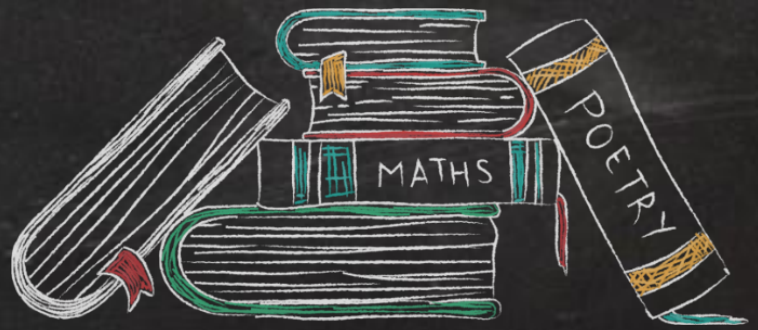

大家好
欢迎加入LightDir（光向）研习社
欢迎大家一同探索开源共享的知识分享模式
LightDir.光向研习社

今日内容

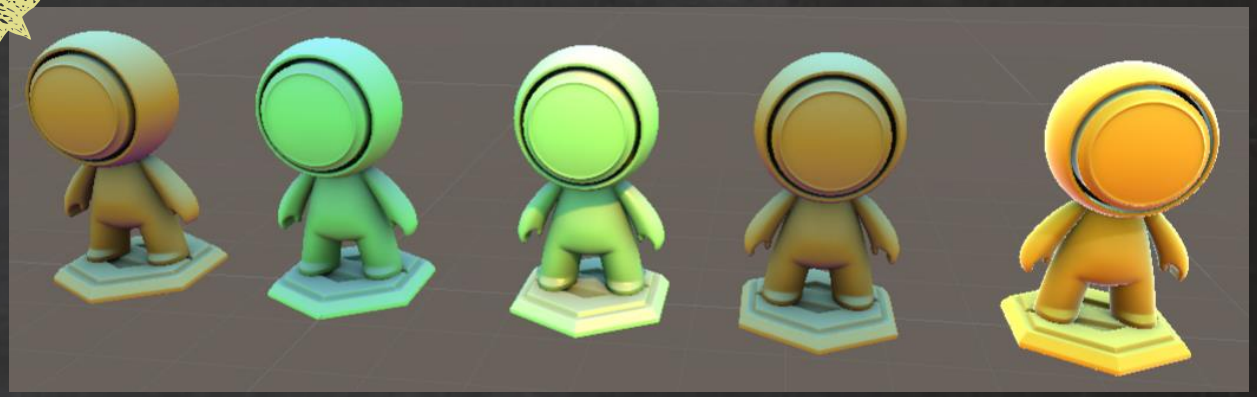
- 作业·点评 答疑
- 作业·答案 批改
- 结印·筑基·法线
- 符文·筑基·法线

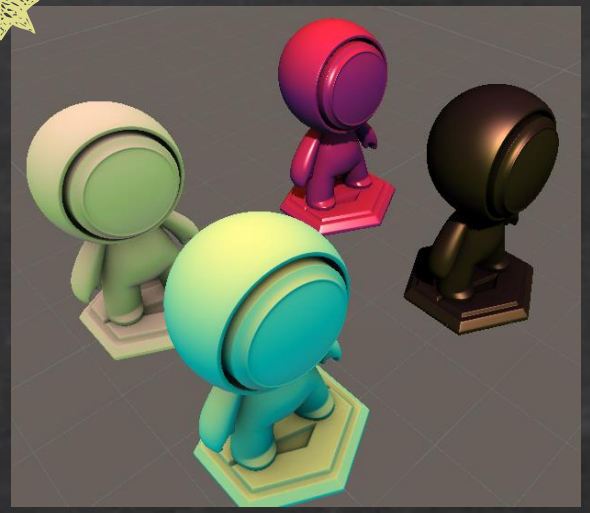
$\frac{24 \times 5}{33}$

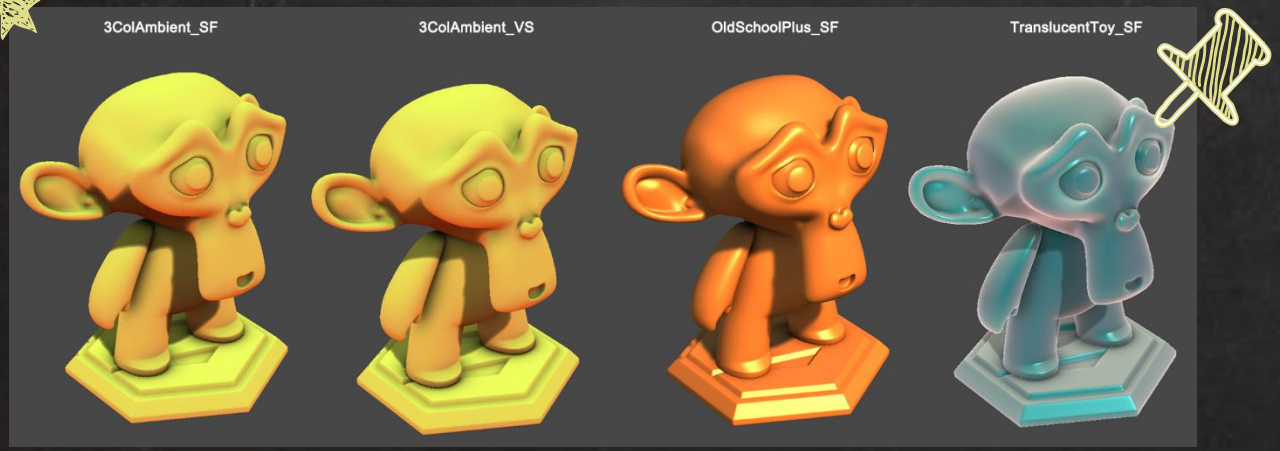
1 作业·点评 答疑

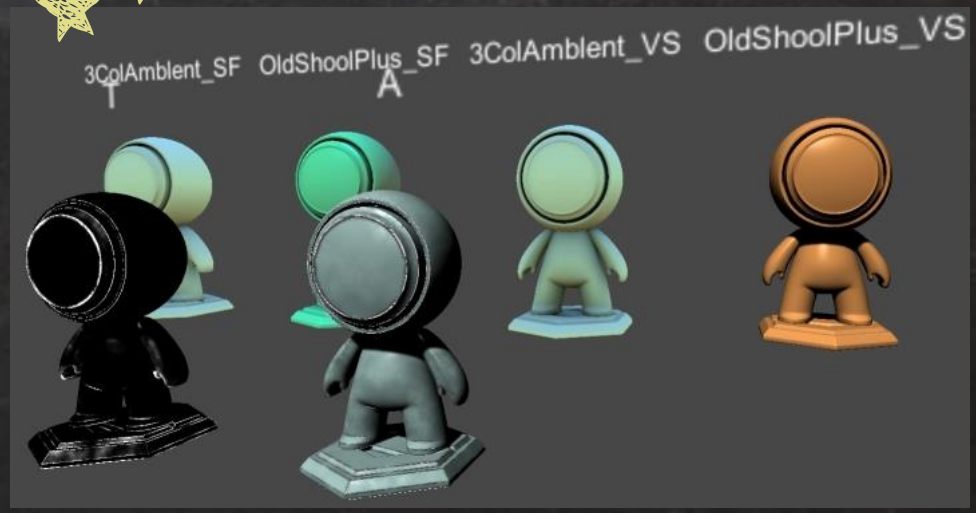


01 一组

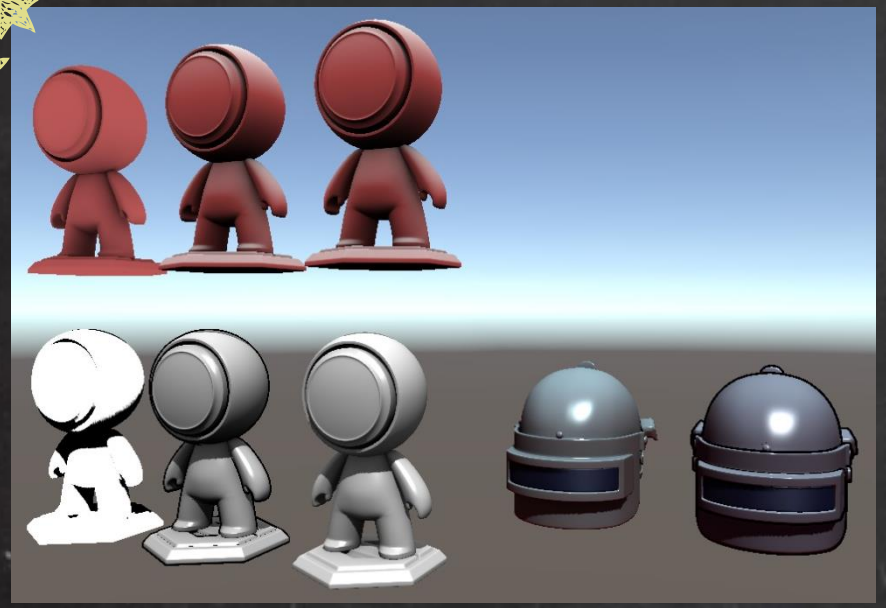


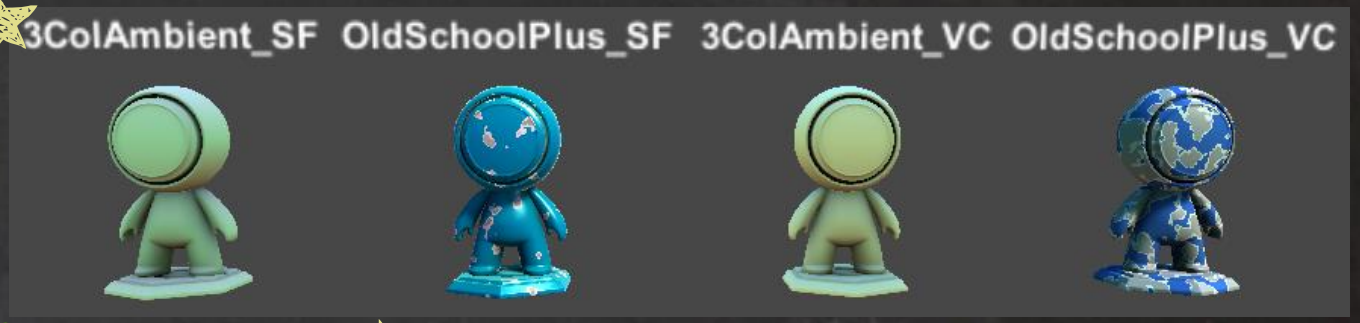






02 二组





英文体系 无；

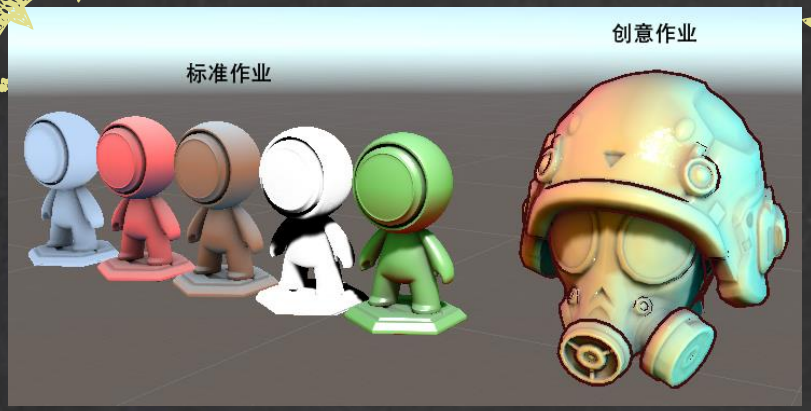
```
Properties
{
    _Occlustion ("AO", 2d) = "white" {}
    _EnvUpCol ("上方环境色", Color) = (1.1.1.1);
    _EnvSideCol ("中间环境色", Color) = (1.1.1.1);
    _EnvDownCol ("下方环境色", Color) = (1.1.1.1);
}
```

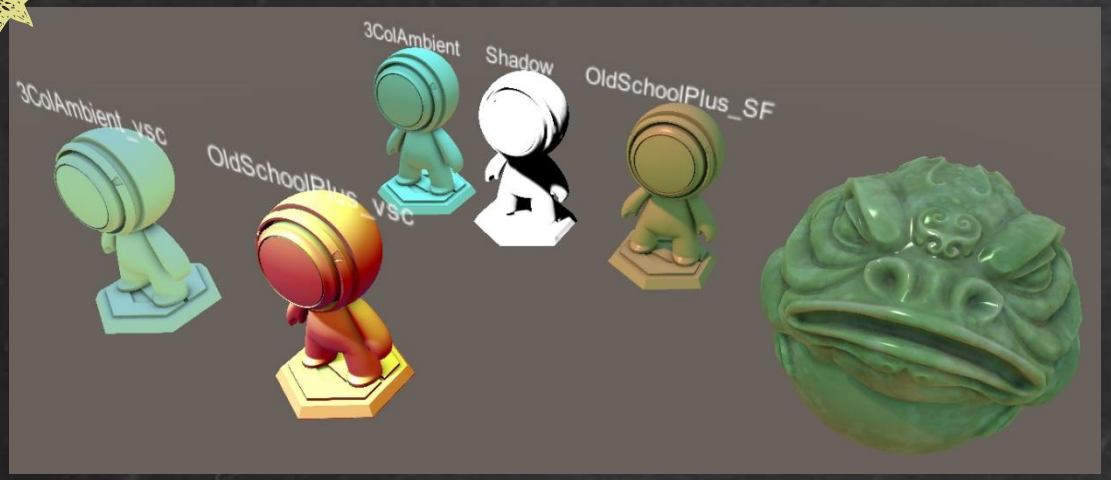
；号

```
uniform sampler2D _Occlustion;
```

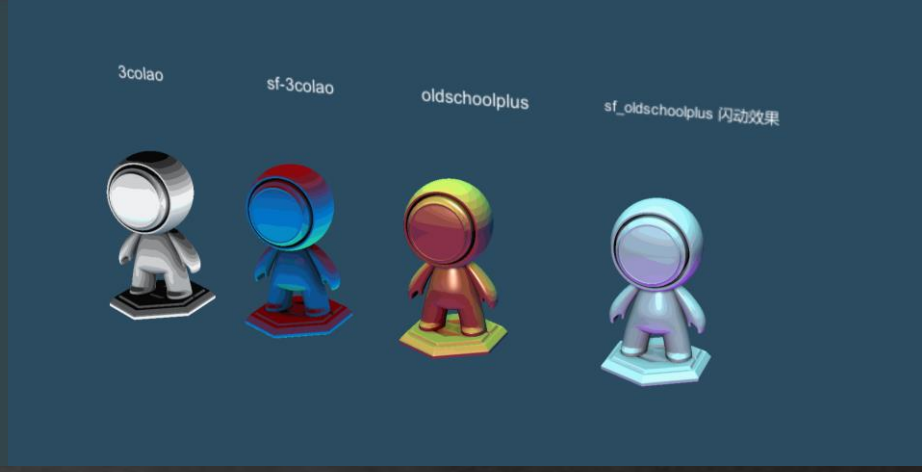
多个t

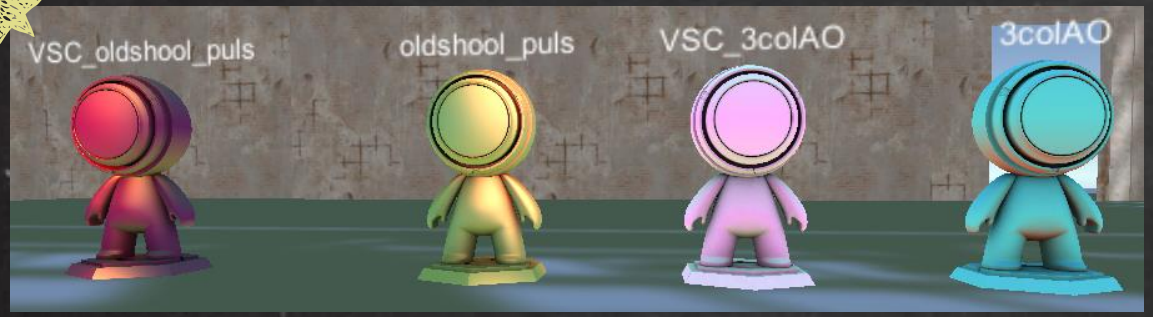
03 三组



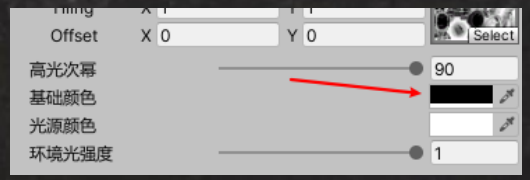


04 四组






```
// 计算环境光照
float3 EnvLighting = EnvCol * occlusion;
float3 Oldschool = lambert * _BaseColor + phong ;
float3 Shadow  = attenuation ;
float3 OldschoolPlus = ((Oldschool * _LightColor) * Shadow) + EnvLighting;
// 返回最终颜色
return float4 (OldschoolPlus , 1.0);
```



BaseColor 全黑，且发乘给 EnvLighting

手写 连连看

05 总结

姓名	代码作业	连连看作业	截图	打包	创意题
周川	✓	✓	✓	✓	曲率计算 超纲 不讲
赵井才	✓	✓	✓	✓	追加菲涅尔 周日讲
叶小芸	✓	✓	✓	✓	AO 插值 和白色插值
赵翔	✓	✓	✓	✓	奇异效果 可讲
张	✓	✓	✓	✓	无
周翰林	✓	✓	缺大合照	✓	OSP改掉漆 不讲
杨易	✓	✓	✓	缺场景 Prefab？	卡通金属头盔 可讲
廖宴楠	✓✗	✓	✓	✓	奇异效果 可讲
冯超越	✓	✓	✓	✓	模贴尝试 不讲
邵志刚	✓	✓	✓	✓	魔法球 特效环节讲
申伏琳	✓	✓	✓	缺场景	OSP改Ramp 不讲
陈沛霖	✓	✓	✓	✓	模贴尝试 不讲
顾友海	✓	✓	✓	✓	OldSchoolPlus效果尝试 不讲
冷翰林	✓	✓	✓	✓	横向环境变化 不讲
王岩	✓	✓	✓	✓	OSP改掉漆 不讲
宋歌	✓	✓	✓	✓	效果可商用 可讲
冯晓晨	✓	✓	✓	✓	闪动超纲 效果尝试 不讲
罗陈	✓	✓	✓	✓	买房了？
昊	✓	✓	✓	✓	模贴尝试 不讲
沈术磐					

作业情况：

- 代码和连连看都没啥问题；
- 创意尝试也比前几周多了很多；

作业规范：

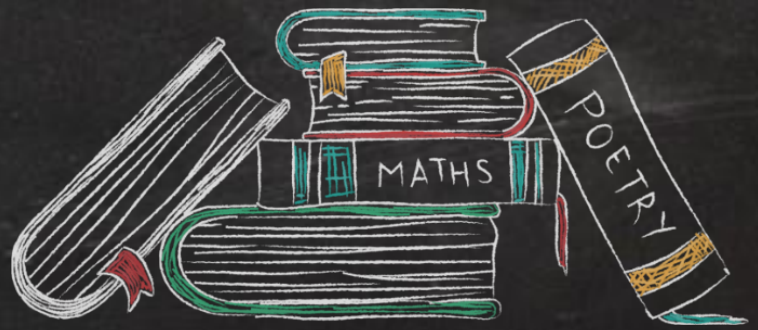
- 大合照，合照场景；

06 答疑

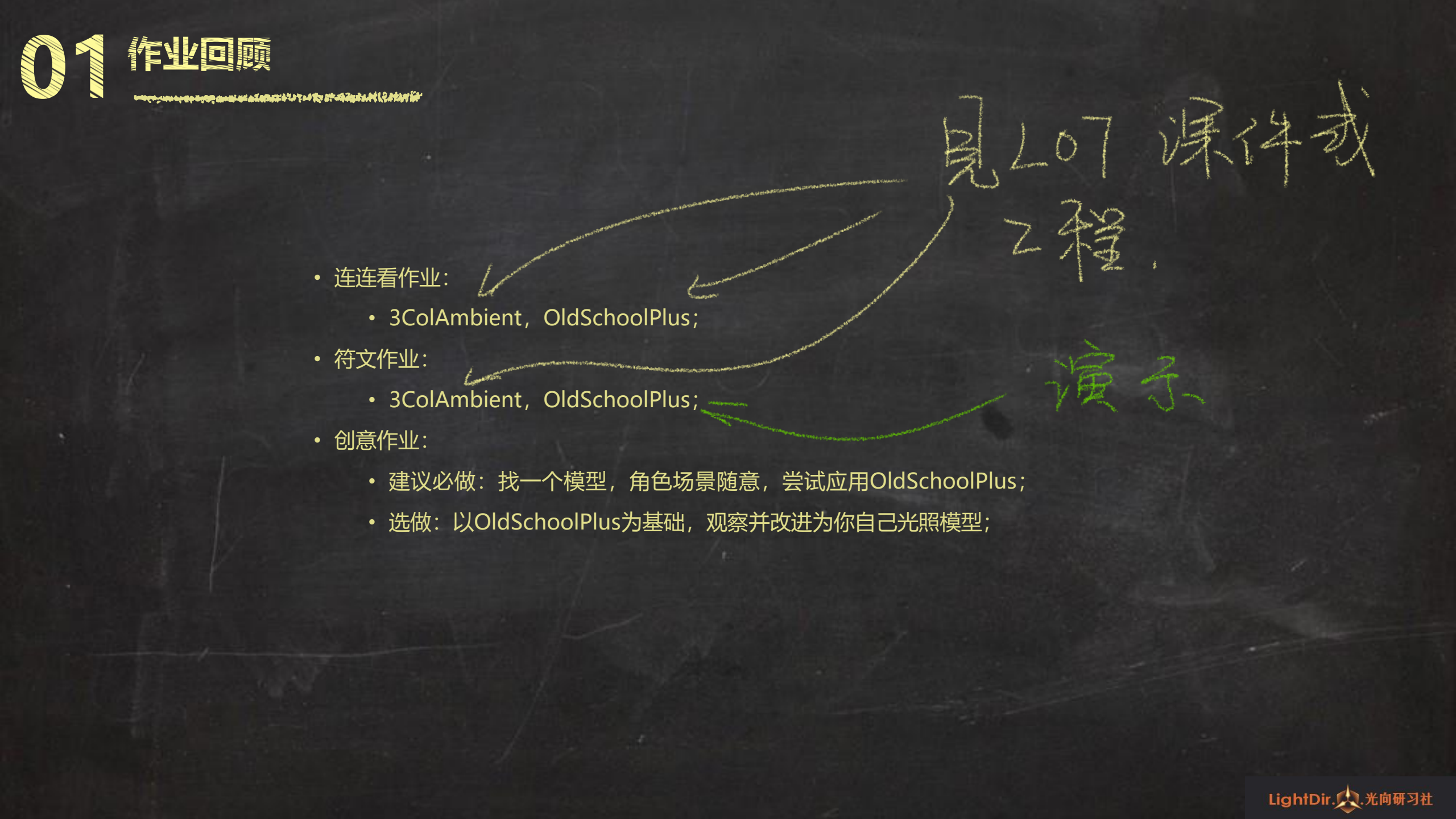
- if判断断开部分Shader语句，有无优化效果?
  - 没有优化效果，渲染时会执行每一个if分支，一般不写if;
  - 一般用ShaderFeature，MultiCompile；超纲不讲;
- 只能连连看做完再翻成代码，对代码恐惧;
  - 如果课程能让你立即达到直接手撸代码，为啥还设计个连连看环节给你过渡哦；慢慢过渡，需要时间；至少你现在借助连连看是可以写代码的;
  - 对付恐惧最好的。。。
- 会不会讲PBR?
  - 会，不是现在，不带黑盒的讲，目前大部分同学还无法接受;
  - 带黑盒的不会讲，搜索下SurfaceShader怎么写就完事了;

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

2 作业·答案 批改

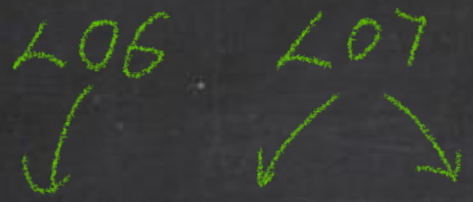


01 作业回顾

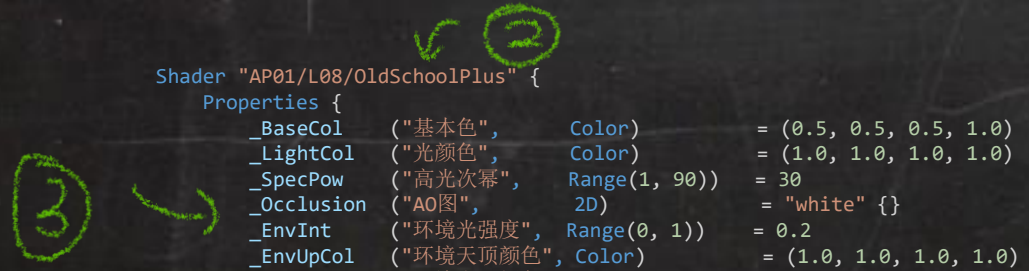


- 连连看作业：
  - 3ColAmbient，OldSchoolPlus；
- 符文作业：
  - 3ColAmbient，OldSchoolPlus；
- 创意作业：
  - 建议必做：找一个模型，角色场景随意，尝试应用OldSchoolPlus；
  - 选做：以OldSchoolPlus为基础，观察并改进为你自己光照模型；

02 OldSchoolPlus 代码



1. 分析构成：
   - OldSchoolPlus=OldSchoolP+3ColAmbient+Shadow；
   - 以OldSchoolP为模板，追加3ColAmbient，Shadow功能；
   - OldSchoolPlus_SF作为参考；
2. 复制OldSchoolP代码，改Shader路径名；
3. 从OldSchoolPlus_SF复制面板参数，并修改；
4. 注意追加Shadow相关的Include文件；
5. 对应面板参数声明输入参数，一一对应，注意类型；
6. 输入结构，顶点Shader，输出结构：
   1. 追加UV相关代码；
   2. 追加投影相关代码；输出结构注意编号；顶点Shader注意pos命名；




```
Shader "AP01/L08/OldSchoolPlus" {
    Properties {
        _BaseCol    ("基本色",      Color)          = (0.5, 0.5, 0.5, 1.0)
        _LightCol   ("光颜色",      Color)          = (1.0, 1.0, 1.0, 1.0)
        _SpecPow    ("高光次幂",    Range(1, 90))   = 30
        _Occlusion  ("AO图",        2D)             = "white" {}
        _EnvInt     ("环境光强度",  Range(0, 1))    = 0.2
        _EnvUpCol   ("环境天顶颜色", Color)         = (1.0, 1.0, 1.0, 1.0)
        _EnvSideCol ("环境水平颜色", Color)         = (0.5, 0.5, 0.5, 1.0)
        _EnvDownCol ("环境地表颜色", Color)         = (0.0, 0.0, 0.0, 0.0)
    }
```


```
    SubShader {
        Tags {
            "RenderType"="Opaque"
        }
        Pass {
            Name "FORWARD"
            Tags {
                "LightMode"="ForwardBase"
            }
            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag
            #include "UnityCG.cginc"
            // 追加投影相关包含文件
            #include "AutoLight.cginc"
            #include "Lighting.cginc"
            #pragma multi_compile_fwdbase_fullshadows
            #pragma target 3.0
            // 输入参数
            uniform float3 _BaseCol;
            uniform float3 _LightCol;
            uniform float _SpecPow;
            uniform sampler2D _Occlusion;
            uniform float _EnvInt;
            uniform float3 _EnvUpCol;
            uniform float3 _EnvSideCol;
            uniform float3 _EnvDownCol;
            // 输入结构
            struct VertexInput {
                float4 vertex   : POSITION;   // 顶点信息 Get✔
                float4 normal   : NORMAL;     // 法线信息 Get✔
                float2 uv0      : TEXCOORD0;  // UV信息 Get✔
            };
            // 输出结构
            struct VertexOutput {
                float4 pos      : SV_POSITION;  // 裁剪空间（暂理解为屏幕空间吧）顶点位置
                float2 uv0      : TEXCOORD0;    // UV0
                float4 posWS    : TEXCOORD1;    // 世界空间顶点位置
                float3 nDirWS   : TEXCOORD2;    // 世界空间法线方向
                LIGHTING_COORDS(3,4)            // 投影相关
            };
            // 输入结构>>>顶点Shader>>>输出结构
            VertexOutput vert (VertexInput v) {
                VertexOutput o = (VertexOutput)0;                           // 新建输出结构
                o.pos = UnityObjectToClipPos( v.vertex );                   // 变换顶点位置 OS>CS
                o.uv0 = v.uv0;                                              // 传递UV
                o.posWS = mul(unity_ObjectToWorld, v.vertex);               // 变换顶点位置 OS>WS
                o.nDirWS = UnityObjectToWorldNormal(v.normal);              // 变换法线方向 OS>WS
                TRANSFER_VERTEX_TO_FRAGMENT(o)                              // 投影相关
                return o;                                                   // 返回输出结构
            }
```

03 OldSchoolPlus 代码

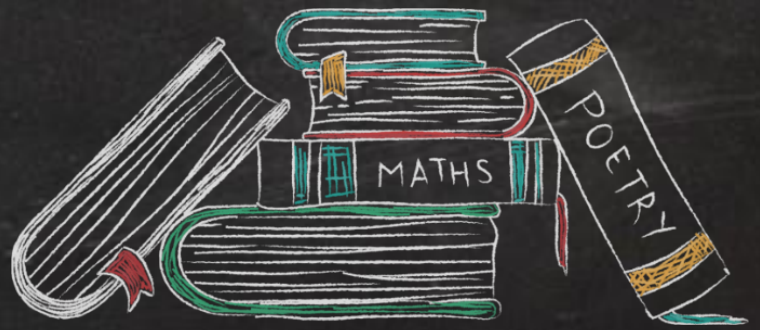
7. 像素Shader
    1. 向量，点积结果准备部分与OldSchoolP一致，不用改；
    2. 光照模型拆为 直接光照 环境光照 两部分：
        1. 直接光照：在Lambert+Phong的基础上追加：
            - 投影：算法可Copy自L07_Shadow；
            - 光颜色：乘以_LightCol；
        2. 环境光照：算法可Copy自L07_3ColAmbient；
    3. 将直接光照和环境光照相加输出；

```
// 输出结构>>>像素
float4 frag(VertexOutput i) : COLOR {
    // 准备向量
    float3 nDir = normalize(i.nDirWS);
    float3 lDir = _WorldSpaceLightPos0.xyz;
    float3 vDir = normalize(_WorldSpaceCameraPos.xyz - i.posWS.xyz);
    float3 rDir = reflect(-lDir, nDir);

    // 准备点积结果
    float ndotl = dot(nDir, lDir);
    float vdotr = dot(vDir, rDir);

    // 光照模型(直接光照部分)
    float shadow = LIGHT_ATTENUATION(i);        // 获取投影
    float lambert = max(0.0, ndotl);
    float phong = pow(max(0.0, vdotr), _SpecPow);
    float3 dirLighting = (_BaseCol * lambert + phong) * _LightCol * shadow;

    // 光照模型(环境光照部分)
    float upMask = max(0.0, nDir.g);                // 获取朝上部分遮罩
    float downMask = max(0.0, -nDir.g);             // 获取朝下部分遮罩
    float sideMask = 1.0 - upMask - downMask;       // 获取侧面部分遮罩
    // 混合环境色
    float3 envCol = _EnvUpCol * upMask + _EnvSideCol * sideMask + _EnvDownCol * downMask;
    float occlusion = tex2D(_Occlusion, i.uv0);         // 采样Occlusion贴图
    float3 envLighting = envCol * _EnvInt * occlusion;  // 计算环境光照

    // 返回结果
    float3 finalRGB = dirLighting + envLighting;
    return float4(finalRGB, 1.0);
}
```

04 批改

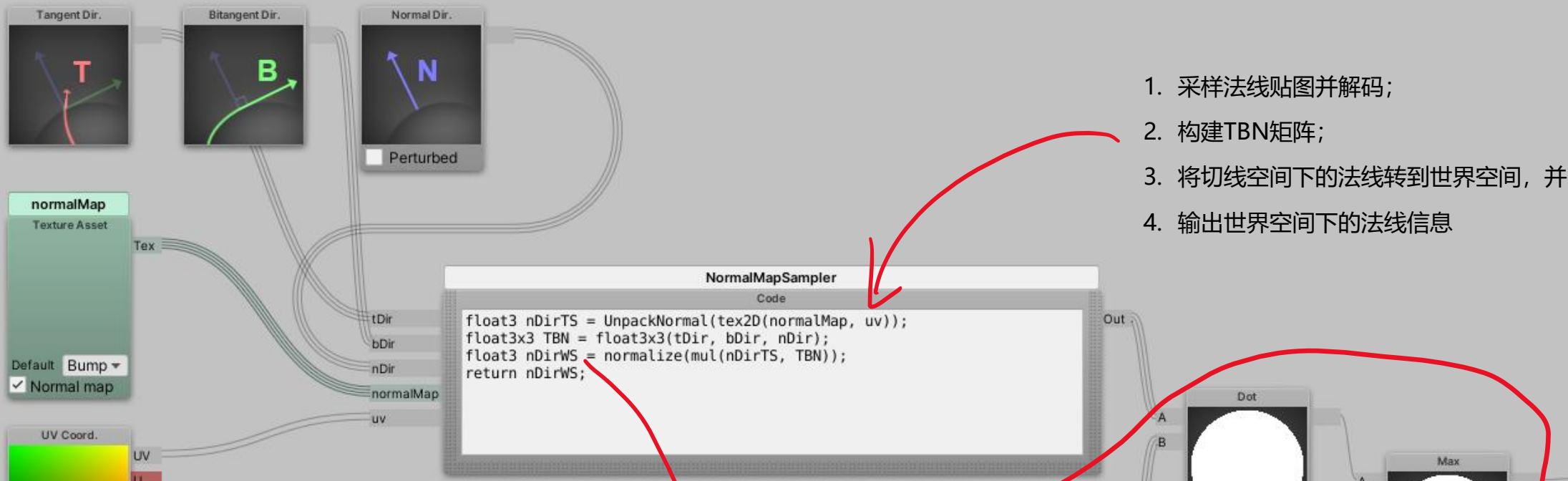
$\frac{24 \times 5}{33}$

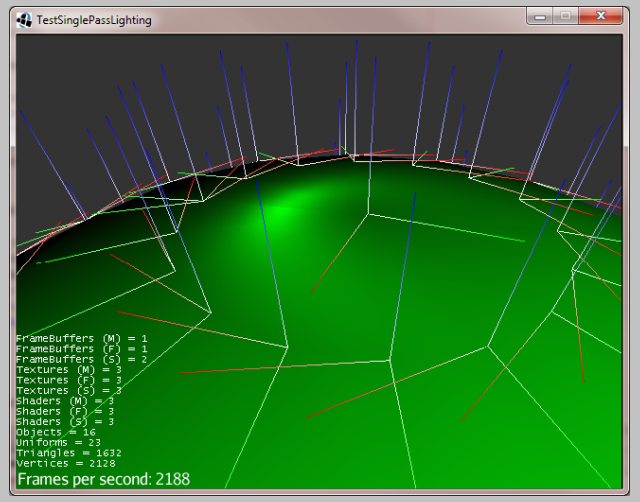
3 结印·筑基·法线



01 连连看比代码头疼的例子

1. 采样法线贴图并解码；
2. 构建TBN矩阵；
3. 将切线空间下的法线转到世界空间，并归一化；
4. 输出世界空间下的法线信息

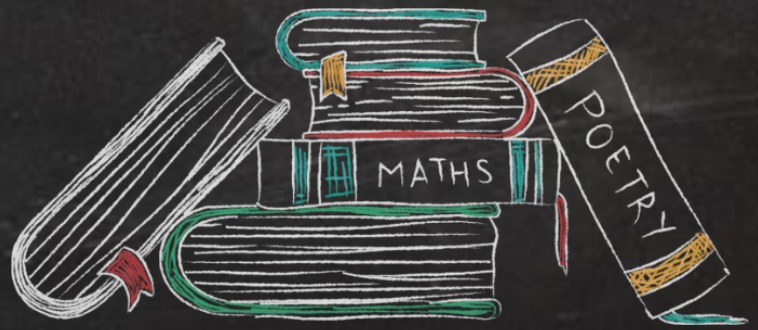




TBN

一般lambert

4 符文·筑基·法线



01 代码比连连看清爽的例子

1. 我们需要一个光照模型来显示法线效果，选取最简单的Lambert；所以以Lambert为模板；
2. 复制L03_Lambert，修改Shader路径名；
3. 面板参数追加一张法线贴图；
4. 输出参数对应追加；
5. 输入结构追加：
    1. UV0：用于采样法线贴图；
    2. Tangent：用于构建TBN矩阵；
6. 输出结构追加：
    1. UV0：用于采样法线贴图；
    2. tDirWS，bDirWS，nDirWS：切线空间3轴向方向，用于构建TBN矩阵；
7. 顶点Shader追加相应内容；
8. 像素Shader中nDir的获取方法与连连看类似；其余保持Lambert不变；

```shader
Shader "AP01/L08/NormalMap" {
    Properties {
        _NormalMap ("法线贴图", 2D) = "bump" {}
    }
    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
            // 输入参数
            uniform sampler2D _NormalMap;
            // 输入结构
            struct VertexInput {
                float4 vertex : POSITION;    // 顶点信息
                float2 uv0 : TEXCOORD0;      // 需要UV坐标 采样法线贴图
                float4 normal : NORMAL;      // 法线信息
                float4 tangent : TANGENT;    // 构建切线空间需要模型切线信息
            };
            // 输出结构
            struct VertexOutput {
                float4 pos : SV_POSITION;
                float2 uv0 : TEXCOORD0;      // UV信息
                float3 nDirWS : TEXCOORD1;   // 世界空间法线信息
                float3 tDirWS : TEXCOORD2;   // 世界空间切线信息
                float3 bDirWS : TEXCOORD3;   // 世界空间切线信息
            };
            // 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.uv0 = v.uv0;                                     // 传递UV信息
                o.nDirWS = UnityObjectToWorldNormal(v.normal);     // 世界空间法线信息
                o.tDirWS = normalize(mul( unity_ObjectToWorld, float4(v.tangent.xyz, 0.0)).xyz);     // 世界空间切线信息
                o.bDirWS = normalize(cross(o.nDirWS, o.tDirWS) * v.tangent.w);    // 世界空间切线信息
                return o;                                          // 将输出结构 输出
            }
            // 输出结构>>>像素
            float4 frag(VertexOutput i) : COLOR {
                // 获取nDir
                float3 var_NormalMap = UnpackNormal(tex2D(_NormalMap, i.uv0)).rgb;  // 采样法线纹理并解码 切线空间nDir
                float3x3 TBN = float3x3(i.tDirWS, i.bDirWS, i.nDirWS);              // 构建TBN矩阵
                float3 nDir = normalize(mul(var_NormalMap, TBN));                   // 世界空间nDir
                // 获取lDir
                float3 lDir = _WorldSpaceLightPos0.xyz;
                // 一般Lambert
                float nDotl = dot(nDir, 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"
}
```