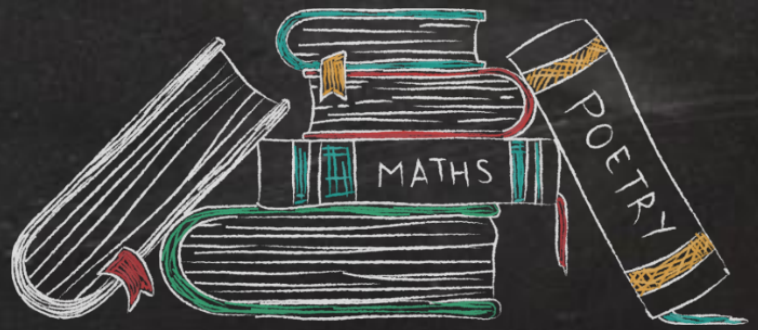
今日内容

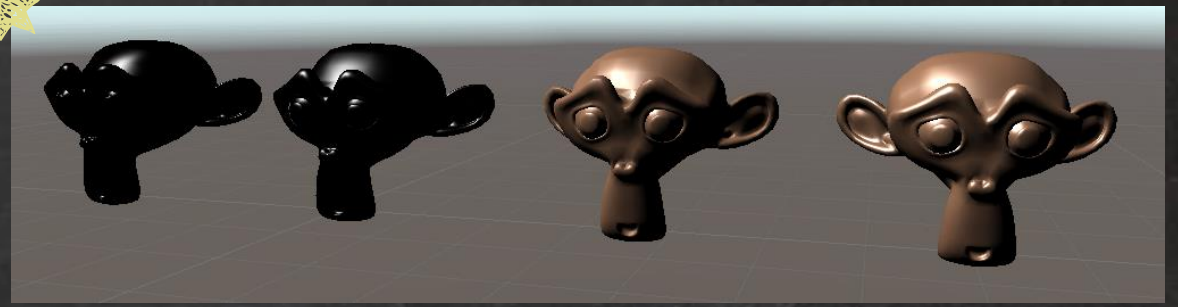
- 作业·点评
- 作业·答案 批改 答疑
- 情报·BRDF

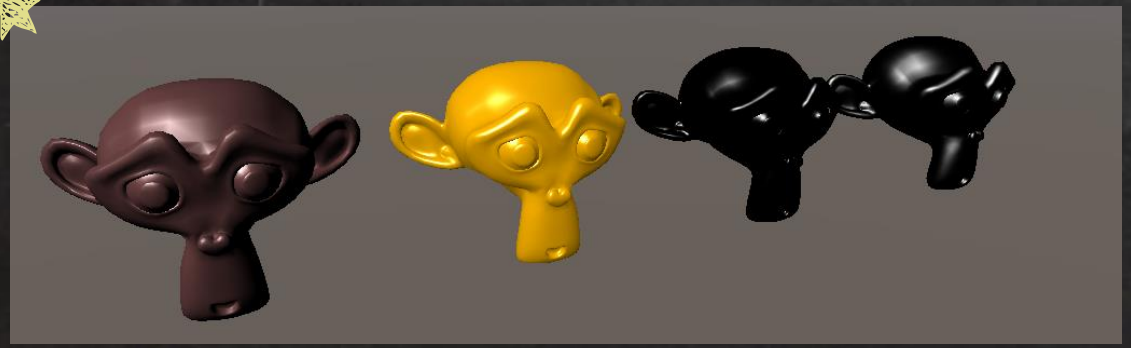
$\frac{24 \times 5}{33}$

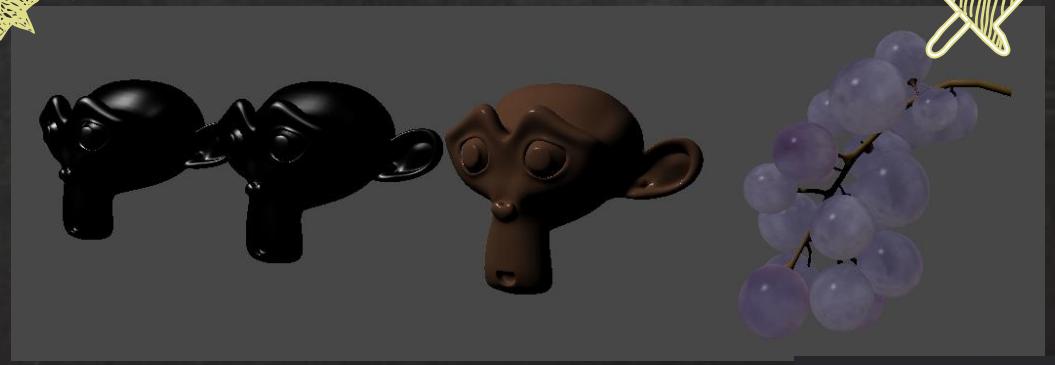
1 作业·点评



01 一组

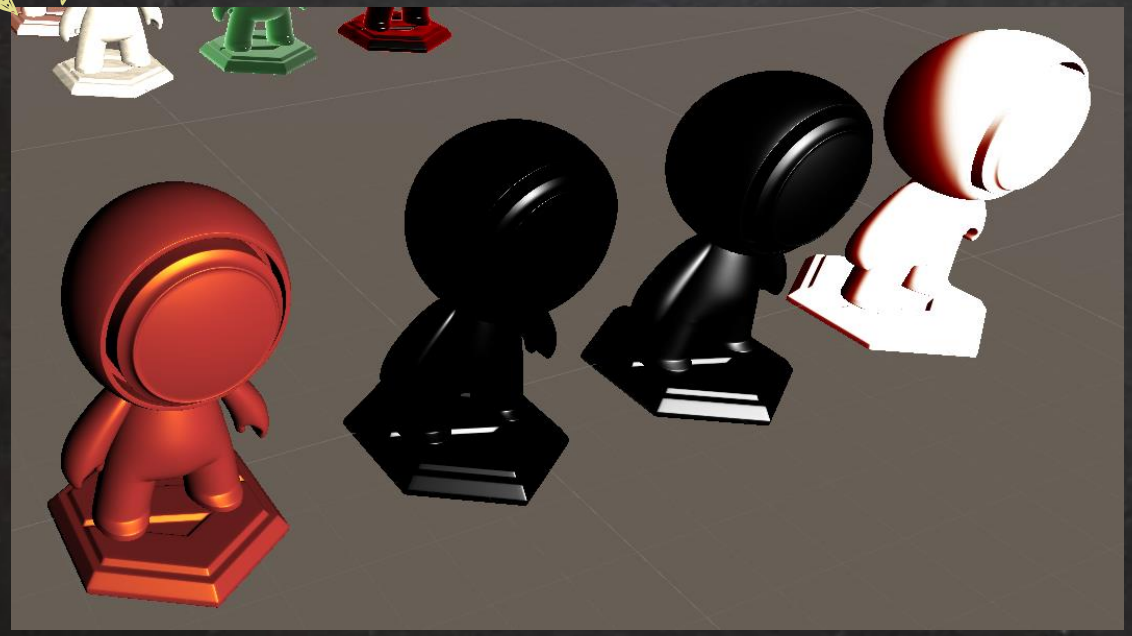


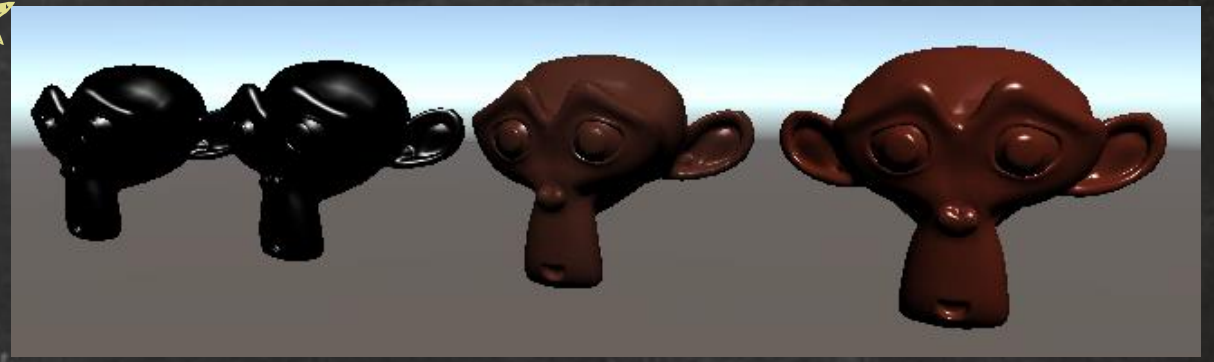


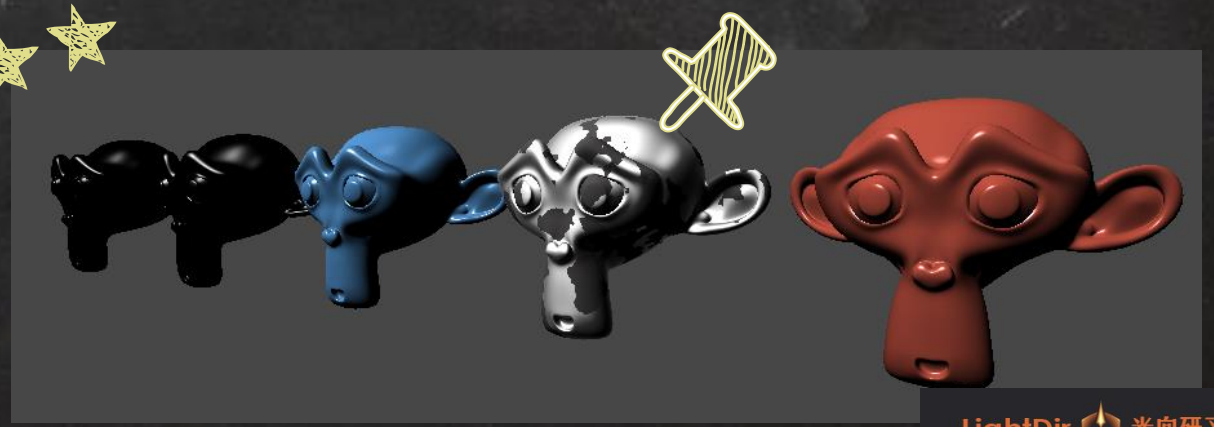


02 二组

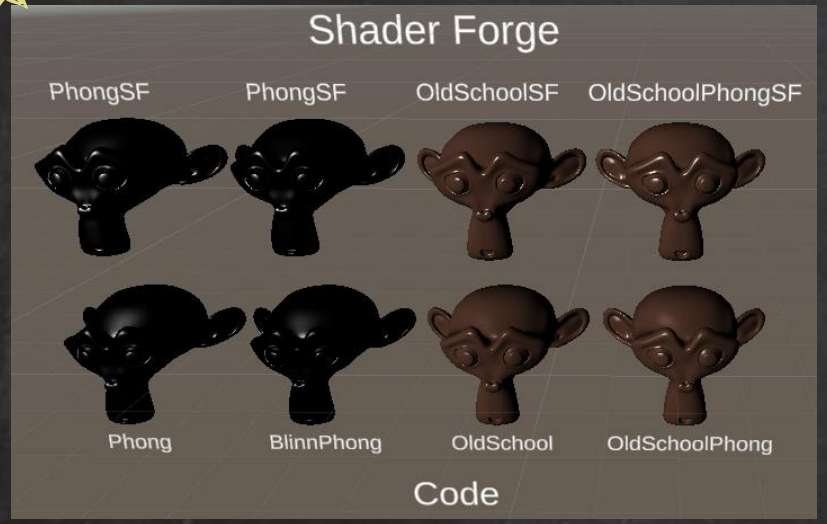
OldSchool_Phone_VS　OldSchool_Phone_SF　Phone_SF　BlinnPhone_SF　OldSchool_BlinnPhone_SF　OldSchool_BlinnPhone_VS

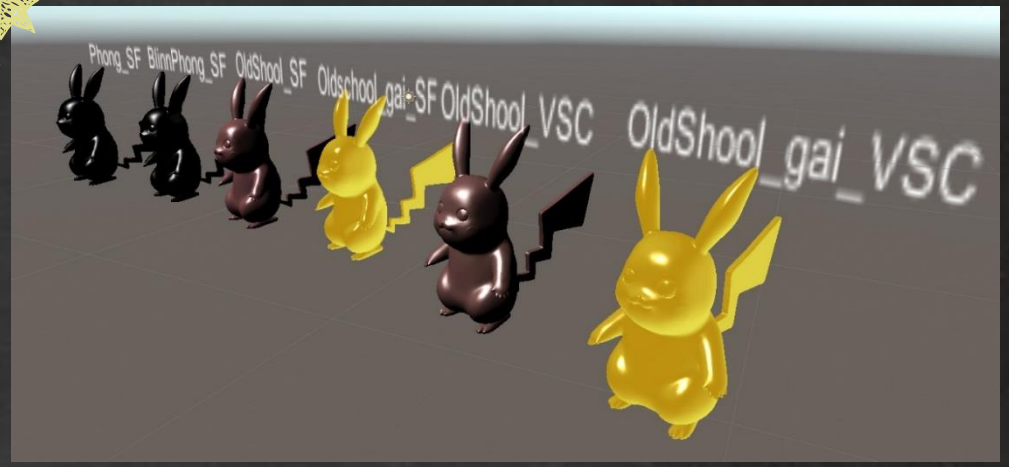


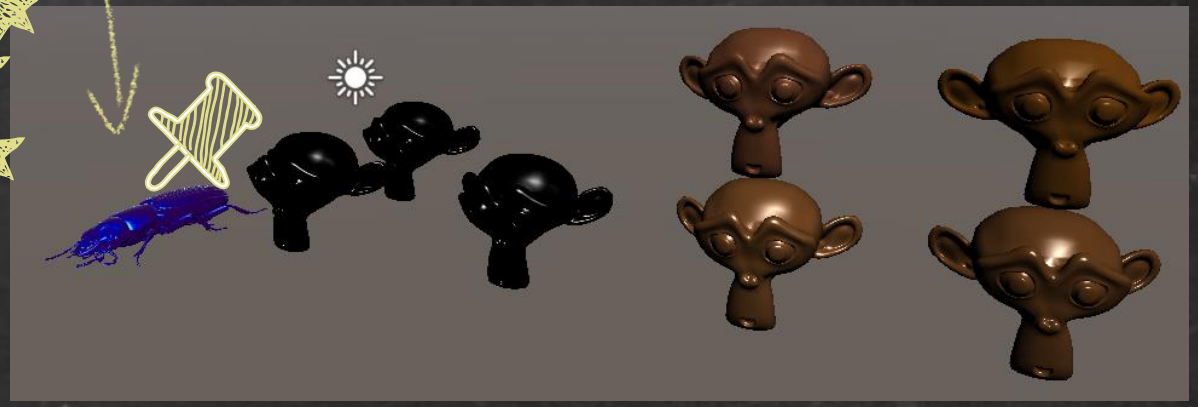


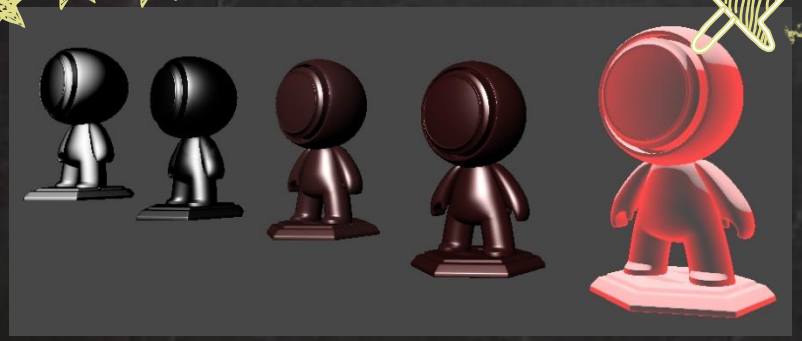


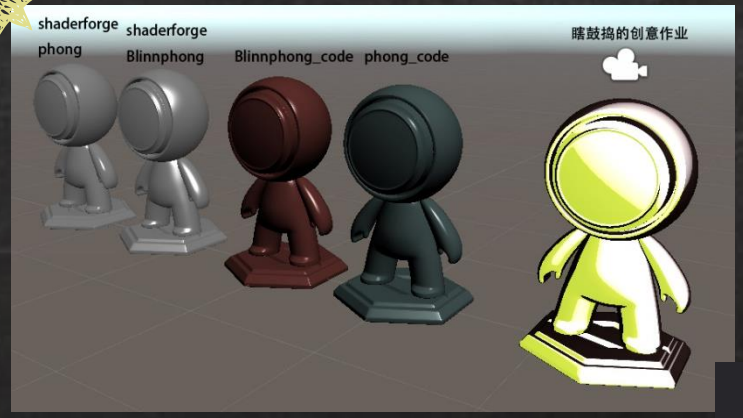
03 三组



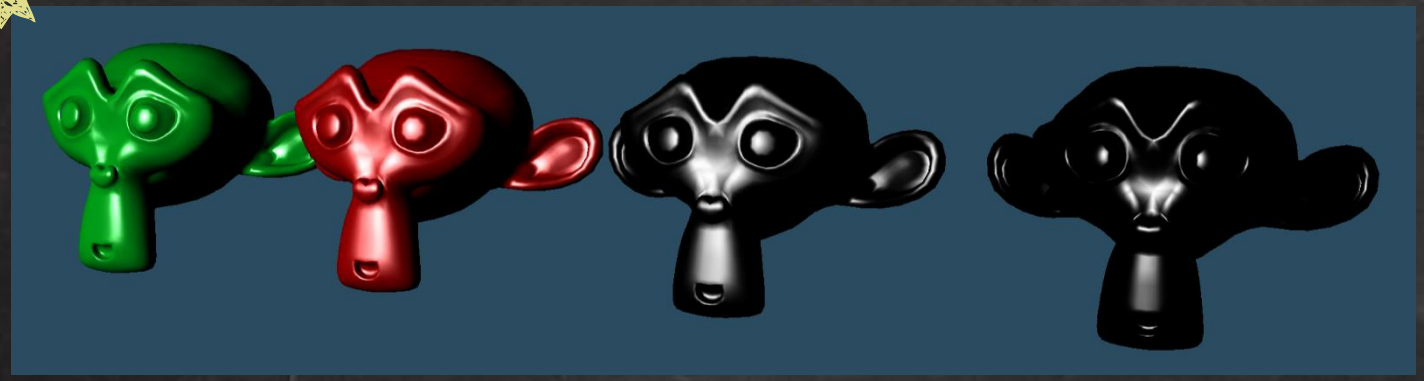


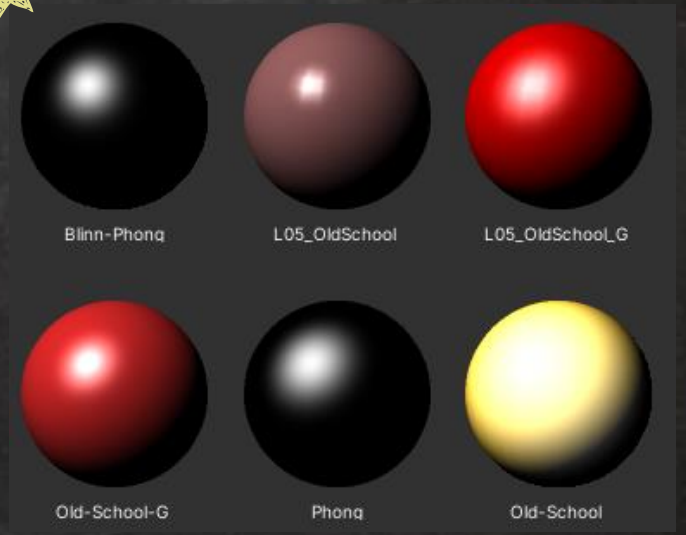






04 四组





06 总结

姓名	代码作业	连连看作业	截图	打包	创意题
周川	✓	✓	✓	✓	葡萄可讲
赵井才	✓	✓	✓	✓	无
叶小芸	✓	✓	✓	✓	无
赵翔	✓	✓	✓	✓	HalfLambert+Phong 可讲
张	✓	✓	✓	✓	无
周翰林	✓	✓	缺大合照	✓	铁锈斑驳 可讲
杨易	✓	✓	✓	缺场景	无
廖宴楠	✓	✓	✓	✓	Phong高光叠色 不讲
冯超越	✓	✓	✓	✓	无
申伏琳	✓	✓	✓	缺场景	无
陈沛霖	✓	✓	✓	✓	无
顾友海	✓	✓	✓	✓	透光树脂效果 可讲
冷翰林	✓	✓	✓	✓	卡通材质+高光 可讲
王岩	✓	✓	✓	✓	奇异材质 可讲
宋歌	✓	✓	缺大合照	缺场景	无
冯晓晨	✓	✓	✓	✓	无
罗陈	✓	✓	✓	✓	Hatching+高光 可讲
昊	✓	✓	✓	✓	无

作业情况：

- 代码和连连看都没啥问题；
- 希望大家再多点创意尝试；

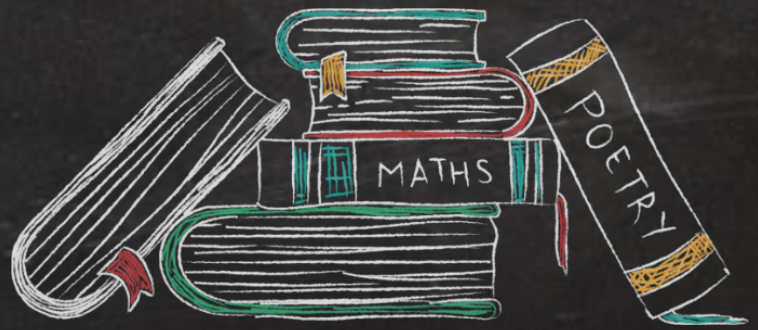
作业规范：

- 大家在Package包保存一个场景，可以展示所有样例；
- 记得截图大合照啊；

补录~别又忘了~

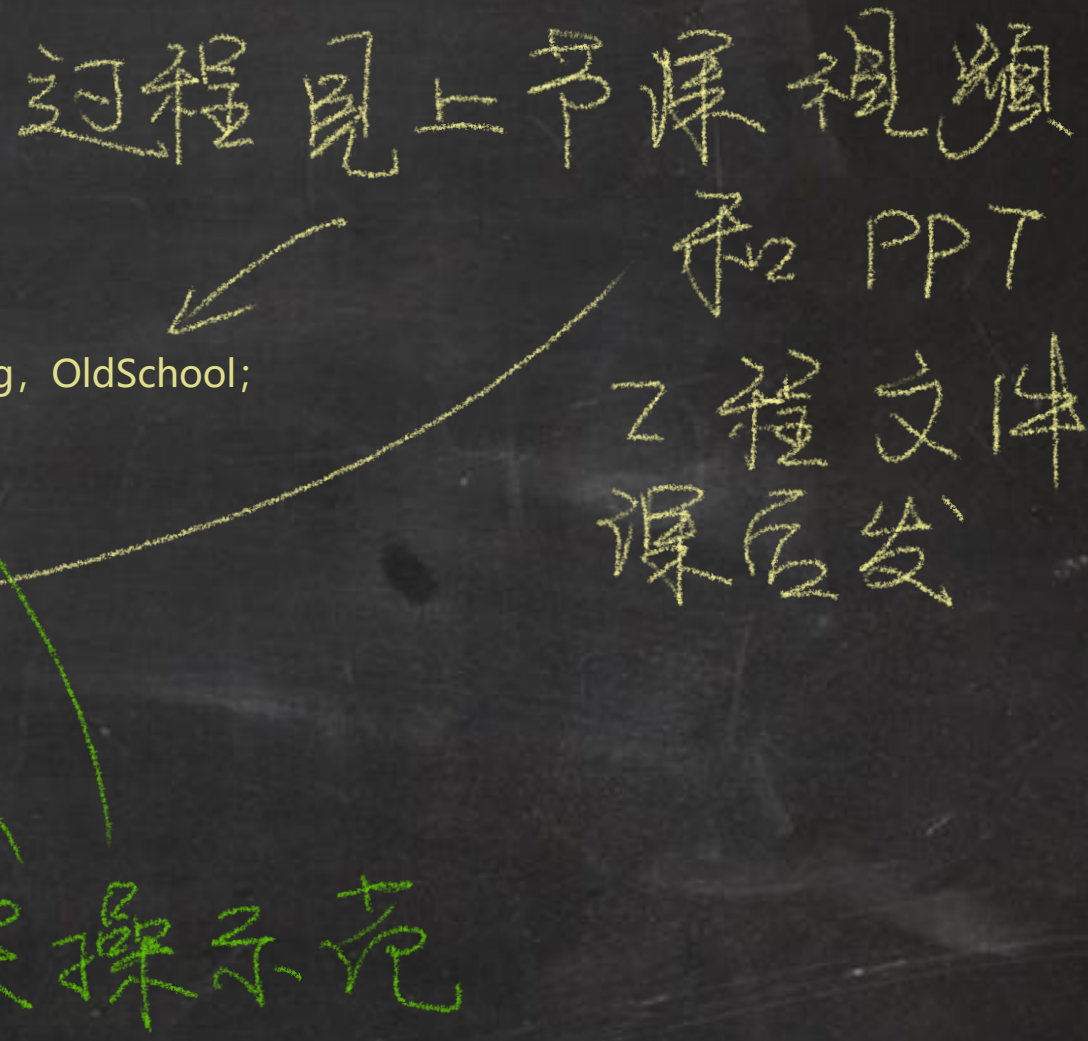
$\frac{24 \times 5}{33}$

2 作业·答案 批改 答疑

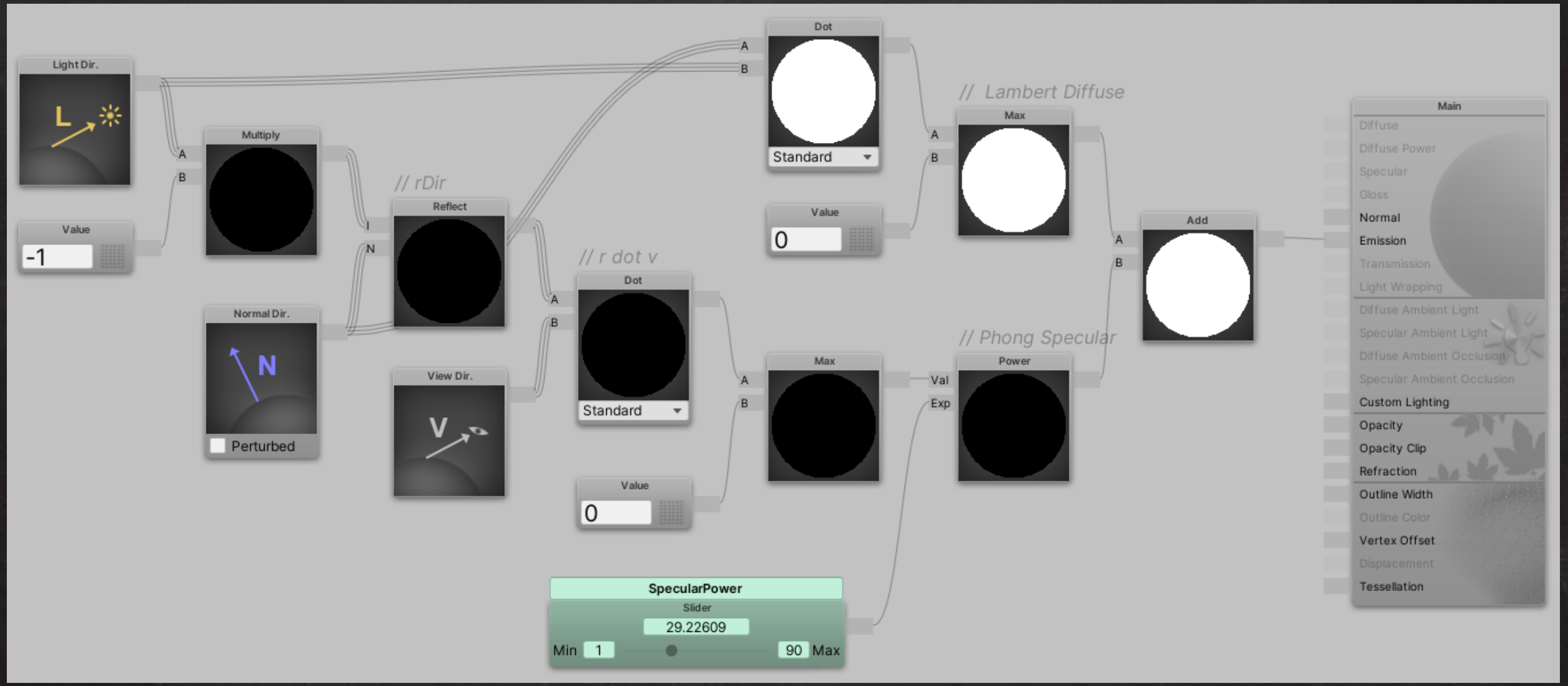


01 作业回顾

- 连连看作业：
  - 本节所有连连看例子：Phong，Blinn-Phong，OldSchool；
  - OldSchool·改（Lambert+Phong）
- 符文作业：
  - OldSchool
  - 尝试OldSchool·改
- 创意：
  - 不限，尽量根据已学知识发挥创意。



02 OldSchool·改 连连看



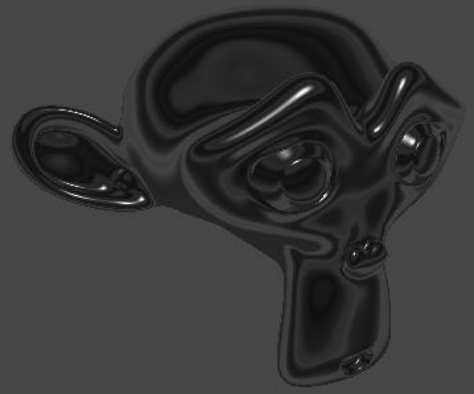
03 OldSchool·改 代码

1. 珍惜生命，善用模板；显然直接用OldSchool来改非常省事；
2. 改Shader路径名；
3. 面板参数声明不用改；
4. 输入参数声明不用改；
5. 输入结构，顶点Shader，输出结构也不用改；
6. 像素Shader需要修改：
    1. 向量准备：不需要hDir，但要追加rDir；rDir：光反射向量 = reflect(-lDir, nDir);
       注意 lDir是光方向的反方向；
    2. 点积结果准备：不需要ndoth，追加vdotr；
    3. 改blinnPhong为phong，phong=pow(max(0, vdotr), _SpecularPow);

```
Shader "AP1/L06/OldSchoolP" {
    Properties {
        _MainCol ("颜色", color) = (1.0, 1.0, 1.0, 1.0)
        _SpecularPow ("高光次幂", range(1, 90)) = 30
    }
    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 float3 _MainCol;
            uniform float _SpecularPow;
            // 输入结构
            struct VertexInput {
                float4 vertex : POSITION;    // 顶点信息 Get✔
                float4 normal : NORMAL;      // 法线信息 Get✔
            };
            // 输出结构
            struct VertexOutput {
                float4 posCS : SV_POSITION;      // 裁剪空间（暂理解为屏幕空间吧）顶点位置
                float4 posWS : TEXCOORD0;        // 世界空间顶点位置
                float3 nDirWS : TEXCOORD1;       // 世界空间法线方向
            };
            // 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.posCS = UnityObjectToClipPos( v.vertex );         // 变换顶点位置 OS>CS
                    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
                return o;                                               // 返回输出结构
            }
            // 输出结构>>>像素
            float4 frag(VertexOutput i) : COLOR {
                // 准备向量
                float3 nDir = i.nDirWS;
                float3 lDir = _WorldSpaceLightPos0.xyz;
                float3 rDir = reflect(-lDir, nDir);
                float3 vDir = normalize(_WorldSpaceCameraPos.xyz - i.posWS.xyz);
                // 准备点积结果
                float ndotl = dot(nDir, lDir);
                float vdotr = dot(vDir, rDir);
                // 光照模型
                float lambert = max(0.0, ndotl);
                float phong = pow(max(0.0, vdotr), _SpecularPow);
                float3 finalRGB = _MainCol * lambert + phong;
                // 返回结果
                return float4(finalRGB, 1.0);
            }
            ENDCG
        }
    }
    FallBack "Diffuse"
}
```

04 批改

FakeEnvReflect



批改1

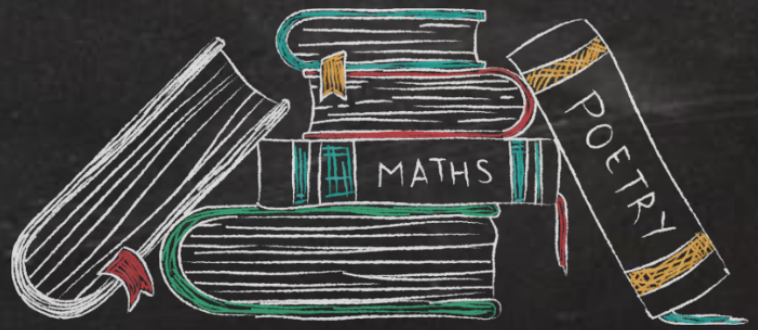
批改2

05 答疑

- 代码照着输入可以，离开范本照样懵逼；连连看不清楚类型匹配；
  - 学任何事情都是懵逼到牛逼的过程，至少你现在能写，已经胜过市面上一些假TA一筹了；
  - 类型匹配问题注意看结点接口，有类型限定遵循类型限定；（示范）
- VS SF效果不一致；
  - 原因在于，SF代码逐个像素对nDir做了归一化，VS代码没做；（示范）
- Hatching排线规整化问题；
  - 解决方法超纲了，不能用模型某一点的深度，要有模型中心得深度；
  - 模型中心深度 = UnityObjectToClipPos(float4(0,0,0,0)).w；（示范）
- 创意作业无头绪；
  - 武器库样式不足，创意难正常；
  - 把一样武器变着花玩，也是种创意；

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

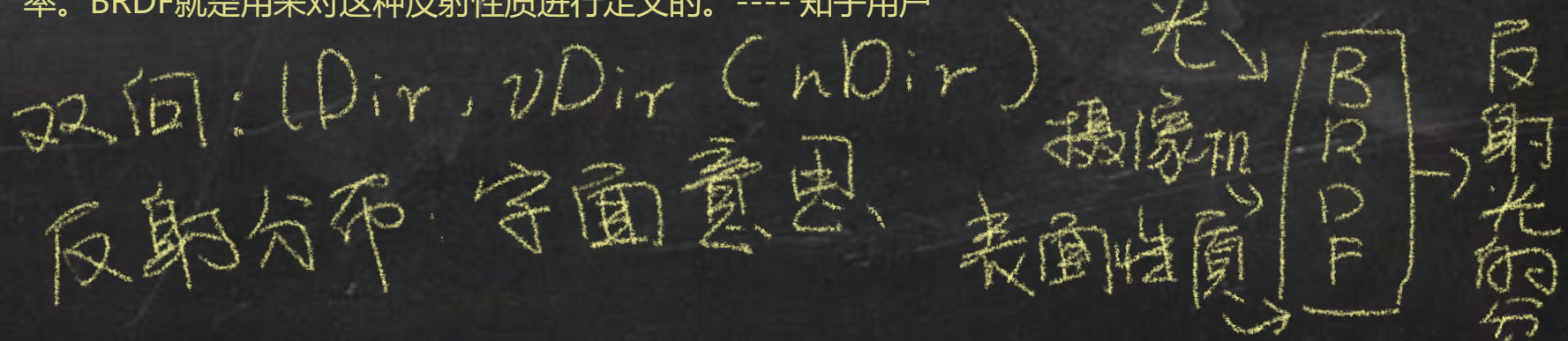
3 情报·BRDF



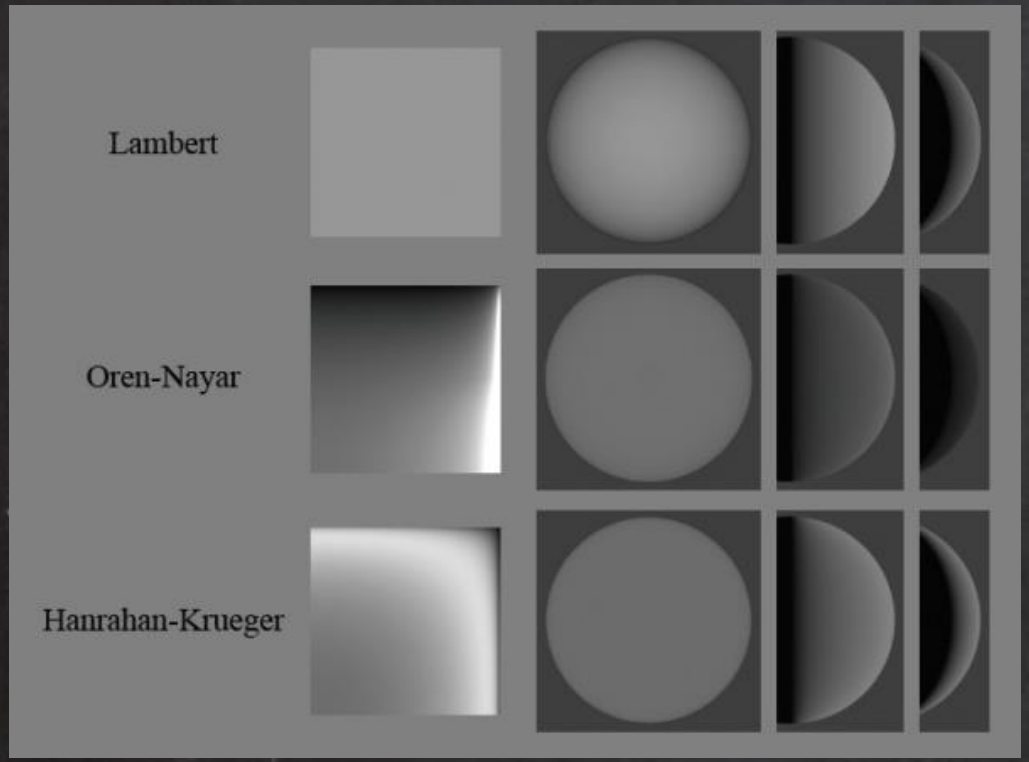
01 BRDF 双向反射分布函数

- 双向反射分布函数（bidirectional reflectance distribution function、BRDF），是一个定义光线在不透明表反射的四次元函数，基本式为： ，在这里 是指光线的入射方向，另外 是指光线反射的方向，除此之外，还有一个 代表法线，这个值的意义是在 方向的反射光线的辐射率和同一点上从 方向射入的光线的辐射率的比值。每一个 方向可以被参数化 为方位角 和天顶角 因此BRDF是一个四维函数。BRDF的单位是 sr, 其中 (sr) 是球面度的单位。 --- 李彦宏

- 想象你有一个不透明的桌面，一个激光发射器。你先让激光向下垂直地射在那个桌面上，这样你就可以在桌面上看到一个亮点，接着你从各个不同的方向来观察那个亮点，你会发现亮点的亮度随着观察方向的不同而发生了改变。然后你站着不动，改变激光发射方向和桌面的夹角，你又会发现亮点的亮度发生了改变。这就是说，一个表面对不同的光线入射角和反射角的组合，拥有不同的反射率。BRDF就是用来对这种反射性质进行定义的。 ---- 知乎用户



02 君の名は



- 真的大佬都是这样，你没见过他，也不暗恋他，但是不得不天天叨他的名字，为他掉发，直到死也忘不了。。。

03 来自迪士尼的爱·BRDF浏览器

BRDF Explorer

https://github.com/wdas/brdf/downloads

(示范)