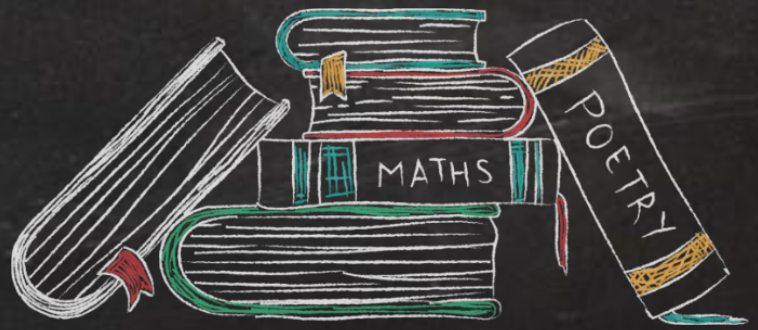
今日内容

- 幻术·筑基·菲涅尔
- 幻术·化神·Matcap
- 幻术·化神·Cubemap
- 符文·化神·Matcap
- 符文·化神·Cubemap
- 课业修行

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

1 幻术·筑基·菲涅尔



01 菲涅尔现象

菲涅尔现象：真实世界中，除了金属之外其它物质，视线垂直于表面时，
反射较弱，而当视线非垂直表面时，夹角越小，反射越明显。

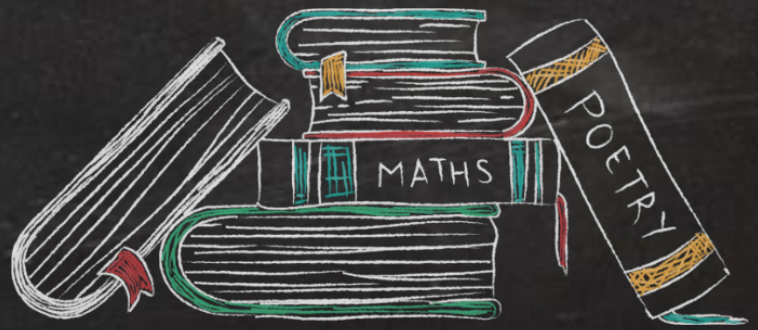
02 菲涅尔结点实现

算法：Fresnel = pow(1-ndotv，powVal)

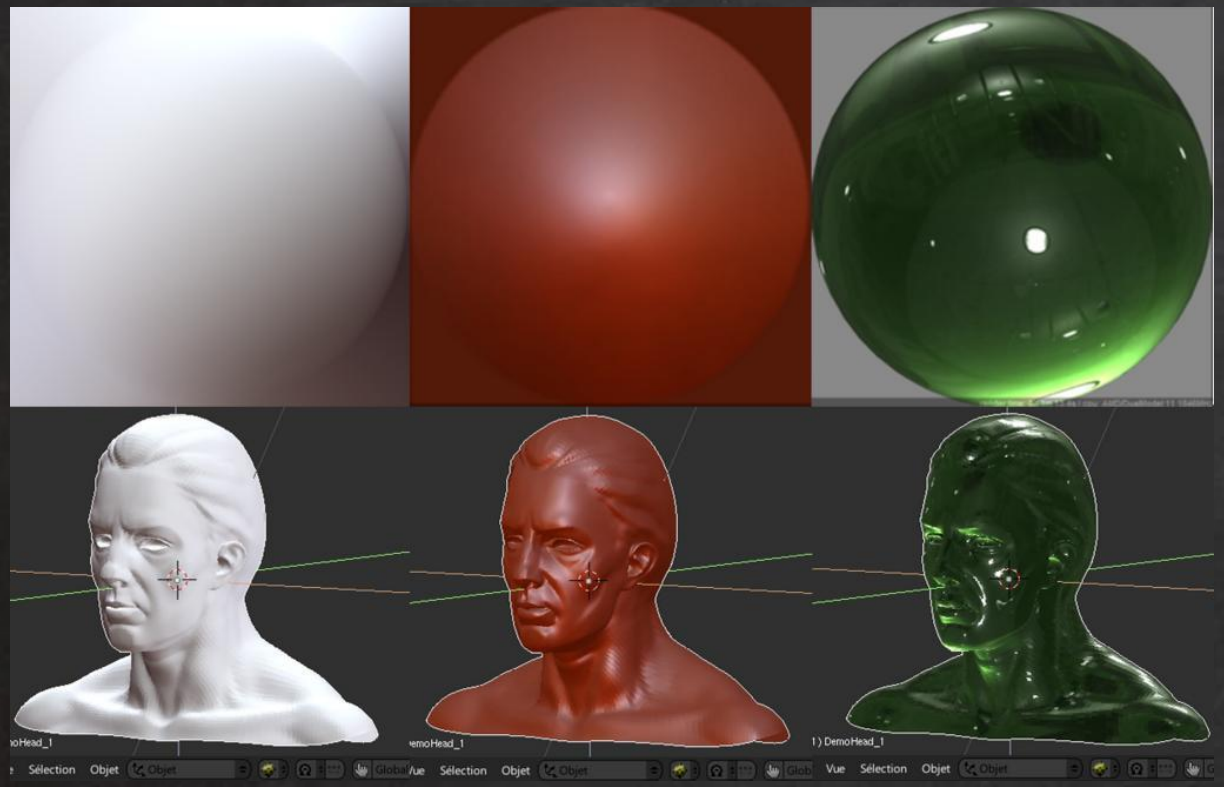
- ldotv：理解为光从眼睛发出时的Lambert；中间亮，边缘暗；
- 1-ldotv：黑白反相，中间暗，边缘亮；
- power：套一个power控制边缘亮的范围；

(演示)

2 幻术·化神·Matcap

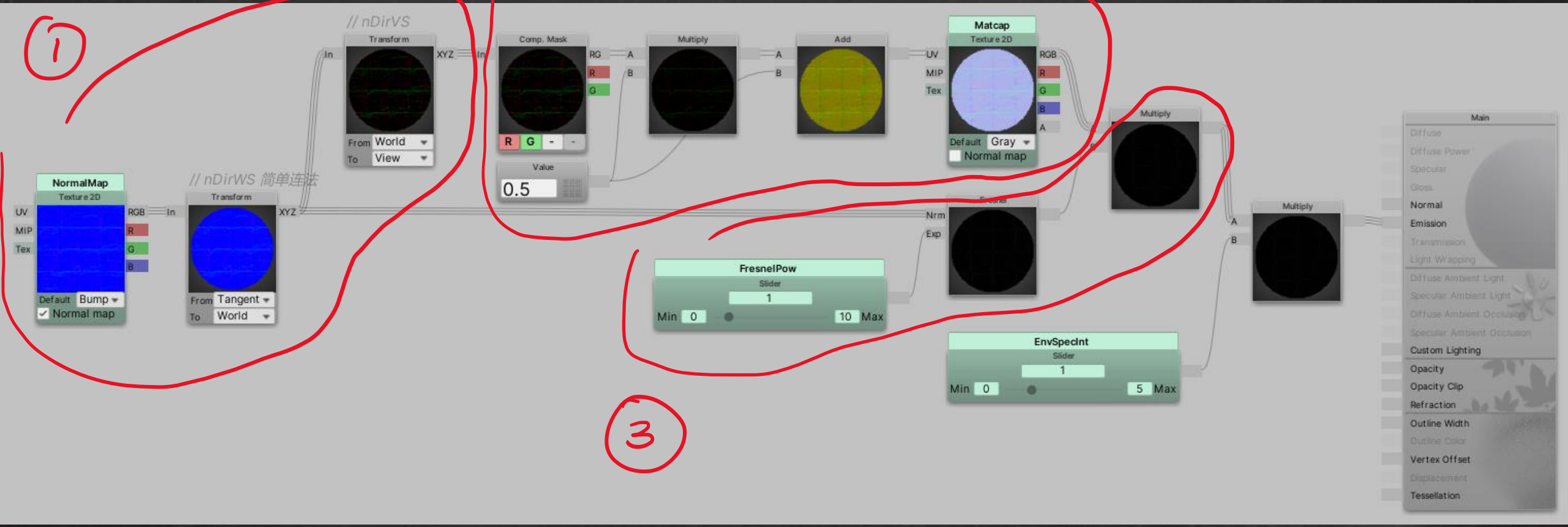


01 什么是Matcap?



- 一种无视BRDF，将BRDF渲染结果，用View空间法线朝向，直接映射到模型表面的流氓算法；
- 常用来模拟环境反射；

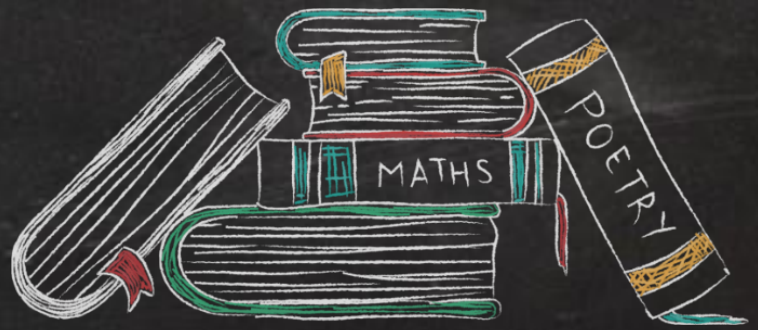
02 结印·Matcap



算法：

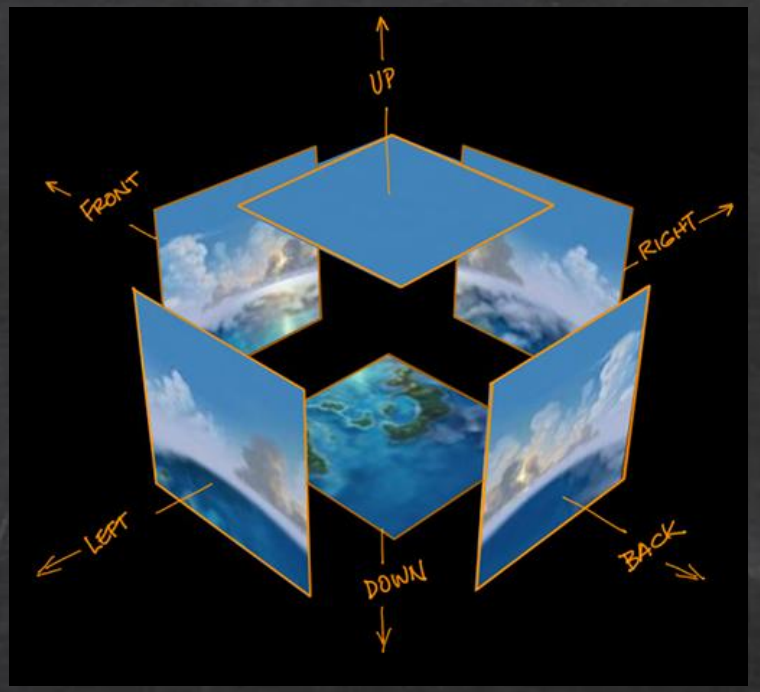
1. 将nDir从切线空间转到观察空间；
2. 取RG通道Remap到(0~1)，作为UV对Matcap图采样；
3. 叠加菲涅尔效果，以模拟金属和非金属不同质感；

3 结印·化神·Cubemap

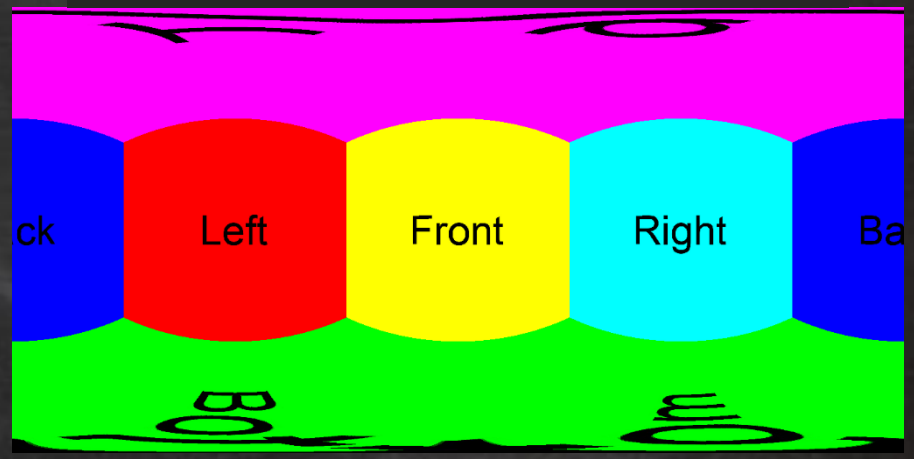


01 什么是Cubemap?

简单理解就是：全景图

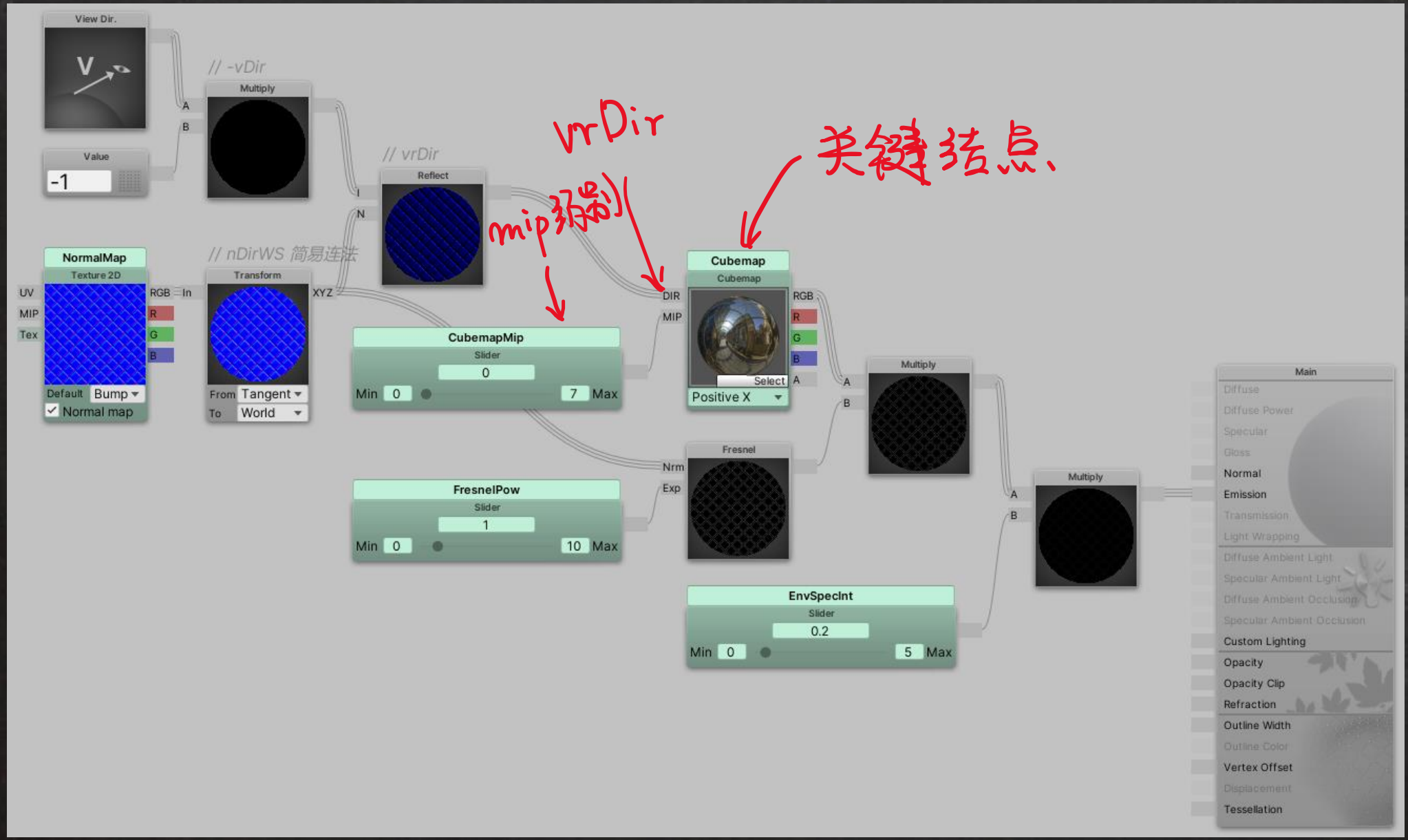


Cube字面意思的由来



那些不像Cube的Cubemap是映射后的结果

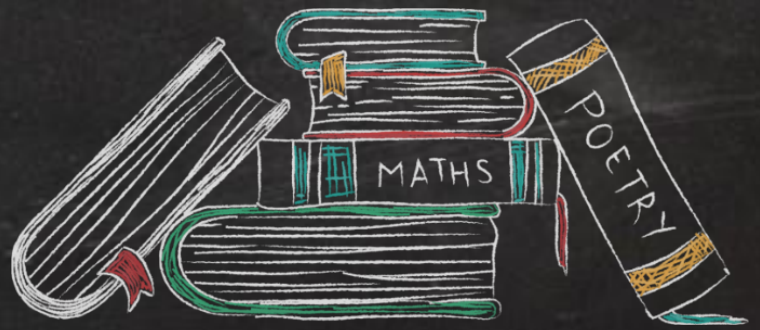
02 结印·Cubemap



1. SF中有现成Cubemap结点，实现起来并不复杂；注意两个输入结点；
2. 第一个为vrDir，即观察方向的反射方向；注意vDir同样为观察方向的反方向；做reflect之前要乘以-1；
3. Mip为Cubemap的Mipmap等级；

(演示)

4 符文·化神·Matcap



01 matcap代码实现

1. 以手写FlatCol（第3课）作为模板，复制粘贴大法；
2. 修改Shader路径名；
3. 面板参数Copy自L09_Mapcap_SF，并修改；
4. 按面板参数一一申明输入参数，注意类型匹配；
5. 输入结构追加uv0，normal，tangent；参考L08_Normal；
6. 输出结构追加uv0，posWS，nDirWS，tDirWS，bDirWS；参考L08_Normal；
7. 顶点Shader中增加相应代码；

```
Shader "AP01/L09/Matcap" {
    Properties {
        _NormalMap  ("法线贴图", 2D) = "bump" {}
        _Matcap     ("Matcap", 2D) = "gray" {}
        _FresnelPow ("菲涅尔次幂", Range(0, 10)) = 1
        _EnvSpecInt ("环境镜面反射强度", Range(0, 5)) = 1
    }
```

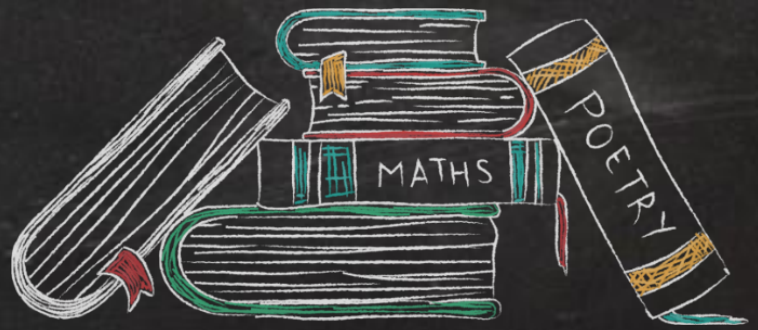
```
CGPROGRAM
#pragma vertex vert
#pragma fragment frag
#include "UnityCG.cginc"
#pragma multi_compile_fwdbase_fullshadows
#pragma target 3.0
// 输入参数
uniform sampler2D _NormalMap;
uniform sampler2D _Matcap;
uniform float _FresnelPow;
uniform float _EnvSpecInt;
// 输入结构
struct VertexInput {
    float4 vertex   : POSITION;     // 顶点信息
    float2 uv0      : TEXCOORD0;    // uv信息
    float3 normal   : NORMAL;       // 法线信息
    float4 tangent  : TANGENT;      // 切线信息
};
// 输出结构
struct VertexOutput {
    float4 pos : SV_POSITION;       // 屏幕顶点位置
    float2 uv0 : TEXCOORD0;         // uv信息
    float4 posWS : TEXCOORD1;       // 世界顶点位置
    float3 nDirWS : TEXCOORD2;      // 世界法线方向
    float3 tDirWS : TEXCOORD3;      // 世界切线方向
    float3 bDirWS : TEXCOORD4;      // 世界副切线方向
};
// 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 );
        o.uv0 = v.uv0;                                      // 传递uv信息
        o.posWS = mul(unity_ObjectToWorld, v.vertex);   // 顶点位置 OS>WS
        o.nDirWS = UnityObjectToWorldNormal(v.normal);  // 法线方向 OS>WS
        o.tDirWS = normalize(mul(unity_ObjectToWorld, float4(v.tangent.xyz, 0.0)).xyz); // 切线方向 OS>WS
        o.bDirWS = normalize(cross(o.nDirWS, o.tDirWS) * v.tangent.w);  // 根据nDir tDir求bDir
    return o;                                       // 将输出结构 输出
}
```

02 matcap代码实现

1. 向量准备：
    1. 采样并解码nDirTS；
    2. 构造TBN矩阵，计算nDirWS；
    3. 进一步变换为nDirVS；
    4. 计算vDirWS；
2. 中间量准备：
    1. vdotn：用于计算菲涅尔；
    2. matcapUV：用于采样Matcap；
3. 光照模型：
    1. 采样matcap；
    2. 计算fresnel；
    3. 计算最终环境镜面反射；
4. 返回值；

```
// 输出结构>>>像素
float4 frag(VertexOutput i) : COLOR {
    // 准备向量
    float3 nDirTS = UnpackNormal(tex2D(_NormalMap, i.uv0)).rgb;
    float3x3 TBN = float3x3(i.tDirWS, i.bDirWS, i.nDirWS);
    float3 nDirWS = normalize(mul(nDirTS, TBN));        // 计算nDirVS 计算Fresnel
    float3 nDirVS = mul(UNITY_MATRIX_V, nDirWS);        // 计算MatcapUV
    float3 vDirWS = normalize(_WorldSpaceCameraPos.xyz - i.posWS.xyz); // 计算Fresnel

    // 准备中间变量
    float vdotn = dot(vDirWS, nDirWS);
    float2 matcapUV = nDirVS.rg * 0.5 + 0.5;

    // 光照模型
    float3 matcap = tex2D(_Matcap, matcapUV);
    float fresnel = pow(max(0.0, 1.0 - vdotn), _FresnelPow);
    float3 envSpecLighting = matcap * fresnel * _EnvSpecInt;

    // 返回值
    return float4(envSpecLighting, 1.0);
}
```

5 符文·化神·Cubemap



01 cubemap代码实现

1. 以手写FlatCol（第3课）作为模板，复制粘贴大法；
2. 修改Shader路径名；
3. 面板参数Copy自L09_Cubemap_SF，并修改；
4. 按面板参数一一申明输入参数，注意类型匹配；
5. 输入结构，输出结构，顶点Shader与L09_Matcap一致，整段Copy；

```
Shader "AP01/L09/Cubemap" {
    Properties {
        _Cubemap     ("环境球", Cube) = "_Skybox" {}
        _NormalMap   ("法线贴图", 2D) = "bump" {}
        _CubemapMip  ("环境球Mip", Range(0, 7)) = 0
        _FresnelPow  ("菲涅尔次幂", Range(0, 5)) = 1
        _EnvSpecInt  ("环境镜面反射强度", Range(0, 5)) = 0.2
    }
```

```
CGPROGRAM
#pragma vertex vert
#pragma fragment frag
#include "UnityCG.cginc"
#pragma multi_compile_fwdbase_fullshadows
#pragma target 3.0
// 输入参数
uniform samplerCUBE _Cubemap;
uniform sampler2D _NormalMap;
uniform float _CubemapMip;
uniform float _FresnelPow;
uniform float _EnvSpecInt;
// 输入结构
struct VertexInput {
    float4 vertex   : POSITION;     // 顶点信息
    float2 uv0      : TEXCOORD0;    // uv信息
    float3 normal   : NORMAL;       // 法线信息
    float4 tangent  : TANGENT;      // 切线信息
};
// 输出结构
struct VertexOutput {
    float4 pos : SV_POSITION;       // 屏幕顶点位置
    float2 uv0 : TEXCOORD0;         // uv信息
    float4 posWS : TEXCOORD1;       // 世界顶点位置
    float3 nDirWS : TEXCOORD2;      // 世界法线方向
    float3 tDirWS : TEXCOORD3;      // 世界切线方向
    float3 bDirWS : TEXCOORD4;      // 世界副切线方向
};
// 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 );
        o.uv0 = v.uv0;                                          // 传递uv信息
        o.posWS = mul(unity_ObjectToWorld, v.vertex);   // 顶点位置 OS>WS
        o.nDirWS = UnityObjectToWorldNormal(v.normal);  // 法线方向 OS>WS
        o.tDirWS = normalize(mul(unity_ObjectToWorld, float4(v.tangent.xyz, 0.0)).xyz); // 切线方向 OS>WS
        o.bDirWS = normalize(cross(o.nDirWS, o.tDirWS) * v.tangent.w);  // 根据nDir tDir求bDir
    return o;                                   // 将输出结构 输出
}
```

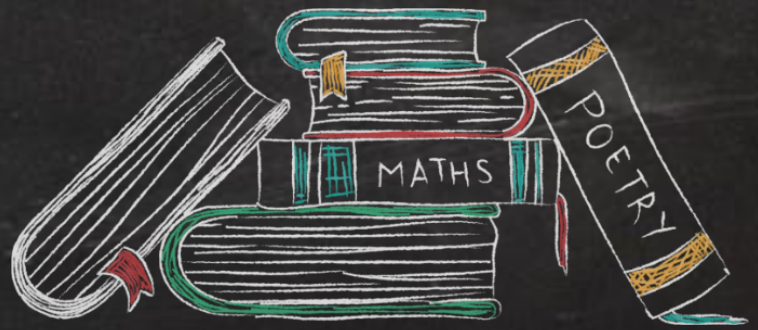
02 cubemap代码实现

1. 向量准备：
    1. 采样并解码nDirTS；
    2. 构造TBN矩阵，计算nDirWS；
    3. 计算vDirWS；
    4. 用nDirWS，vDirWS计算vrDirWS；
2. 中间量准备：
    1. vdotn：用于计算菲涅尔；
3. 光照模型：
    1. 采样Cubemap；
        - 采样方法：texCUBElod(_Cubemap, float4)；
        - Float4参数：xyz：vrDir；w：mip等级；
    2. 计算fresnel；
    3. 计算最终环境镜面反射；
4. 返回值；

```
// 输出结构>>>像素
float4 frag(VertexOutput i) : COLOR {
    // 准备向量
    float3 nDirTS = UnpackNormal(tex2D(_NormalMap, i.uv0)).rgb;
    float3x3 TBN = float3x3(i.tDirWS, i.bDirWS, i.nDirWS);
    float3 nDirWS = normalize(mul(nDirTS, TBN));   // 计算Fresnel 计算vrDirWS
    float3 vDirWS = normalize(_WorldSpaceCameraPos.xyz - i.posWS.xyz);  // 计算Fresnel
    float3 vrDirWS = reflect(-vDirWS, nDirWS);// 采样Cubemap

    // 准备中间变量
    float vdotn = dot(vDirWS, nDirWS);

    // 光照模型
    float3 var_Cubemap = texCUBElod(_Cubemap, float4(vrDirWS, _CubemapMip)).rgb;
    float fresnel = pow(max(0.0, 1.0 - vdotn), _FresnelPow);
    float3 envSpecLighting = var_Cubemap * fresnel * _EnvSpecInt;

    // 返回值
    return float4(envSpecLighting, 1.0);
}
```

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

7 课业修行



01 可以搞点事情了

- 连连看作业：
  - Matcap，Cubemap；
  - OldSchoolPro = OldSchoolPlus + Normal + Cubemap；
- 符文作业：
  - Matcap；
  - OldSchoolPro；
- 创意作业：
  - 改进你的完备光照模型；