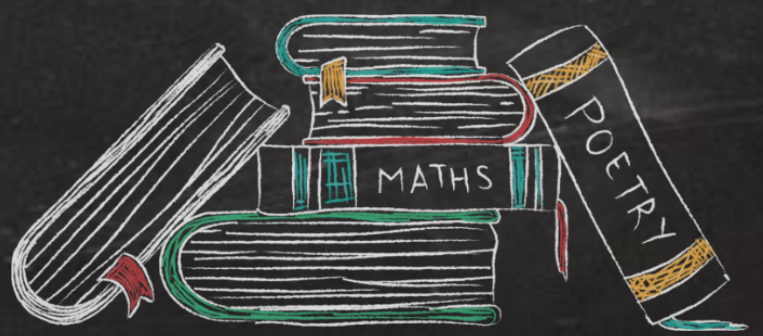

大家好
欢迎加入LightDir（光向）研习社
欢迎大家一同探索开源共享的知识分享模式

今日内容

- 回顾·渲染流程
- 符文·筑基·空色
- 符文·筑基·照
- 结点·常量组&参数组
- 作业

$\frac{24 \times 5}{33}$

1 回顾·渲染流程



01 渲染流程

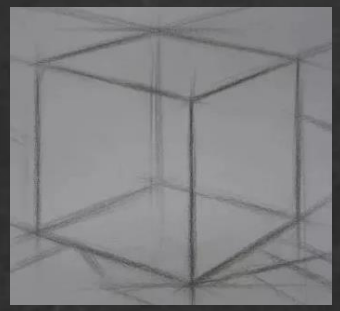
模型 → 输入结构 → 顶点 Shader → 输出结构 → 像素 Shader → 渲染结果

石膏正方体教具：

理解其为一个信息载体，眼睛收集信息，脑子对其过滤处理，手笔输出到纸；

考虑哪些信息有用：

1. 几何构成：8顶点，12条边，6个面；(即：顶点位置，连接关系与面朝向，必要，输入脑中)；
2. 颜色：老师让视为纯白，污迹，商标等细节不描绘；(即：每个位置的细节颜色不做考虑，不必要，脑中过滤)；
3. 重量，价格，等无关因素，通通脑中过滤；



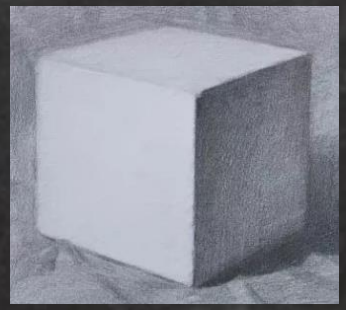
脑内材料：几何构成；

思考方法：透视理论；

输出：纸上符合透视关系的大型；

将透视变换后的几何构成信息推广至块面：

比如：已知一个面的朝向为左35度，那么所在整个块面上所有位置朝向均为左35度（前提：平面）；

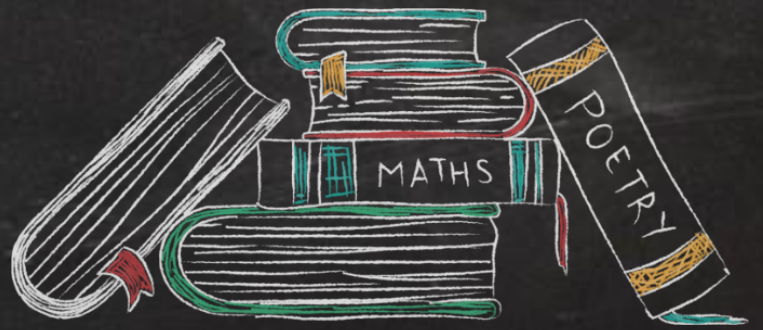


根据环境信息，块面信息；深入绘制各个块面的素描关系；

如果你画完还习惯用手指，纸巾糊几下，吾愿称之为后处理。

$\frac{24 \times 5}{33}$

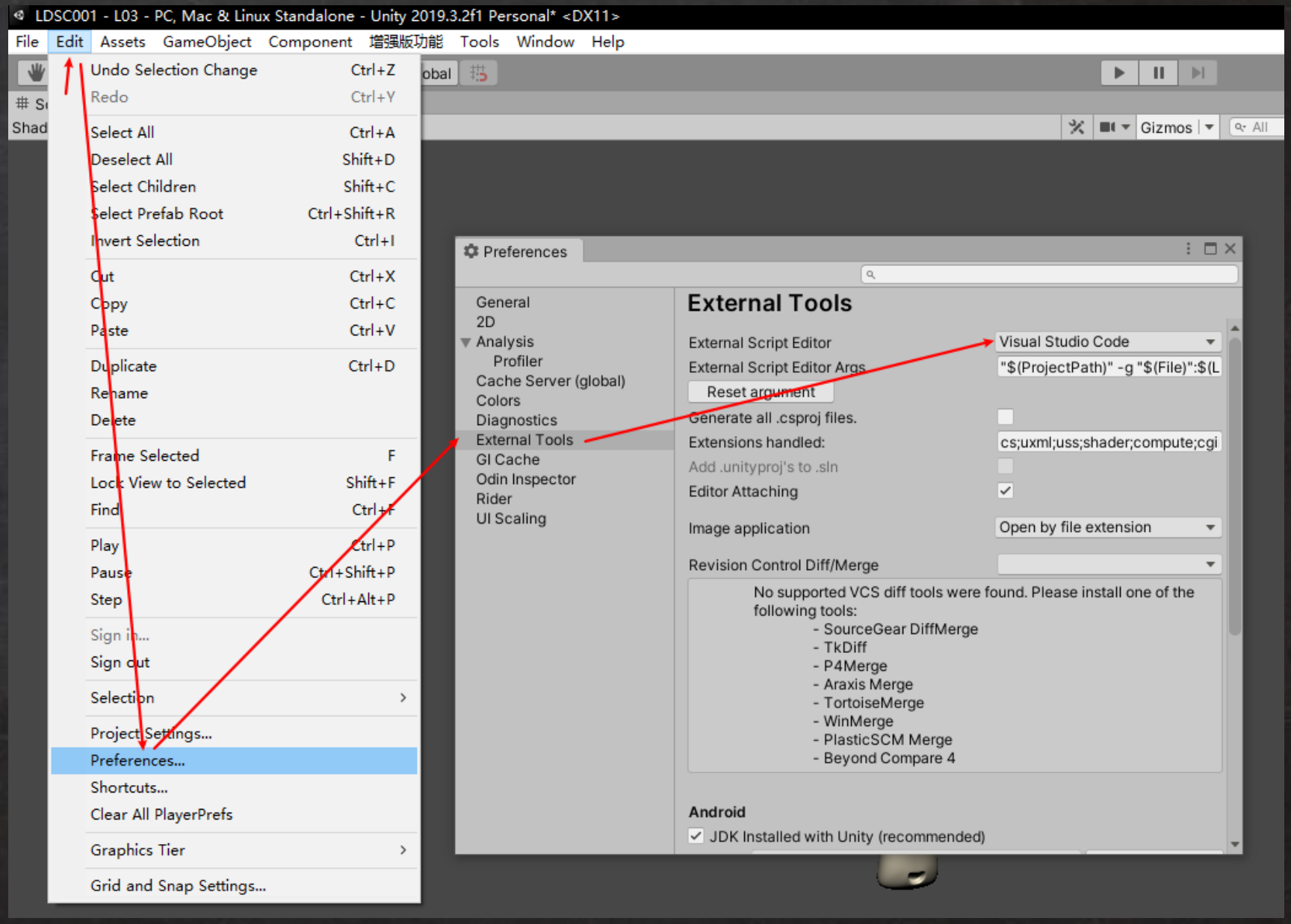
2 符文·筑基·空色



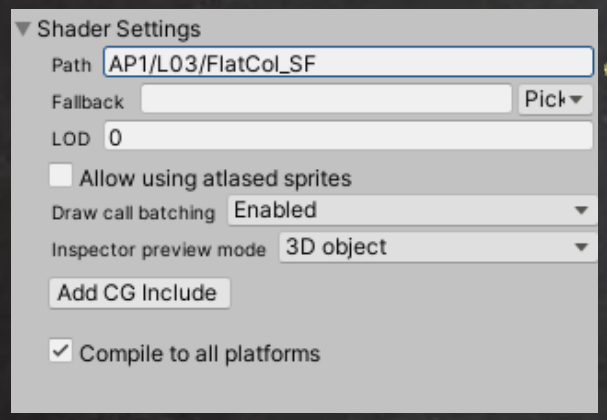
01 关联编辑器

如何关联：

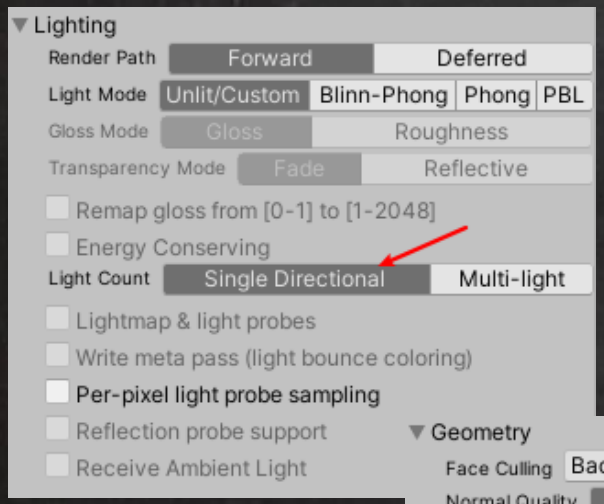
1. 菜单栏
2. Edit子菜单
3. Preferences弹窗
4. ExternalTools项
5. ExternalScriptEditor子项
6. 设置为:VisulStudioCode

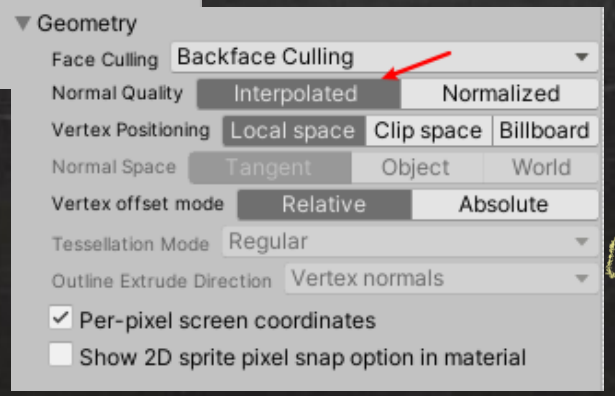


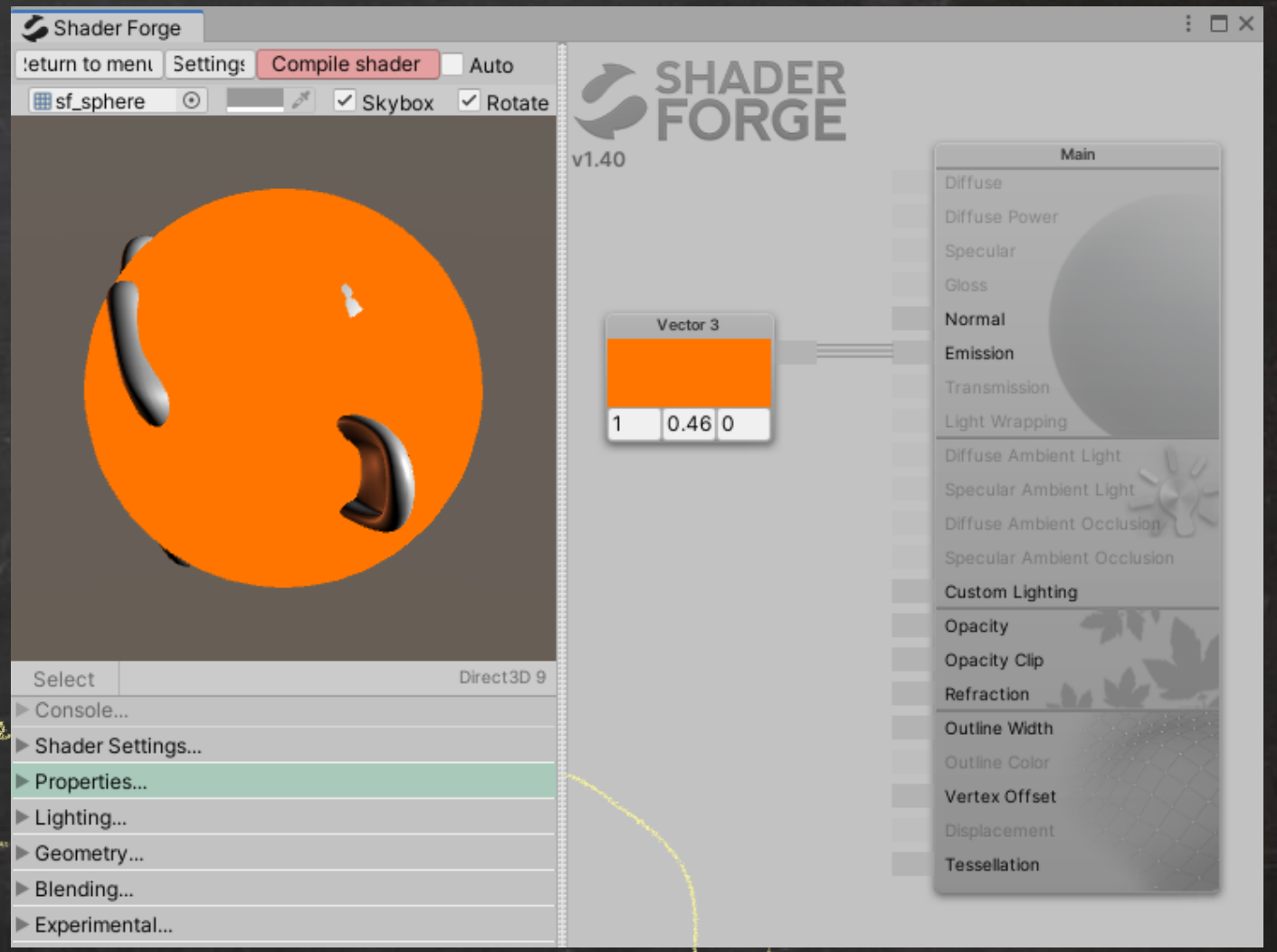
02 SF新建最简Shader模板

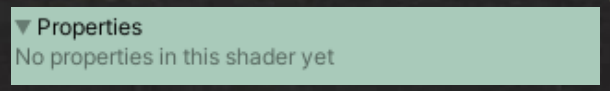


改名









默认无参数

03 最简Shader模板

```
// Shader created with Shader Forge v1.40
// Shader Forge (c) Freya Holmer - http://www.acegikmo.com/shaderforge/
// Note: Manually altering this data may prevent you from opening it in Shader Forge
/*SF_DATA;ver:1.40;sub:START;pass:START;ps:flbk:,iptp:0,cusa:False,bamd:0,cgin:,cpap:True,lico:0,lgpr:1,limd:0,spmd:1,trmd:0,grmd:0,uamb:True,mssp:True,bkdf:False,hqlp:False,rprd:False,enco:False,rmgx:True,imps:True,rpth:0,vtps:0,hqsc:True,nrmq:0,nrsp:0,vomd:0,spxs:False,tesm:0,olmd:1,culm:0,bsrc:0,bdst:1,dpts:2,wrdp:True,dith:0,atcv:False,rfrpo:True,rfrpn:Refraction,coma:15,ufog:False,aust:True,igpj:False,qofs:0,qpre:1,rntp:1,fgom:False,fgoc:False,fgod:False,fgor:False,fgmd:0,fgcr:0.5,fgcg:0.5,fgcb:0.5,fgca:1,fgde:0.01,fgrn:0,fgrf:300,stcl:False,atwp:False,stva:128,stmr:255,stmw:255,stcp:6,stps:0,stfa:0,stfz:0,ofsf:0,ofsu:0,f2p0:False,fnsp:False,fnfb:False,fsmp:False;n:type:ShaderForge.SFN_Final,id:3138,x:32719,y:32712,varname:node_3138,prsc:2|emission-5693-OUT;n:type:ShaderForge.SFN_Vector3,id:5693,x:32526,y:32812,varname:node_5693,prsc:2,v1:1,v2:0.4605795,v3:0;pass:END;sub:END;*/
```

```
Shader "AP1/L03/FlatCol_SF" {
    Properties {
    }
    SubShader {
        Tags {
            "RenderType"="Opaque"
        }
        Pass {
            Name "FORWARD"
            Tags {
                "LightMode"="ForwardBase"
            }


            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag
            #include "UnityCG.cginc"
            #pragma multi_compile_fwdbase_fullshadows
            #pragma target 3.0
            struct VertexInput {
                float4 vertex : POSITION;
            };
            struct VertexOutput {
                float4 pos : SV_POSITION;
            };
            VertexOutput vert (VertexInput v) {
                VertexOutput o = (VertexOutput)0;
                o.pos = UnityObjectToClipPos( v.vertex );
                return o;
            }
            float4 frag(VertexOutput i) : COLOR {
////// Lighting:
////// Emissive:
                float3 emissive = float3(1,0.4605795,0);
                float3 finalColor = emissive;
                return fixed4(finalColor,1);
            }
            ENDCG
        }
    }
    FallBack "Diffuse"
    CustomEditor "ShaderForgeMaterialInspector"
}
```

无用段：SF相关，删掉；

形式段：暂不去理解，Copy大法；

操作段：我们要去编辑修改的部分；

```
Shader "AP1/L03/FlatCol_SF" {
```

操作段1：Shader路径名

```
    Properties {
    }
```

操作段2：材质面板参数

```
            struct VertexInput {
                float4 vertex : POSITION;
            };
```

操作段3：输入结构

```
            struct VertexOutput {
                float4 pos : SV_POSITION;
            };
```

操作段4：输出结构

```
            VertexOutput vert (VertexInput v) {
                VertexOutput o = (VertexOutput)0;
                o.pos = UnityObjectToClipPos( v.vertex );
                return o;
            }
```

操作段5：顶点Shader

```
            float4 frag(VertexOutput i) : COLOR {
////// Lighting:
////// Emissive:
                float3 emissive = float3(1,0.4605795,0);
                float3 finalColor = emissive;
                return fixed4(finalColor,1);
            }
```

操作段6：像素Shader

04 HelloWorld

如何编写：

1. 将最简Shader模板Copy过来；
2. 删去SF相关无用段代码段；
3. 保持形式段代码, 不做任何修改；
4. 操作段1：自定义Shader路径；
5. 操作段2：本例无材质参数, 无需修改；
6. 操作段3：本例仅需输入顶点信息, 无需追加；
7. 操作段4：本例仅需输出顶点信息, 无需追加；
8. 操作段5：本例仅需变换顶点信息，无需修改；
9. 操作段6：输出一个颜色值；

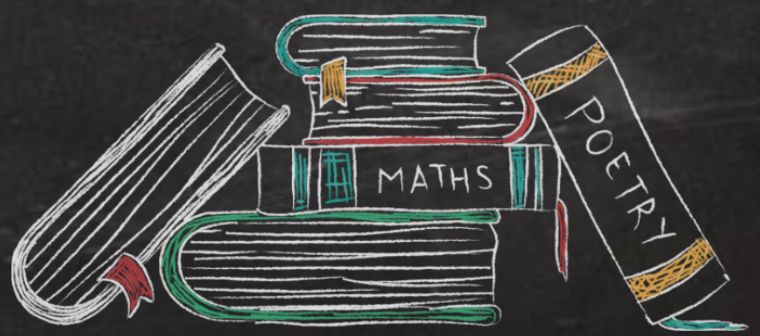
```
Shader "AP1/L03/FlatCol" {
    Properties {
    }
    SubShader {
        Tags {
            "RenderType"="Opaque"
        }
        Pass {
            Name "FORWARD"
            Tags {
                "LightMode"="ForwardBase"
            }


            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag
            #include "UnityCG.cginc"
            #pragma multi_compile_fwdbase_fullshadows
            #pragma target 3.0
            // 输入结构
            struct VertexInput {
                float4 vertex : POSITION;    // 将模型的顶点信息输入进来
            };
            // 输出结构
            struct VertexOutput {
                float4 pos : SV_POSITION;    // 由模型顶点信息换算而来的顶点屏幕位置
            };
            // 输入结构>>>顶点Shader>>>输出结构
            VertexOutput vert (VertexInput v) {
                VertexOutput o = (VertexOutput)0;              // 新建一个输出结构
                o.pos = UnityObjectToClipPos( v.vertex );      // 变换顶点信息 并将其塞给输出结构
                return o;                                      // 将输出结构 输出
            }
            // 输出结构>>>像素
            float4 frag(VertexOutput i) : COLOR {
////// Lighting:
////// Emissive:
                //float3 emissive = float3(1,0.4605795,0);
                //float3 finalColor = emissive;
                //return fixed4(finalColor,1);
                return float4(0.0, 1.0, 0.0, 1.0);
            }

            ENDCG
        }
    }
    FallBack "Diffuse"
}
```

$\frac{24 \times 5}{33}$

3 符文·筑基·照



01 代码Lambert

如何编写：

1. 将最简Shader模板Copy过来；
2. 删去SF相关无用段代码段；
3. 保持形式段代码, 不做任何修改；
4. 操作段1：自定义Shader路径；
5. 操作段2：本例无材质参数, 无需修改；
6. 操作段3：本例需追加法线信息,；
7. 操作段4：本例需追加法线信息；
8. 操作段5：本例需追加法线信息变换；
9. 操作段6：获取nDir, lDir, 点乘, 截断负值后输出；

```
Shader "AP1/L03/Lambert" {
    Properties {
    }
    SubShader {
        Tags {
            "RenderType"="Opaque"
        }
        Pass {
            Name "FORWARD"
            Tags {
                "LightMode"="ForwardBase"
            }


            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag
            #include "UnityCG.cginc"
            #pragma multi_compile_fwdbase_fullshadows
            #pragma target 3.0
            // 输入结构
            struct VertexInput {
                float4 vertex : POSITION;    // 将模型顶点信息输入进来
                float4 normal : NORMAL;      // 将模型法线信息输入进来
            };
            // 输出结构
            struct VertexOutput {
                float4 pos : SV_POSITION;    // 由模型顶点信息换算而来的顶点屏幕位置
                float3 nDirWS : TEXCOORD0;   // 由模型法线信息换算来的世界空间法线信息
            };
            // 输入结构>>>顶点Shader>>>输出结构
            VertexOutput vert (VertexInput v) {
                VertexOutput o = (VertexOutput)0;                   // 新建一个输出结构
                o.pos = UnityObjectToClipPos( v.vertex );           // 变换顶点信息 并将其塞给输出结构
                o.nDirWS = UnityObjectToWorldNormal(v.normal);      // 变换法线信息 并将其塞给输出结构
                return o;                                           // 将输出结构 输出
            }
            // 输出结构>>>像素
            float4 frag(VertexOutput i) : COLOR {
                float3 nDir = i.nDirWS;                             // 获取nDir
                float3 lDir = _WorldSpaceLightPos0.xyz;             // 获取lDir
                float nDotl = dot(i.nDirWS, lDir);                  // nDir点积lDir
                float lambert = max(0.0, nDotl);                    // 截断负值
                return float4(lambert, lambert, lambert, 1.0);                  // 输出最终颜色
            }
            ENDCG
        }
    }
    FallBack "Diffuse"
}
```

02 与SF同效Shader比对

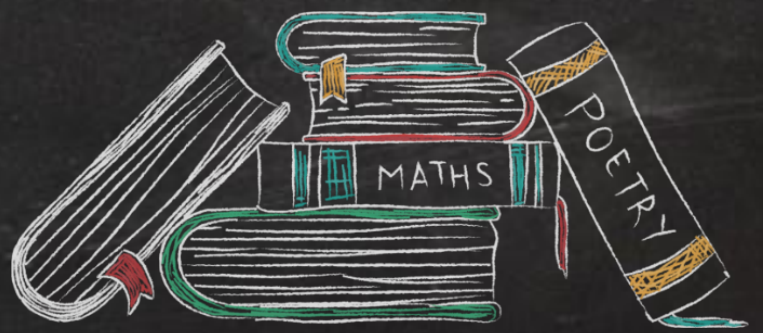
区别：

1. 输出结构多了posWorld，但实际没用到；
2. 为了适配Unity的Lightmaping和实时全局光，输出结构多了额外的LightingCoords，但实际没用到；
3. 逐像素的归一化光向量，实际效果没差别；
4. 其他因为程序化生产代码带来的多余逻辑；

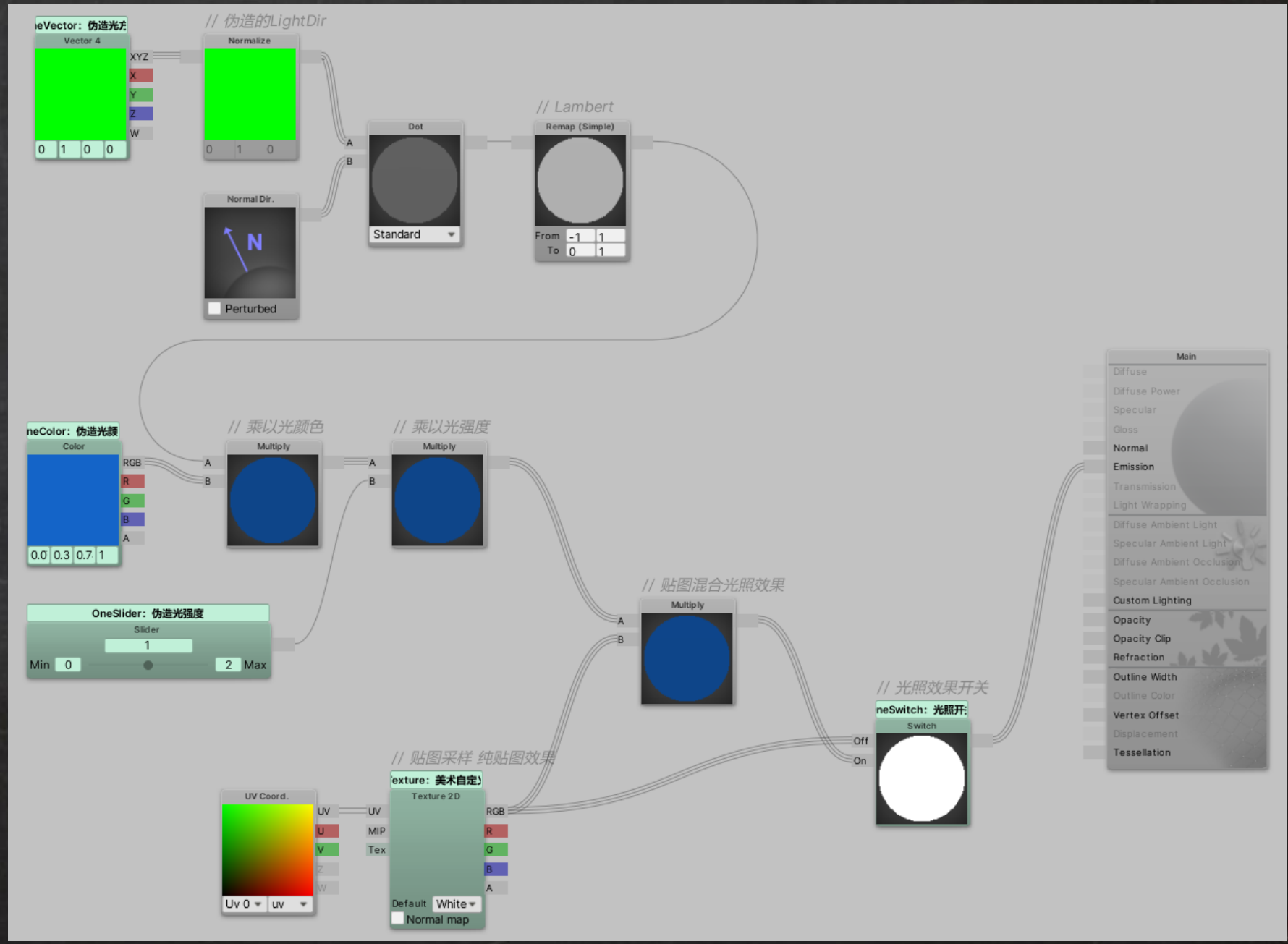
```
struct VertexInput {
    float4 vertex : POSITION;
    float3 normal : NORMAL;
};
struct VertexOutput {
    float4 pos : SV_POSITION;
    float4 posWorld : TEXCOORD0;
    float3 normalDir : TEXCOORD1;
    LIGHTING_COORDS(2,3)
};
VertexOutput vert (VertexInput v) {
    VertexOutput o = (VertexOutput)0;
    o.normalDir = UnityObjectToWorldNormal(v.normal);
    o.posWorld = mul(unity_ObjectToWorld, v.vertex);
    o.pos = UnityObjectToClipPos( v.vertex );
    TRANSFER_VERTEX_TO_FRAGMENT(o)
    return o;
}
float4 frag(VertexOutput i) : COLOR {
    float3 normalDirection = i.normalDir;
    float3 lightDirection = normalize(_WorldSpaceLightPos0.xyz);
////// Lighting:
////// Emissive:
    float node_4587 = saturate(dot(i.normalDir,lightDirection));
    float3 emissive = float3(node_4587,node_4587,node_4587);
    float3 finalColor = emissive;
    return fixed4(finalColor,1);
}
```

$\frac{24 \times 5}{33}$

4 结点组·参数

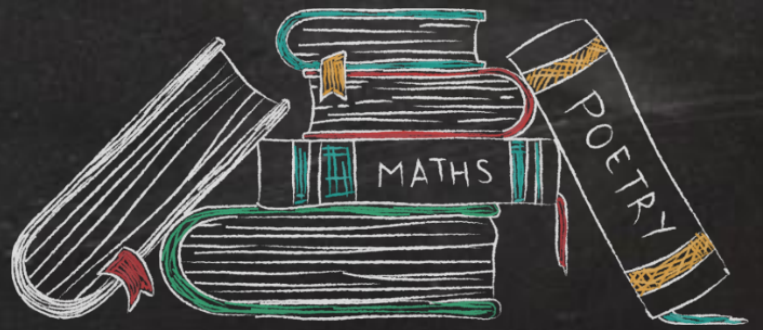


01 参数结点常用全家桶



$\frac{24 \times 5}{33}$

5 作业



01 作业

符文作业：

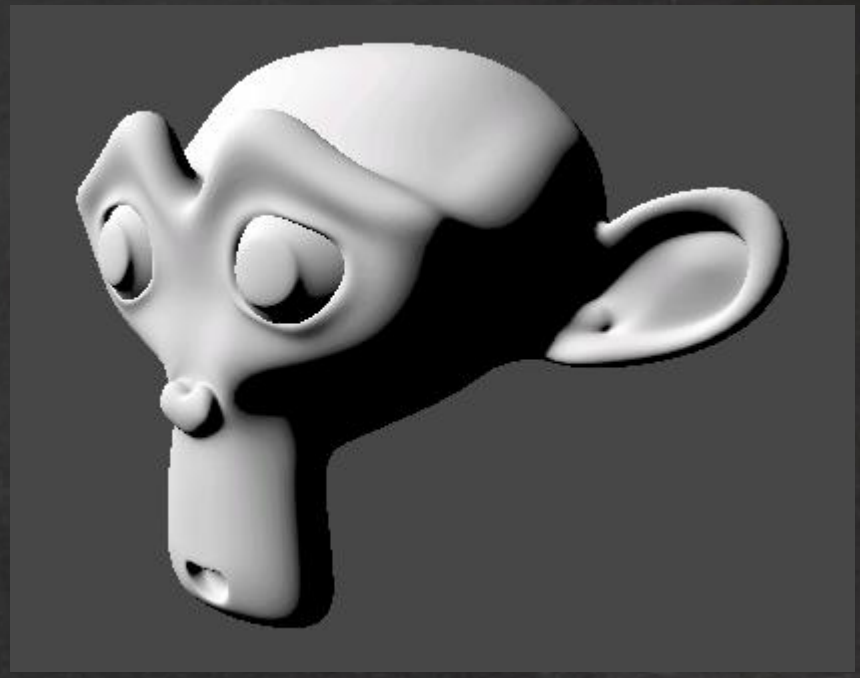
- 必做：HelloWorld，Lambert；
- 选作：半兰伯特HalfLambert；

连连看作业：

- 如Gif图创建一个材质，面板上提供一个Slider可以平滑调整猴子的明暗调子；

创意题：

- 用所学的知识，任意发挥；




Thanks