic-range）图像。

随着虚拟场景中纹理数量的大幅增长，对 GPU 端显存的需求也在迅速增加（还记得吗，我们需要将所有的纹理都置于显存当中，以便在程序中快速地运用这些资源）。为了缓解这些内存的需求压力，Direct3D 支持下列几种压缩纹理格式（也称作块压缩，block compression）。

1. BC1（DXGI_FORMAT_BC1_UNORM）：如果我们需要将图片压缩为支持 3 个颜色通道和仅有 1 位（开/关）alpha 分量的格式，则使用此格式。
2. BC2（DXGI_FORMAT_BC2_UNORM）：如果我们需要将图片压缩为支持 3 个颜色通道和仅有 4 位 alpha 分量的格式，则应用此格式。
3. BC3（DXGI_FORMAT_BC3_UNORM）：如果我们需要将图片压缩为支持 3 个颜色通道和 8 位 alpha 分量的格式，则采用此格式。
4. BC4（DXGI_FORMAT_BC4_UNORM）：如果我们需要将图片压缩为仅含有 1 个颜色通道的格式（如灰度图像），则运用此格式。
5. BC5（DXGI_FORMAT_BC5_UNORM）：如果我们需要将图片压缩为只支持 2 个颜色通道的格式，则使用此格式。
6. BC6（DXGI_FORMAT_BC6H_UF16）：如果我们需要将图片压缩为 HDR（高动态范围）图像数据，则应用此格式。
7. BC7（DXGI_FORMAT_BC7_UNORM）：此格式用于对 RGBA 数据进行高质量的压缩。特别是，此格式可极大地减少因压缩法线图而造成的误差。

- **注意** 经压缩后的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
- **注意** 由于块压缩算法要以4x4的像素块为基础进行处理，所以纹理必须为4的倍数

创建DDS文件:

- PS插件
- texconv命令行工具
  最新版的 texconv 和 texassemble 工具可以从微弱GitHub上名为 DirectXTex 的工程中找到

1. NVIDIA 公司为 Adobe Photoshop 提供了一款可以将图像导出为 DDS 格式的插件。该插件现存于 https://developer.nvidia.com/nvidia-texture-tools-adobe-photoshop。此插件还有一些其他选项，可供用户指定 DDS 文件的 DXGI_FORMAT 格式，或生成 mipmap 等。
2. 微软公司提供了一个名为 texconv 的命令行工具，该工具能将传统的图像格式转换为 DDS 文件。另外，*texconv* 程序还有更多的其他功能，如调整图像大小、改变像素格式、生成 mipmap 等。可以在网站 https://directxtex.codeplex.com/wikipage?title=Texconv&referringTitle= ocumentation 找到它的文档与下载链接。

- 创建以及启用纹理

加载DDS文件

- GitHub: microsoft/DirectXTK12 : DDSTextureLoader.h/cpp 专门为 DirectX12提供了一个新的读取DDS文件的方法。
  新的DDSTextureLoader提供了批处理工具
- 创建着色器资源描述符堆
  D3D12_DESCRIPTOR_HEAP_TYPE_CBV_SRV_UAV 类型
  D3D12_DESCRIPTOR_HEAP_FLAG_SHADER_VISIBLE 着色器可见
- 创建着色器资源描述符　D3D12_SHADER_RESOURCE_VIEW_DESC
- 将纹理绑定到流水线
  - 添加漫反射反照率纹理图 DiffuseAlbedoTextureMap

- 过滤器

> **放大**
>
> - 点过滤：通过常数插值来求得纹素之间纹理坐标处的纹理数据
> - 线性过滤：通过线性插值来求得纹素之间纹理坐标处的纹理数据
>
> **缩小**
>
> - 针对mipmap的点过滤：在纹理贴图时，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。
> - 针对mipmap的线性过滤：在纹理贴图时，选择与待投影到屏幕上的几何体分辨率最为匹配的两个临近mipmap层级（一个大于屏幕上几何体的分辨率，一个小于），接下来对这两种mipmap层级分别应用常量过滤或象形过滤，以生成它们各自相应的纹理颜色。最后，在这两种插值纹理之间再进行颜色的插值计算。
>
> **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所导致的失真现象。这种过滤器开销最大，但是其矫正失真的效果的确对得起它所消耗的资源。

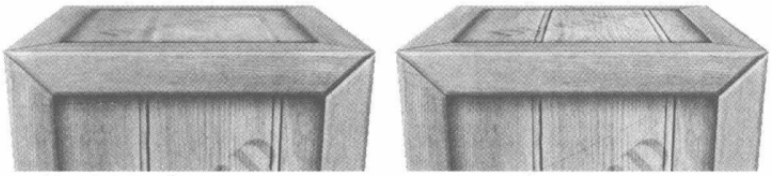


图 9.9　板条箱的顶面基本上已正交于观察窗口。左图中的板条箱顶面采用了线性过滤，其效果模糊得一塌糊涂。右图以同样的角度观察通过各向异性过滤绘制的板条箱顶面，却呈现出细节更佳的渲染效果

- 寻址模式

> - 寻址模式由枚举类型 `D3D12_TEXTURE_ADDRESS_MODE` 来表示
> - 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
> - 边框颜色寻址模式：通过将每个不在范围[0, 1]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
> - 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]内距离(u, v)最近的点
> - 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

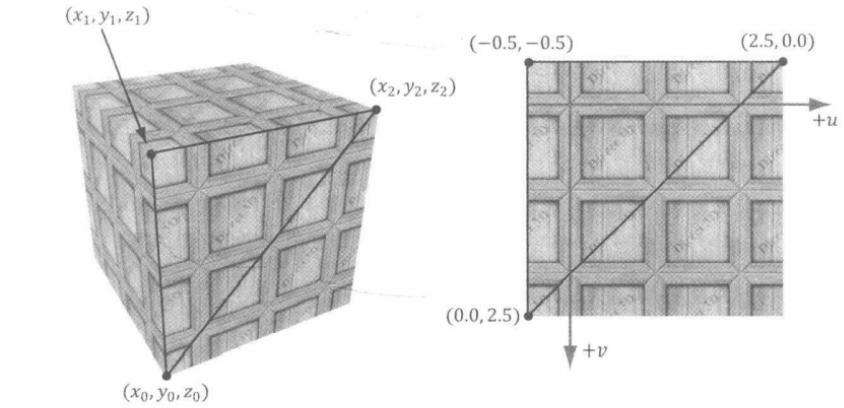


图 9.10　重复寻址模式

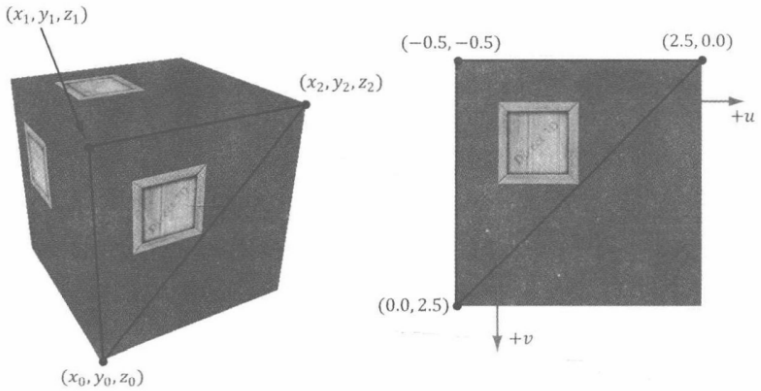


图 9.11　边框颜色寻址模式

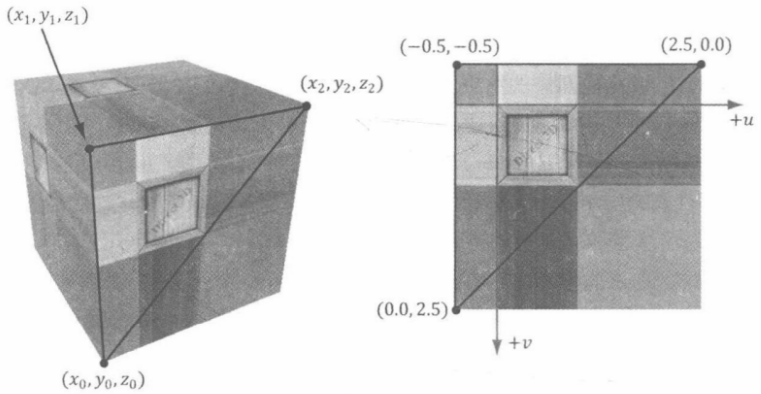


图 9.12　钳位寻址模式

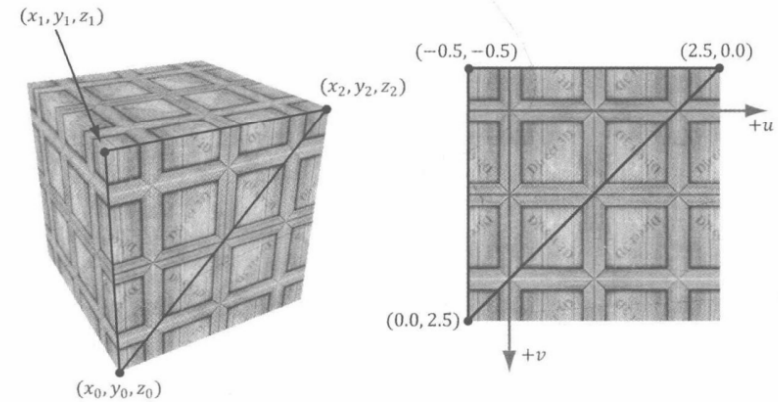


图 9.13　镜像寻址模式

- 采样器对象

  - 采集纹理资源时，**过滤器**和**寻址模式**都是由**采样器对象**定义的

  创建采样器

  1. 创建描述符表
  2. 初始化为采样器描述符表
     `D3D12_DESCRIPTOR_RANGE_TYPE_SAMPLER`
  3. 创建根参数
  4. 创建根签名
  5. 创建采样器描述符堆，`D3D12_DESCRIPTOR_HEAP_DESC`，将Type指定为
     `D3D12_DESCRIPTOR_HEAP_TYPE_SAMPLER`

6. 创建采样器描述符，D3D12_SAMPLER_DESC

静态采样器

- 用户可以在不创建采样器堆的情况下也能对采样器数组进行配置
- 通过 D3D12_STATIC_SAMPLER_DESC 来描述静态采样器
  与 D3D12_SAMPLER_DESC 不同的是：
  1. 边框颜色存在一些限制，即静态采样器的边框颜色必须为 enum D3D12_STATIC_BORDER_COLOR 的成员之一
  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。
  3. 用户只能定义2032个静态采样器。
- 通过 CD3DX12_ROOTSIGNATURE_DESC 构造函数的第3，4个参数设置

- 在着色器中堆纹理进行采样

  - 在HLSL文件定义纹理对象

    ```
    Texture2D gDiffuseMap : register(t0);
    ```

  - 同理，定义多个采样器对象

    ```
    SamplerState gsamPointWrap     : register(s0);
    SamplerState gsamPointClamp    : register(s1);
    SamplerState gsamLinearWrap    : register(s2);
    SamplerState gsamLinearClamp   : register(s3);
    SamplerState gsamAnisotropicWrap : register(s4);
    SamplerState gsamAnisotropicClamp : register(s5);
    ```

  - 通过 Texture2D::Sample() 进行纹理采样，这个方法将利用 SamplerState 对象指定的过滤方法，返回纹理图在点（u,v）处的插值颜色

    ```
    struct VertexOut
    {
      float4 PosH  : SV_POSITION;
      float3 PosW  : POSITION;
      float3 NormalW : NORMAL;
      float2 TexC  : TEXCOORD;
    };

    float4 PS(VertexOut pin) : SV_Target
    {
      float4 diffuseAlbedo = gDiffuseMap.Sample(gsamAnisotropicWrap, pin.TexC)
        * gDiffuseAlbedo;
      ...
    ```