**纹理采样**

**纹理采样的步骤**

1.在属性栏声明一张纹理。

Properties{  
 \_BaseMap("Texture", 2D) = "white" {}

2.使用TEXTURE2D()宏定义一张2D纹理，并使用SAMPLER（sampler+纹理名）为该纹理指定一个采样器。纹理和采样器是着色器资源，必须在全局定义，不能放入缓冲区中。除此之外还需要获取纹理的平铺和偏移值，这是通过定义一个float4类型的纹理名\_ST属性来获取的，该属性可以在UnityPerMaterial缓冲区中定义，设置给每个对象实例。

TEXTURE2D(\_BaseMap);  
SAMPLER(sampler\_BaseMap);   
 UNITY\_INSTANCING\_BUFFER\_START(UnityPerMaterial)  
 UNITY\_DEFINE\_INSTANCED\_PROP(float4, \_BaseMap\_ST)  
 UNITY\_DEFINE\_INSTANCED\_PROP(float4, \_BaseColor)  
UNITY\_INSTANCING\_BUFFER\_END(UnityPerMaterial)

3. 要采样纹理，我们还需要一套UV坐标，它应该被定义在顶点输入结构体中，纹理坐标要传到片元函数中进行采样，所以片元输入结构体也要定义UV坐标。

//用作顶点函数的输入参数  
struct Attributes {  
 float2 baseUV : TEXCOORD0;  
};  
//用作片元函数的输入参数  
struct Varyings {  
 float2 baseUV : VAR\_BASE\_UV;  
};

4. 在顶点函数中，传递纹理坐标之前把为纹理的缩放和偏移也计算在内。

//顶点函数  
Varyings UnlitPassVertex(Attributes input){  
 //计算缩放和偏移后的UV坐标  
 float4 baseST = UNITY\_ACCESS\_INSTANCED\_PROP(UnityPerMaterial, \_BaseMap\_ST);  
 output.baseUV = input.baseUV \* baseST.xy + baseST.zw;  
 return output;  
}

5. 最后我们将片元函数通过SAMPLE\_TEXTURE2D宏对纹理采样，采样结果和颜色值相乘得到最终表面颜色。

//片元函数  
float4 UnlitPassFragment (Varyings input) : SV\_TARGET {  
 UNITY\_SETUP\_INSTANCE\_ID(input);  
 float4 baseMap = SAMPLE\_TEXTURE2D(\_BaseMap, sampler\_BaseMap, input.baseUV);  
 float4 baseColor = UNITY\_ACCESS\_INSTANCED\_PROP(UnityPerMaterial, \_BaseColor);  
 return baseMap \* baseColor;  
}