



唐老狮系列教程

深度和法线纹理 背后的原理

WELCOME
TO THE
UNITY
SPECIALTY COURSE
STUDY

版权所有：唐老狮 tpandme@163.com



唐老狮系列教程-深度和法线纹理背后的原理

主要讲解内容



唐老狮系列教程-深度和法线纹理背后的原理

主要讲解内容

1. 知识回顾 裁剪空间变换
2. 深度和法线纹理存储的是什么信息
3. Unity是如何得到深度和法线纹理的
4. 深度和法线纹理使用时调用的函数原理



唐老狮系列教程-深度和法线纹理背后的原理

| 知识回顾 裁剪空间变换



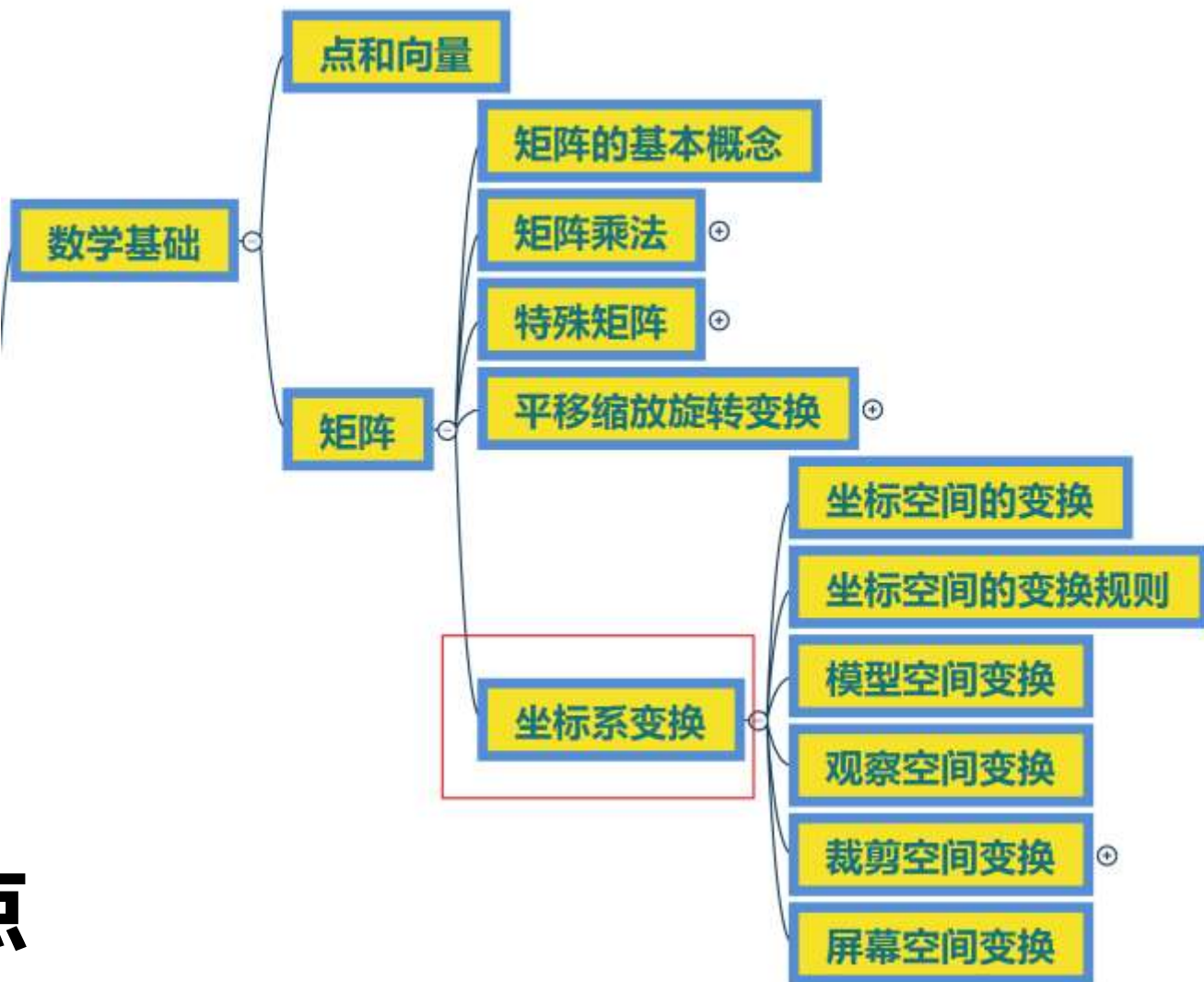
唐老狮系列教程-深度和法线纹理背后的原理

裁剪空间变换

我们之前在学习矩阵相关知识时，花了很多时间去学习坐标系的变换。
在渲染管线中，顶点、法线等相关的模型数据会经过以下空间变换

模型空间→世界空间→观察空间→裁剪空间→屏幕空间

其中我们需要着重复习一下 **观察空间→裁剪空间**



观察空间：也被称为摄像机空间，遵循右手坐标系原则，观察空间中原点为摄像机位置，x、y、z轴正方向分别对应摄像机的右、上、后方

裁剪空间：也被称为齐次裁剪空间，是一个非常特殊的空间，它的坐标范围为(-1,-1,-1)到（1,1,1）的一个正方体范围，我们需要将摄像机视锥体变换到其中

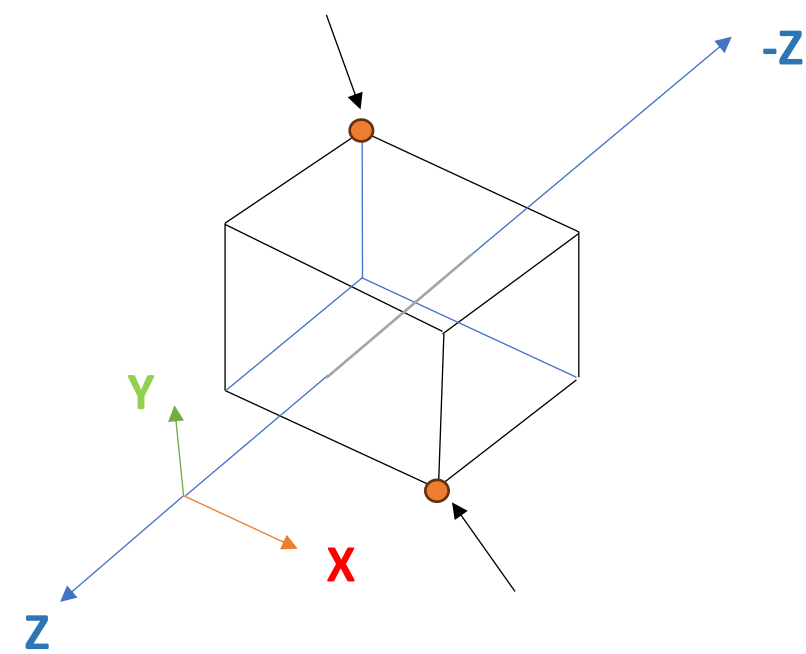


唐老狮系列教程-深度和法线纹理背后的原理

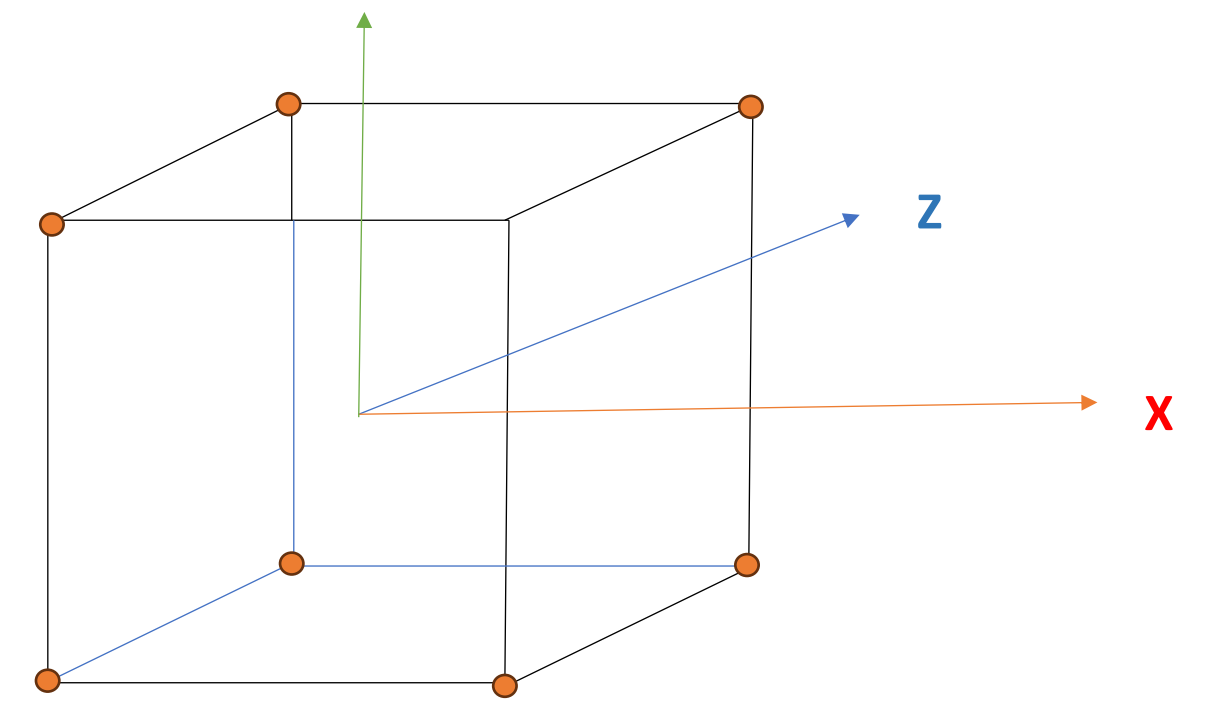
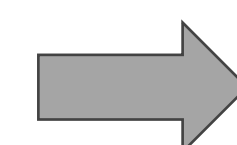
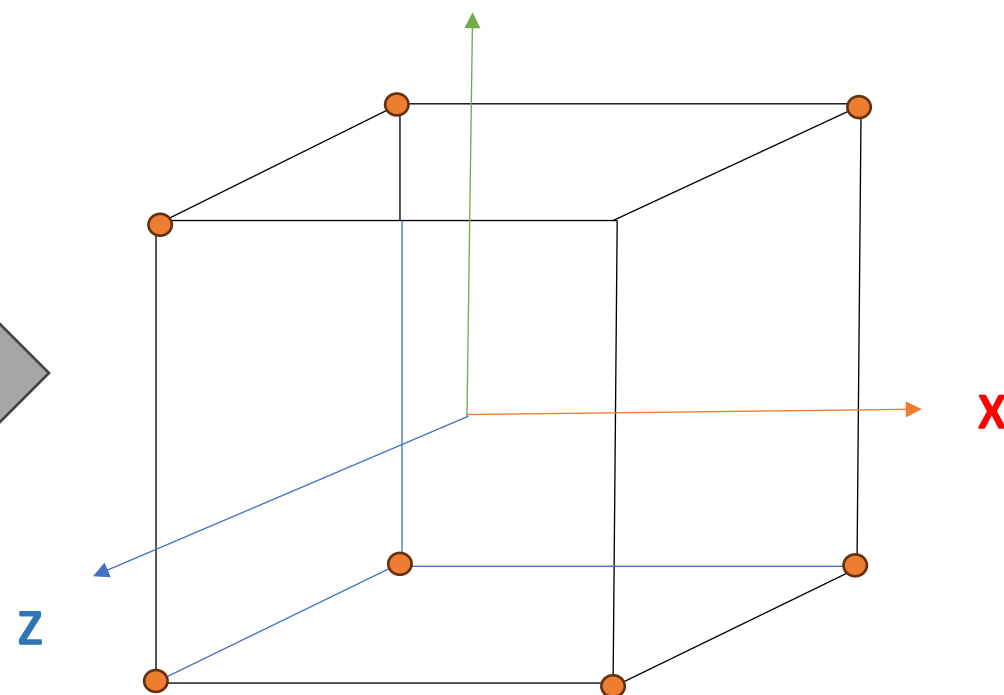
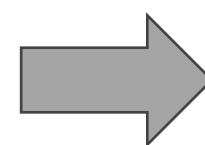
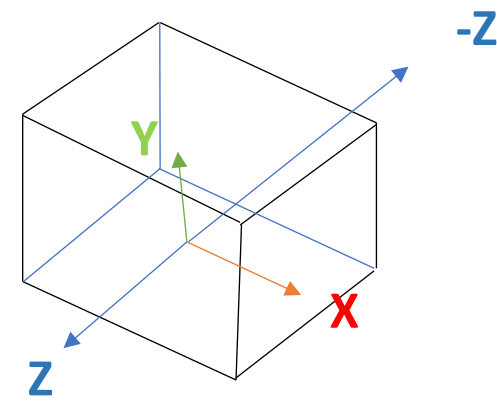
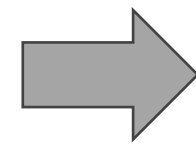
坐标空间的变换

其中相对复杂的是 **摄像机的 透视投影** 和 **正交投影** 到 **齐次裁剪空间** 的 **变换矩阵推导**

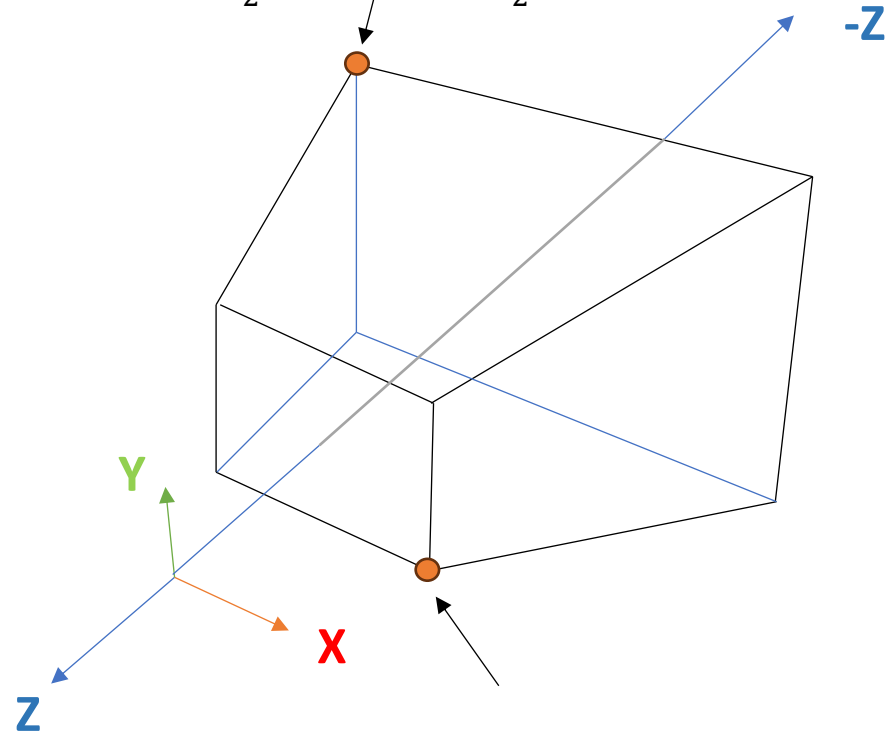
$(-Aspect * Size, Size, -Far)$



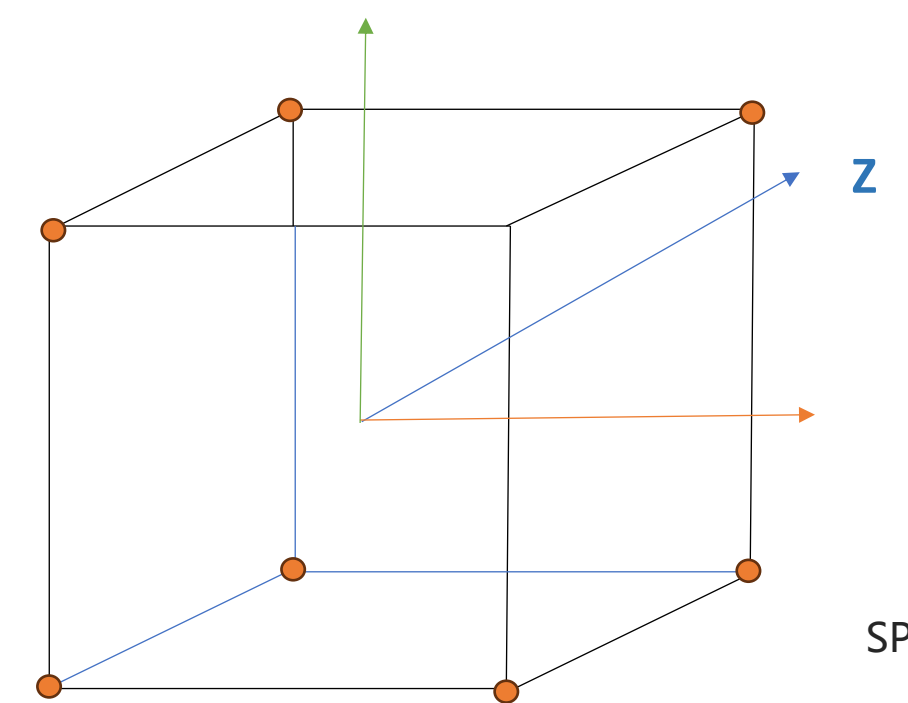
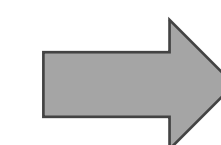
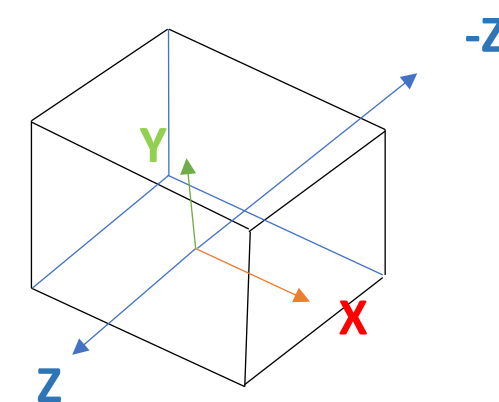
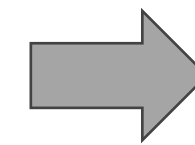
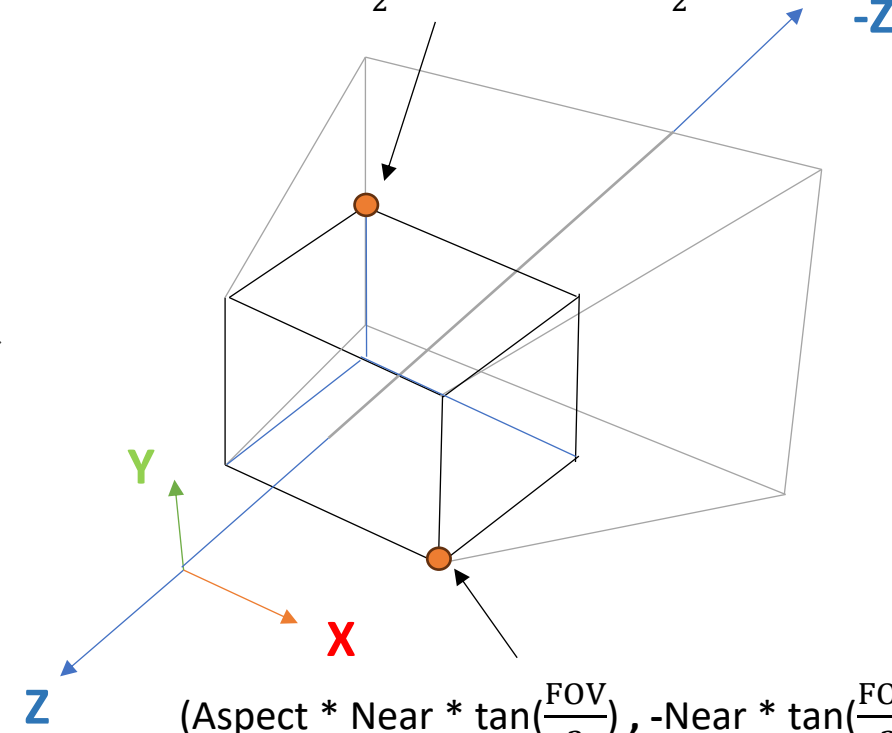
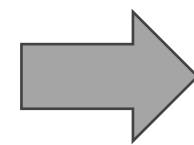
$(Aspect * Size, -Size, -Near)$



$(-Aspect * Far * \tan(\frac{FOV}{2}), Far * \tan(\frac{FOV}{2}), -Far)$



$(-Aspect * Near * \tan(\frac{FOV}{2}), Near * \tan(\frac{FOV}{2}), -Far)$



WELCOME
TO THE
UNITY
SPECIALTY COURSE
STUDY

版权所有：唐老狮 tpandme@163.com



唐老狮系列教程-深度和法线纹理背后的原理

坐标空间的变换

$$\begin{matrix} \frac{1}{Aspect * Size} & 0 & 0 & 0 \\ 0 & \frac{1}{Size} & 0 & 0 \\ 0 & 0 & -\frac{2}{Far - Near} & -\frac{Far + Near}{Far - Near} \\ 0 & 0 & 0 & 1 \end{matrix}$$

正交投影齐次裁剪坐标的最终变换矩阵

$$\begin{matrix} \frac{1}{Aspect * \tan(\frac{FOV}{2})} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan(\frac{FOV}{2})} & 0 & 0 \\ 0 & 0 & -\frac{Far + Near}{Far - Near} & -\frac{2FarNear}{Far - Near} \\ 0 & 0 & -1 & 0 \end{matrix}$$

透视投影齐次裁剪坐标的最终变换矩阵

利用观察空间中的点根据使用的摄像机类型决定和哪一个变换矩阵进行矩阵乘法后
便可以得到齐次裁剪空间下的坐标信息



唐老狮系列教程-深度和法线纹理背后的原理

深度和法线纹理存储的是什么信息



唐老狮系列教程-深度和法线纹理背后的原理

深度纹理中存储的是什么信息

Unity中的深度纹理中存储的信息

也就是Shader中使用 `_CameraDepthTexture` 或 `_CameraDepthNormalsTexture` 采样的信息

是进行裁剪空间变换后的 z 分量再转换到 $0 \sim 1$ 之后的结果

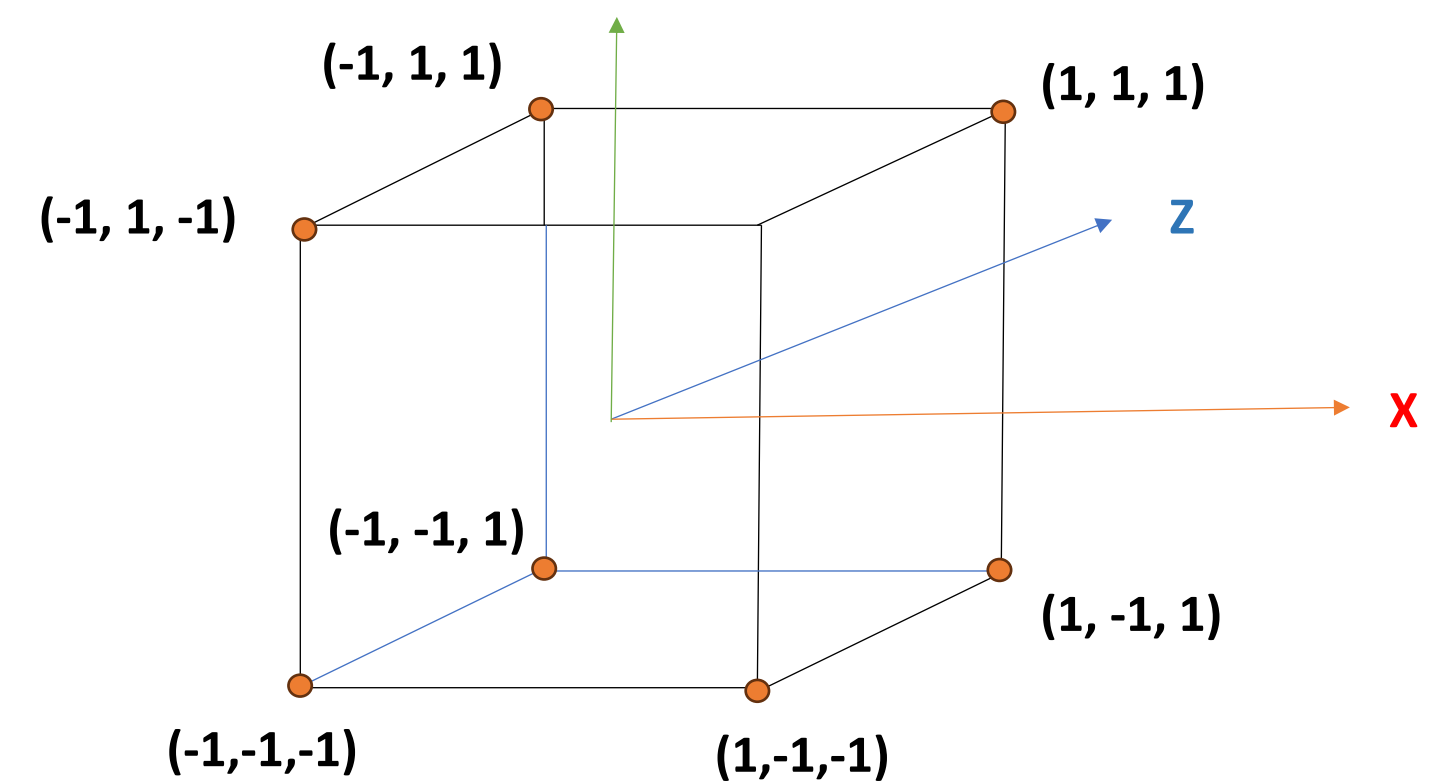
因为齐次裁剪空间坐标范围为 $-1 \sim 1$ ，而纹理中存储的信息范围是 $0 \sim 1$

因此Unity会将其利用以下公式进行转换

深度纹理值 = $0.5 * z + 0.5$

也就是说我们通过深度纹理直接采样得到的深度纹理值是

是进行裁剪空间变换后的 z 分量再转换到 $0 \sim 1$ 之后的结果





唐老狮系列教程-深度和法线纹理背后的原理

法线纹理中存储的是什么信息

Unity中的法线纹理中存储的信息

也就是Shader中使用_CameraDepthNormalsTexture采样得到的float4中的部分信息

它是观察空间下的 法线 再转换到0~1之后的结果

因为观察空间下的单位向量的分量取值范围是 $-1 \sim 1$ ，而纹理中存储的信息范围是 $0 \sim 1$

因此Unity会将其利用以下公式进行转换

法线纹理值 = (观察空间下法线 + 1) * 0.5

也就是说我们通过法线纹理直接采样得到的法线纹理值是

是观察空间下的 法线 再转换到0~1之后的结果



唐老狮系列教程-深度和法线纹理背后的原理

| Unity是如何得到深度和法线纹理的



唐老狮系列教程-深度和法线纹理背后的原理

Unity是如何得到深度和法线纹理的

深度和法线纹理的获取：

Unity中深度和法线纹理一般通过**两种**途径获取

1. 从**G-buffer几何缓冲区**中获取
2. 由一个专门的**Pass渲染**而来

具体Unity是通过哪种方式获取，取决于使用的渲染路径和设备的硬件限制。

当使用**延迟渲染路径**时，深度和法线纹理可以直接访问到，因为延迟渲染路径会把信息存储到**G-buffer几何缓冲区**中（深度和法线等信息都存储在其中）。

而当**无法直接获取**到深度和法线纹理时（比如硬件不支持延迟渲染路径 或 使用的是前向渲染路径时），Unity会**通过一个单独的Pass来进行渲染**，获取深度和法线信息。



唐老狮系列教程-深度和法线纹理背后的原理

Unity是如何得到深度和法线纹理的

需要注意的是，当使用单独的Pass渲染获得深度和法线纹理时，两者是有区别的

对于**深度纹理**来说：

Unity内部会使用**着色器替换技术**选择**渲染类型** `RenderType = "Opaque"`（不透明物体）

然后判断它们的**渲染队列** Queue是否小于等于2500（Background-1000、Geometry-2000、AlphaTest-2450）

如果**满足这个条件**，就会使用物体投射阴影时的Pass（LightMode 为 ShadowCaster 的Pass）

来得到深度纹理，如果没有这个Pass，那么该物体不会出现在深度纹理中！

因此这里的**重点是**：

如果我们希望物体能够正确的出现在深度纹理中

- 1. 必须在Shader中正确的设置RenderType标签**
- 2. 必须有投射阴影用的Pass（LightMode为ShadowCaster的Pass）**



唐老狮系列教程-深度和法线纹理背后的原理

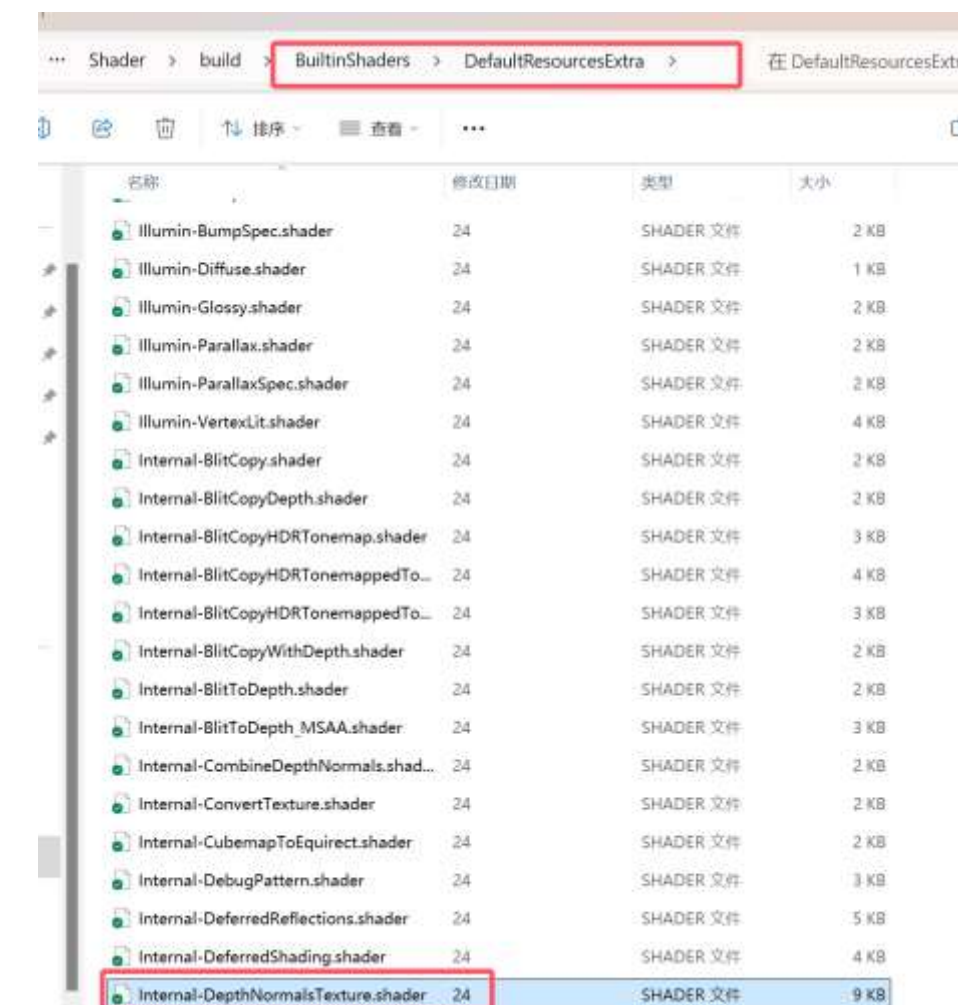
Unity是如何得到深度和法线纹理的

对于法线纹理来说：

Unity底层会使用一个单独的Pass把整个场景再次渲染一次，从而得到深度和法线信息

这里为什么是深度和法线信息呢，因为当需要得到法线纹理时，Unity中是和深度一起获取的
上节课的使用知识点中有讲解（_CameraDepthNormalsTexture）

这个Pass包含在Unity内置的Shader中，我们可以在官方下载源文件解压后进行查看





唐老狮系列教程-深度和法线纹理背后的原理

Unity是如何得到深度和法线纹理的

总体而言，在获取深度和法线纹理时

Unity内部帮助我们做了很多事情，我们不需要操心太多

我们只需要按照上节课的规则就能够获取到对应的信息了

需要注意的就是，在获取深度纹理时，一定要注意

1. 必须在Shader中正确的设置RenderType标签
2. 必须有投射阴影用的Pass (LightMode为ShadowCaster的Pass)



唐老狮系列教程-深度和法线纹理背后的原理

深度和法线纹理使用时调用的函数原理



唐老狮系列教程-深度和法线纹理背后的原理

深度和法线纹理使用时调用的函数原理

通过上节课的学习，我们知道，想要在Shader当中使用深度和法线纹理需要做以下的事情

```
private void Start()
{
    //获取深度纹理（一张纹理）
    Camera.main.depthTextureMode = DepthTextureMode.Depth;
    //获取深度+法线纹理（一张纹理）
    Camera.main.depthTextureMode = DepthTextureMode.DepthNormals;
    //获取深度纹理 和 深度+法线纹理（两张纹理）
    Camera.main.depthTextureMode = DepthTextureMode.Depth | DepthTextureMode.DepthNormals;
}
```

```
fixed4 frag (v2f i) : SV_Target
{
    // 使用基本深度纹理采样 得到的结果是非线性的
    float depth = SAMPLE_DEPTH_TEXTURE(_CameraDepthTexture, i.uv);
    // 将非线性的深度值 转换到观察空间下
    float viewDepth = LinearEyeDepth(depth);
    // 将非线性的深度值 转换到[0,1]区间内 线性深度值
    float linearDepth = Linear01Depth(depth);
```

```
sampler2D _CameraDepthTexture;
sampler2D _CameraDepthNormalsTexture;
```

```
fixed4 frag (v2f i) : SV_Target
{
    //用于存储深度值的变量
    float depth;
    //用于存储法线的变量
    float3 normals;
    //对深度+法线纹理进行采样（其中xy是法线信息，zw是深度信息）
    float4 depthNormal = tex2D(_CameraDepthNormalsTexture, i.uv);
    //UnityCG.cginc 内置文件中的方法 用于得到深度值(0~1)和法线信息(观察空间下)
    //相当于一次处理深度和法线
    DecodeDepthNormal(depthNormal, depth, normals);
    //单独得到深度
    depth = DecodeFloatRG(depthNormal.zw);
    //单独得到法线
    normals = DecodeViewNormalStereo(depthNormal);
```




唐老狮系列教程-深度和法线纹理背后的原理

深度和法线纹理使用时调用的函数原理

```
fixed4 frag (v2f i) : SV_Target
{
    // 使用基本深度纹理采样 得到的结果是非线性的
    float depth = SAMPLE_DEPTH_TEXTURE(_CameraDepthTexture, i.uv);
    // 将非线性的深度值 转换到观察空间下
    float viewDepth = LinearEyeDepth(depth);
    // 将非线性的深度值 转换到[0,1]区间内 线性深度值
    float linearDepth = Linear01Depth(depth);
}
```

SAMPLE_DEPTH_TEXTURE 宏:

它是用于从深度纹理中进行采样的宏，相比直接用tex2D进行采样，**它在内部会帮助我们适配各种不同的平台**，因为不同平台对深度纹理的采样规则会有所不同，因此我们使用它。

它采样得到的深度值是裁剪空间下的z分量转换到0~1之间的结果



唐老狮系列教程-深度和法线纹理背后的原理

深度和法线纹理使用时调用的函数原理

```
fixed4 frag (v2f i) : SV_Target
{
    // 使用基本深度纹理采样 得到的结果是非线性的
    float depth = SAMPLE_DEPTH_TEXTURE(_CameraDepthTexture, i.uv);
    // 将非线性的深度值 转换到观察空间下
    float viewDepth = LinearEyeDepth(depth);
    // 将非线性的深度值 转换到[0,1]区间内 线性深度值
    float linearDepth = Linear01Depth(depth);
}
```

通过SAMPLE_DEPTH_TEXTURE得到的深度值是**非线性**的

所谓的非线性值的是指在透视摄像机的裁剪空间中**深度值分部不均匀**

当深度值接近裁剪面近时，深度值变化迅速，精度高

当深度值接远裁剪面近时，深度值变化缓慢，精度低

更直观的解释：一个相机在观察一个3D场景时，近处的物体移动一点，视觉上变化很大，所以需要更高的精度来记录这种变化。而远处的物体移动同样的距离，视觉上的变化很小，因此可以使用较低的精度来记录



唐老狮系列教程-深度和法线纹理背后的原理

深度和法线纹理使用时调用的函数原理

```
fixed4 frag (v2f i) : SV_Target
{
    // 使用基本深度纹理采样 得到的结果是非线性的
    float depth = SAMPLE_DEPTH_TEXTURE(_CameraDepthTexture, i.uv);
    // 将非线性的深度值 转换到观察空间下
    float viewDepth = LinearEyeDepth(depth);
    // 将非线性的深度值 转换到[0,1]区间内 线性深度值
    float linearDepth = Linear01Depth(depth);
}
```

因此为了让我们在Shader中利用深度值进行的计算更加准确，我们需要获得线性的深度值

只需要把裁剪空间下的深度值转换到观察空间下，便可以得到线性的深度值

Unity Shader中提供了内置函数

LinearEyeDepth 和 Linear01Depth 都可以得到观察空间下的线性深度值

LinearEyeDepth: 得到的是像素到摄像机的实际距离

Linear01Depth: 得到的是实际距离被压缩到0~1之间的值



唐老狮系列教程-深度和法线纹理背后的原理

深度和法线纹理使用时调用的函数原理

```
fixed4 frag (v2f i) : SV_Target
{
    //用于存储深度值的变量
    float depth;
    //用于存储法线的变量
    float3 normals;
    //对深度+法线纹理进行采样 (其中xy是法线信息, zw是深度信息)
    float4 depthNormal = tex2D(_CameraDepthNormalsTexture, i.uv);
    //UnityCG.cginc 内置文件中的方法 用于得到深度值(0~1)和法线信息(观察空间下)
    //相当于一次处理深度和法线
    DecodeDepthNormal(depthNormal, depth, normals);
    //单独得到深度
    depth = DecodeFloatRG(depthNormal.zw);
    //单独得到法线
    normals = DecodeViewNormalStereo(depthNormal);
}
```

DecodeDepthNormal函数内部其实也是执行的**DecodeFloatRG**和**DecodeViewNormalStereo**函数

它的作用就是得到**观察空间下**的对应像素的 **法线** 和 **线性 深度值**(0~1)

你可以一次性的获得两个信息，也可以选择分别调用**DecodeFloatRG**和**DecodeViewNormalStereo**

单独获取深度和法线信息

函数中具体做的事情，就是利用法线的xy算出z，得到最终的法线信息；

将裁剪空间下的非线性深度值 转换为观察空间下线性的范围为0~1的深度值



唐老狮系列教程-深度和法线纹理背后的原理

深度和法线纹理使用时调用的函数原理

通过对这些宏和函数的介绍，大家只要记住，直接采样出来的深度和法线信息是不会直接使用的

我们需要将他们**通过内置函数进行转换**

得到最终我们会使用的**观察空间下的深度和法线信息**

```
private void Start()
{
    //获取深度纹理（一张纹理）
    Camera.main.depthTextureMode = DepthTextureMode.Depth;
    //获取深度+法线纹理（一张纹理）
    Camera.main.depthTextureMode = DepthTextureMode.DepthNormals;
    //获取深度纹理 和 深度+法线纹理（两张纹理）
    Camera.main.depthTextureMode = DepthTextureMode.Depth | DepthTextureMode.DepthNormals;
}
```

```
sampler2D _CameraDepthTexture;
sampler2D _CameraDepthNormalsTexture;
```

```
fixed4 frag (v2f i) : SV_Target
{
    // 使用基本深度纹理采样 得到的结果是非线性的
    float depth = SAMPLE_DEPTH_TEXTURE(_CameraDepthTexture, i.uv);
    // 将非线性的深度值 转换到观察空间下
    float viewDepth = LinearEyeDepth(depth);
    // 将非线性的深度值 转换到[0,1]区间内 线性深度值
    float linearDepth = Linear01Depth(depth);
```

```
fixed4 frag (v2f i) : SV_Target
{
    //用于存储深度值的变量
    float depth;
    //用于存储法线的变量
    float3 normals;
    //对深度+法线纹理进行采样（其中xy是法线信息，zw是深度信息）
    float4 depthNormal = tex2D(_CameraDepthNormalsTexture, i.uv);
    //UnityCG.cginc 内置文件中的方法 用于得到深度值(0~1)和法线信息(观察空间下)
    //相当于一次处理深度和法线
    DecodeDepthNormal(depthNormal, depth, normals);
    //单独得到深度
    depth = DecodeFloatRG(depthNormal.zw);
    //单独得到法线
    normals = DecodeViewNormalStereo(depthNormal);
```



唐老狮系列教程-深度和法线纹理背后的原理

| 总结



唐老狮系列教程-深度和法线纹理背后的原理

总结

1. 深度和法线纹理存储的是什么信息

深度纹理：是裁剪空间变换后的 z 分量再转换到0~1之后的结果

法线纹理：是观察空间下的 法线 再转换到0~1之后的结果

2. Unity是如何得到深度和法线纹理的

1. 从G-buffer几何缓冲区中获取

2. 由一个专门的Pass渲染而来

深度纹理：必须在Shader中正确的设置RenderType标签（因为Unity会通过着色器替换技术处理）

必须有投射阴影用的Pass（LightMode为ShadowCaster的Pass）

法线纹理：Unity底层使用了一个单独的Pass把整个场景再次渲染一次，同时获取法线和深度信息



唐老狮系列教程-深度和法线纹理背后的原理

总结

3. 深度和法线纹理使用时调用的函数原理

SAMPLE_DEPTH_TEXTURE 宏：它是用于从深度纹理中进行采样的宏，相比直接用tex2D进行采样，
它在内部会帮助我们适配各种不同的平台

LinearEyeDepth 和 Linear01Depth 都可以得到观察空间下的线性深度值

LinearEyeDepth：得到的是像素到摄像机的实际距离

Linear01Depth：得到的是实际距离被压缩到0~1之间的值

DecodeDepthNormal函数内部其实也是执行的**DecodeFloatRG**和**DecodeViewNormalStereo函数**

它的作用就是得到**观察空间下**的对应像素的**法线和线性深度值(0~1)**



唐老狮系列教程

Thank
感谢您的聆听