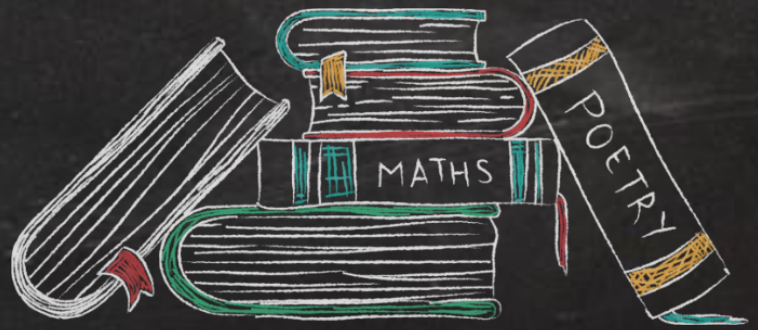
今日内容

- 情报·特效类大纲
- 符文·筑基·透切
- 符文·筑基·透混
- 符文·筑基·透叠
- 工具·化神·混合模式
- 任务委托

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

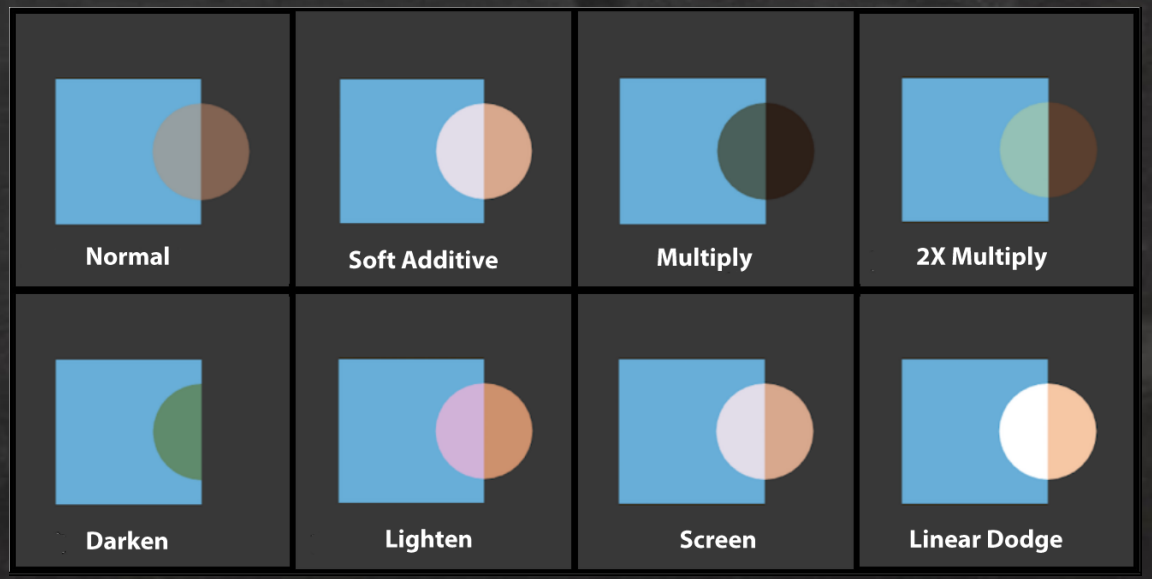
1 情报·特效类大纲



01 特效·透

透：

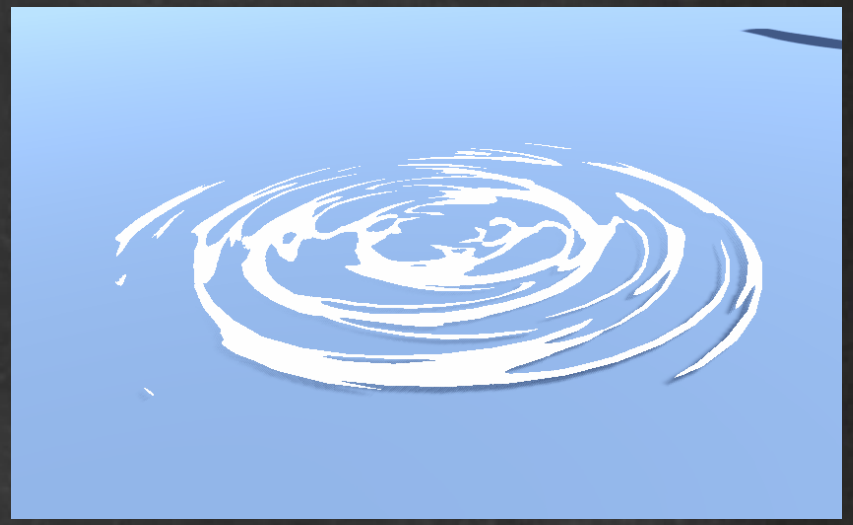
- AB：
- AD；
- AC；
- 自定义混合方式；



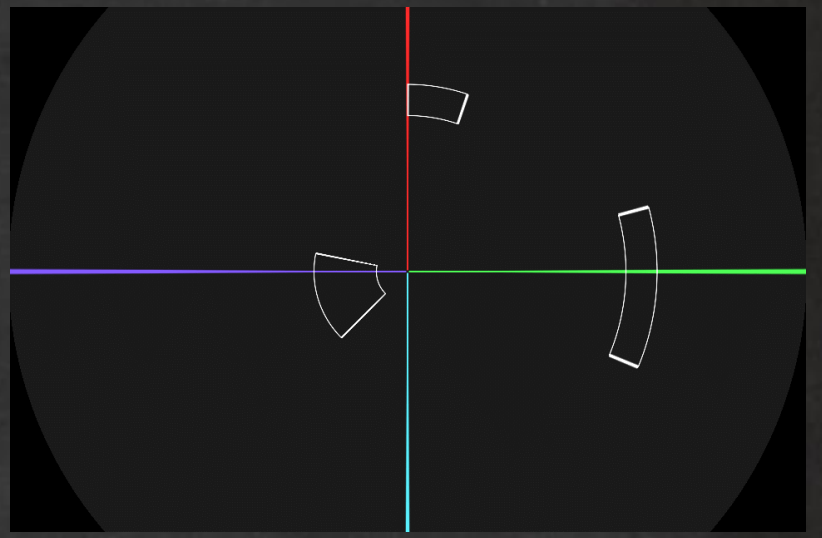
02 特效·动

动：

- 参数动画
- UV动画
  - UV流动
  - UV扰动
  - 序列帧动画
- 顶点动画
  - 顶点位置动画
  - 顶点颜色动画

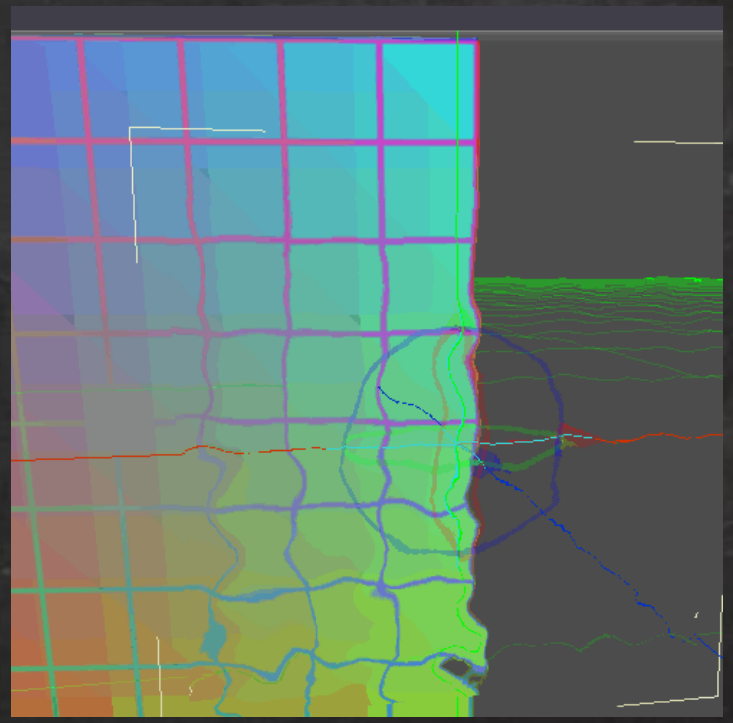


03 特效·映



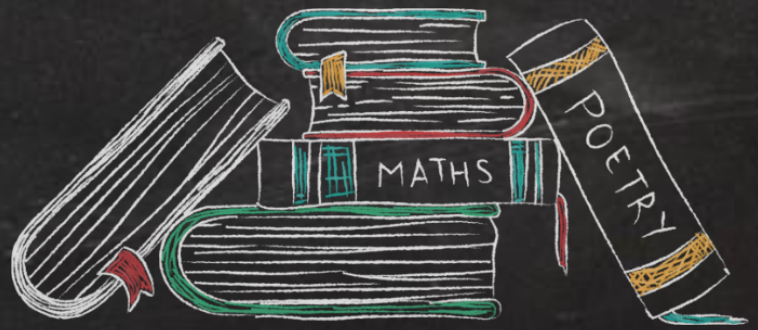
映：

- 极坐标
- 屏幕坐标UV
- 透明扭曲



$\frac{24\times5}{33}$

2 符文·筑基·透切



01 透切·AlphaCutout·AC

用途：

- 常用于复杂轮廓，明确边缘的物体表现，如：镂空金属，裙摆边缘，特定风格下的头发，树叶，等；
- 卡通渲染的特效表现；

优点：

- 没有排序问题；

缺点：

- 边缘效果太实；
- 移动端性能较差；

02 代码·AC

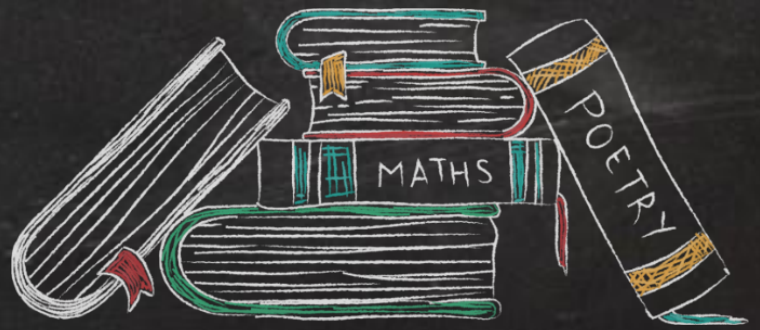
1. 以手写FlatCol（第3课）作为模板，CtrlCV伺候；
2. 修改Shader路径名；
3. 面板参数添加：
    1. _MainTex：主纹理 RGB颜色 A透贴；
    2. _Cutoff：透明剪切阈值；
4. 修改SubShaderTags；
    1. RenderType：修改为对应的TransparentCutout；
    2. ForceNoShadowCasting：设置为True，关闭投影；
    3. IgnoreProjector：设置为True，不响应投射器；
5. 对应声明输入参数；
6. 输入，输出，顶点Shader，增加UV相关代码；
7. 像素Shader：
    1. 对主纹理采样，RGB颜色 A透贴；
    2. 基于透贴灰度和透切阈值，产生透明效果；
    3. 返回值；

示范

```
Shader "AP01/L13/AC" {
    Properties {
        _MainTex ("RGB：颜色 A：透贴", 2d) = "gray"{}
        _Cutoff ("透切阈值", range(0.0, 1.0)) = 0.5
    }
    SubShader {
        Tags {
            "RenderType"="TransparentCutout"    // 对应改为Cutout
            "ForceNoShadowCasting"="True"       // 关闭阴影投射
            "IgnoreProjector"="True"            // 不响应投射器
        }
        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 _MainTex; uniform float4 _MainTex_ST;
            uniform half _Cutoff;
            // 输入结构
            struct VertexInput {
                float4 vertex : POSITION;       // 顶点位置 总是必要
                float2 uv : TEXCOORD0;          // UV信息 采样贴图用
            };
            // 输出结构
            struct VertexOutput {
                float4 pos : SV_POSITION;       // 顶点位置 总是必要
                float2 uv : TEXCOORD0;          // UV信息 采样贴图用
            };
            // 输入结构>>>顶点Shader>>>输出结构
            VertexOutput vert (VertexInput v) {
                VertexOutput o = (VertexOutput)0;
                    o.pos = UnityObjectToClipPos( v.vertex);    // 顶点位置 OS>CS
                    o.uv = TRANSFORM_TEX(v.uv, _MainTex);       // UV信息 支持TilingOffset
                return o;
            }
            // 输出结构>>>像素
            half4 frag(VertexOutput i) : COLOR {
                half4 var_MainTex = tex2D(_MainTex, i.uv);      // 采样贴图 RGB颜色 A透贴
                clip(var_MainTex.a - _Cutoff);                  // 透明剪切
                return var_MainTex;                             // 返回值
            }
            ENDCG
        }
    }
}
```

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

3 符文·筑基·透混



01 透混·AlphaBlend·AB

用途：

- 常用于复杂轮廓，无明确边缘的物体表现；
- 常用于半透明的物体表现；
- 一般的特效表现，打底用；

优点：

- 移动端性能较好；
- 边缘效果较好

缺点：

- 有排序问题；

02 代码·AB

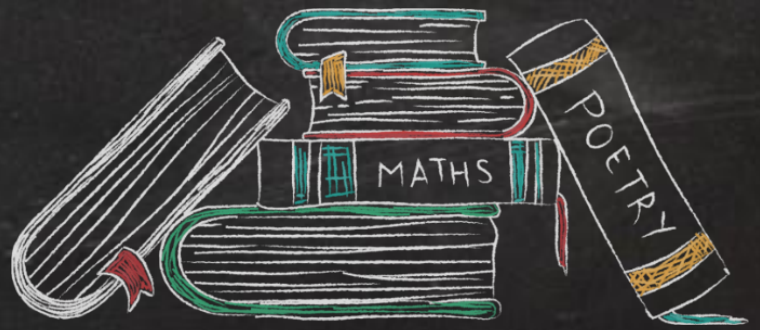
1. 以AC作为模板，CtrlCV伺候；
2. 修改Shader路径名；
3. 面板参数保留：
    1. _MainTex：主纹理 RGB颜色 A透贴；
4. 修改SubShaderTags；
    1. Queue：渲染队列修改为对应的Transparent；
    2. RenderType：修改为对应的Transparent；
    3. ForceNoShadowCasting：设置为True，关闭投影；
    4. IgnoreProjector：设置为True，不响应投射器；
5. 修改混合方式：Blend One/SrcAlpha OneMinusSrcAlpha；
6. 对应声明输入参数；
7. 输入，输出，顶点Shader，不用改；
8. 像素Shader：
    1. 对主纹理采样，RGB颜色 A透贴；
    2. 返回值；

示范

```
Shader "AP01/L13/AB" {
    Properties {
        _MainTex ("RGB: 颜色 A: 透贴", 2d) = "gray"{}
    }
    SubShader {
        Tags {
            "Queue"="Transparent"               // 调整渲染顺序
            "RenderType"="Transparent"          // 对应改为Cutout
            "ForceNoShadowCasting"="True"       // 关闭阴影投射
            "IgnoreProjector"="True"            // 不响应投射器
        }
        Pass {
            Name "FORWARD"
            Tags {
                "LightMode"="ForwardBase"
            }
            Blend One OneMinusSrcAlpha  // 修改混合方式One/SrcAlpha OneMinusSrcAlpha

            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 _MainTex; uniform float4 _MainTex_ST;
            // 输入结构
            struct VertexInput {
                float4 vertex : POSITION;       // 顶点位置 总是必要
                float2 uv : TEXCOORD0;          // UV信息 采样贴图用
            };
            // 输出结构
            struct VertexOutput {
                float4 pos : SV_POSITION;       // 顶点位置 总是必要
                float2 uv : TEXCOORD0;          // UV信息 采样贴图用
            };
            // 输入结构>>>顶点Shader>>>输出结构
            VertexOutput vert (VertexInput v) {
                VertexOutput o = (VertexOutput)0;
                    o.pos = UnityObjectToClipPos( v.vertex);    // 顶点位置 OS>CS
                    o.uv = TRANSFORM_TEX(v.uv, _MainTex);       // UV信息 支持TilingOffset
                return o;
            }
            // 输出结构>>>像素
            half4 frag(VertexOutput i) : COLOR {
                half4 var_MainTex = tex2D(_MainTex, i.uv);      // 采样贴图 RGB颜色 A透贴
                return var_MainTex;                             // 返回值
            }
            ENDCG
        }
    }
}
```

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

4 符文·筑基·透置



01 透叠·Addtive·AD

用途：

- 常用于发光体，辉光的表现；
- 一般的特效表现，提亮用；

问题：

- 有排序问题；
- 多层叠加容易堆爆性能(OverDraw)；
- 作为辉光效果，通常可用后处理代替；

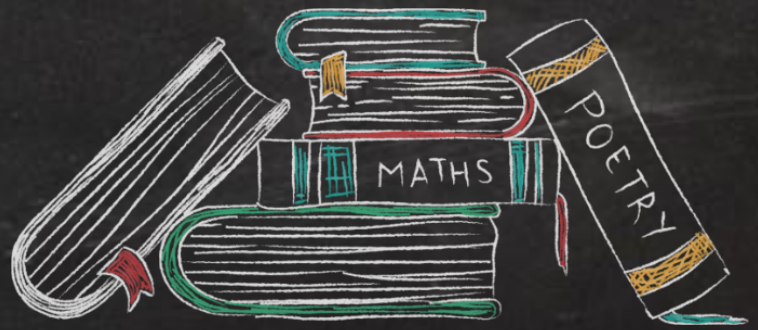
02 代码·AD

1. 以AB作为模板，CtrlCV伺候；
2. 修改Shader路径名；
3. 面板参数，不用改；
4. SubShaderTags，不用改；
5. 混合模式修改为：Blend One One；
6. 输入参数，不用改；
7. 输入，输出，顶点Shader，不用改；
8. 像素Shader：
    1. 对主纹理采样，RGB颜色即可；
    2. 返回值；

示范

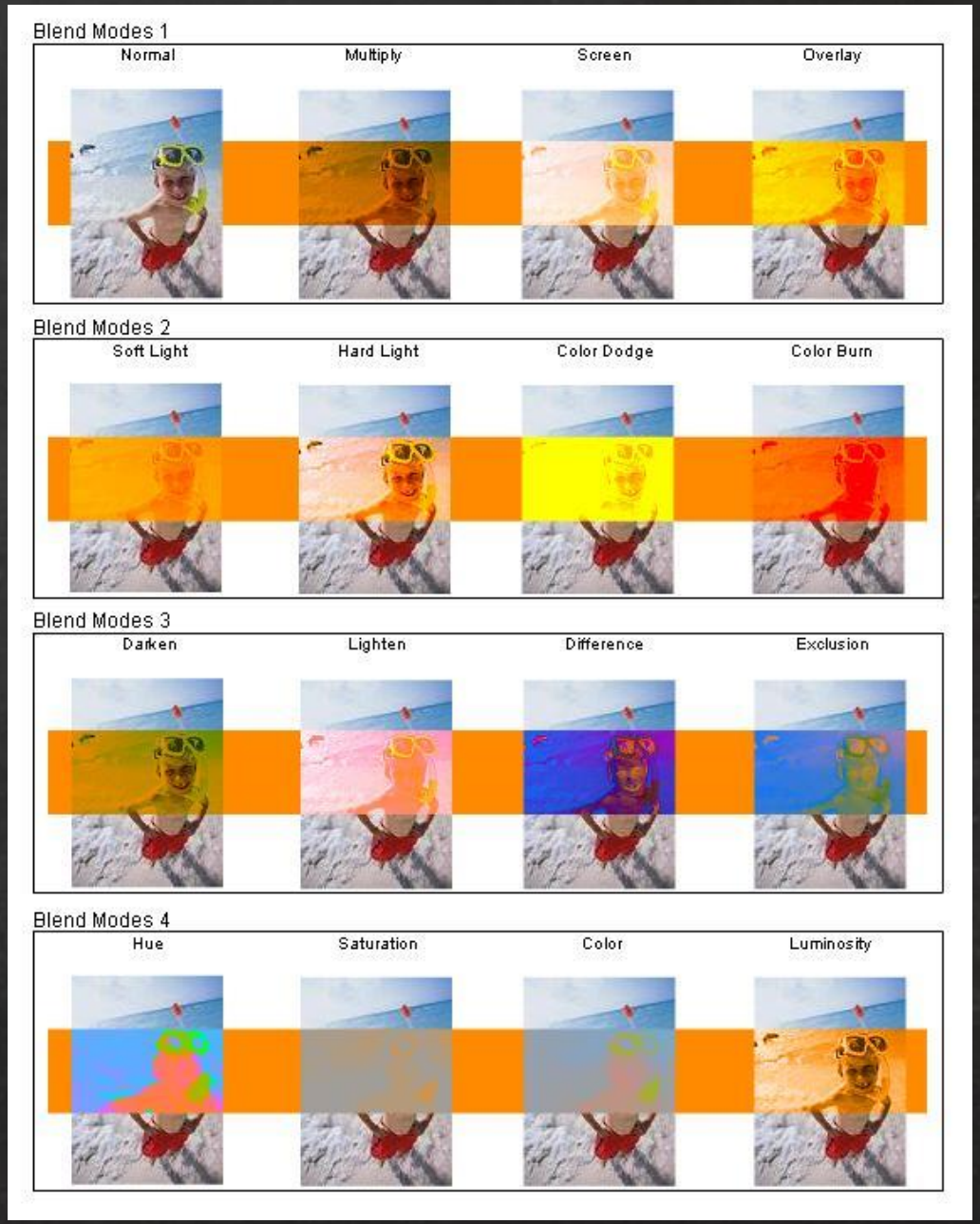
```
Shader "AP01/L13/AD" {
    Properties {
        _MainTex ("RGB: 颜色", 2d) = "gray"{}
    }
    SubShader {
        Tags {
            "Queue"="Transparent"                // 调整渲染顺序
            "RenderType"="Transparent"           // 对应改为Cutout
            "ForceNoShadowCasting"="True"        // 关闭阴影投射
            "IgnoreProjector"="True"             // 不响应投射器
        }
        Pass {
            Name "FORWARD"
            Tags {
                "LightMode"="ForwardBase"
            }
            Blend One One                        // 修改混合方式

            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 _MainTex; uniform float4 _MainTex_ST;
            // 输入结构
            struct VertexInput {
                float4 vertex : POSITION;        // 顶点位置 总是必要
                float2 uv : TEXCOORD0;           // UV信息 采样贴图用
            };
            // 输出结构
            struct VertexOutput {
                float4 pos : SV_POSITION;        // 顶点位置 总是必要
                float2 uv : TEXCOORD0;           // UV信息 采样贴图用
            };
            // 输入结构>>>顶点Shader>>>输出结构
            VertexOutput vert (VertexInput v) {
                VertexOutput o = (VertexOutput)0;
                    o.pos = UnityObjectToClipPos( v.vertex);     // 顶点位置 OS>CS
                    o.uv = TRANSFORM_TEX(v.uv, _MainTex);        // UV信息 支持TilingOffset
                return o;
            }
            // 输出结构>>>像素
            half4 frag(VertexOutput i) : COLOR {
                half3 var_MainTex = tex2D(_MainTex, i.uv).rgb;  // 采样贴图 RGB颜色 A透贴不必须
                return half4(var_MainTex, 1.0);                 // 返回值
            }
            ENDCG
        }
    }
}
```

5 工具·化神·混合模式



01 更多混合模式

- AB，AC，AD只是非常多种混合模式中常用的几种；足够应付一般项目的特效需求；
- 不排除某些风格特殊，或者特殊用途的场景需要用到其他混合模式；一般我们提供工具给美术做探索；
- 探索完毕之后还是要收敛方案，更少的美术选项可让美术流程更可控；



02 混合原理

Src*SrcFactor op Dst*DstFactor

- Src：源，当前Shader绘制的结果；
- Dst：目标，当前Shader绘制前的背景；
- SrcFactor：源乘子，多种形式如下表；
- DstFactor：目标乘子，多种形式如下表；
- Op：混合运算符，多种形式如右表；

BlendFactor:

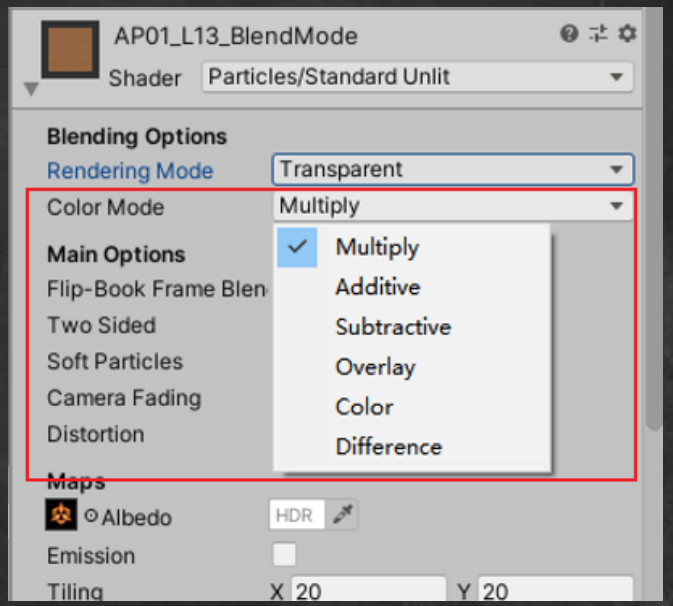
Zero	Blend factor is (0, 0, 0, 0).
One	Blend factor is (1, 1, 1, 1).
DstColor	Blend factor is (Rd, Gd, Bd, Ad).
SrcColor	Blend factor is (Rs, Gs, Bs, As).
OneMinusDstColor	Blend factor is (1 - Rd, 1 - Gd, 1 - Bd, 1 - Ad).
SrcAlpha	Blend factor is (As, As, As, As).
OneMinusSrcColor	Blend factor is (1 - Rs, 1 - Gs, 1 - Bs, 1 - As).
DstAlpha	Blend factor is (Ad, Ad, Ad, Ad).
OneMinusDstAlpha	Blend factor is (1 - Ad, 1 - Ad, 1 - Ad, 1 - Ad).
SrcAlphaSaturate	Blend factor is (f, f, f, 1); where f = min(As, 1 - Ad).
OneMinusSrcAlpha	Blend factor is (1 - As, 1 - As, 1 - As, 1 - As).

BlendOp:

Add	Add (s + d).
Subtract	Subtract.
ReverseSubtract	Reverse subtract.
Min	Min.
Max	Max.
Multiply	Multiply (Advanced OpenGL blending).
Screen	Screen (Advanced OpenGL blending).
Overlay	Overlay (Advanced OpenGL blending).
Darken	Darken (Advanced OpenGL blending).
Lighten	Lighten (Advanced OpenGL blending).
ColorDodge	Color dodge (Advanced OpenGL blending).
ColorBurn	Color burn (Advanced OpenGL blending).
HardLight	Hard light (Advanced OpenGL blending).
SoftLight	Soft light (Advanced OpenGL blending).
Difference	Difference (Advanced OpenGL blending).
Exclusion	Exclusion (Advanced OpenGL blending).
HSLHue	HSL Hue (Advanced OpenGL blending).
HSLSaturation	HSL saturation (Advanced OpenGL blending).
HSLColor	HSL color (Advanced OpenGL blending).
HSLLuminosity	HSL luminosity (Advanced OpenGL blending).

03 美术自定义混合面板

不封装·完全暴露



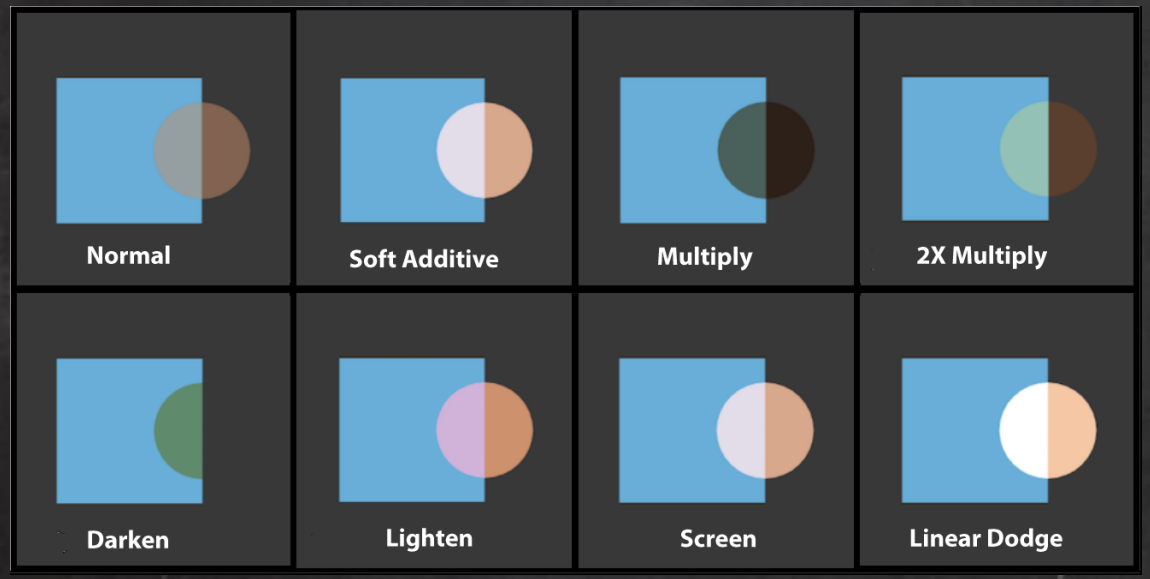
封装·有限选择

04 代码实现

1. 以AB作为模板，CtrlCV伺候；
2. 修改Shader路径名；
3. 面板参数：
    1. _MainTex：主纹理 RGB颜色 A透贴；
    2. _BlendSrc：混合源乘子；
    3. _BlendDst：混合目标乘子；
    4. _BlendOp：混合算符；
4. SubShaderTags，不用改；
5. 混合算符修改为：BlendOp [BlendOp]
6. 混合方式修改为：Blend [BlendSrc] [BlendDst]
7. 输入，输出，顶点Shader，不用改；
8. 像素Shader，不用改；

```
Shader "AP01/L13/BlendMode" {
    Properties {
        _MainTex ("RGB: 颜色 A: 透贴", 2d) = "gray"{}
        [Enum(UnityEngine.Rendering.BlendMode)]
        _BlendSrc ("混合源乘子", int) = 0
        [Enum(UnityEngine.Rendering.BlendMode)]
        _BlendDst ("混合目标乘子", int) = 0
        [Enum(UnityEngine.Rendering.BlendOp)]
        _BlendOp ("混合算符", int) = 0
    }
    SubShader {
        Tags {
            "Queue"="Transparent"                // 调整渲染顺序
            "RenderType"="Transparent"           // 对应改为Cutout
            "ForceNoShadowCasting"="True"        // 关闭阴影投射
            "IgnoreProjector"="True"             // 不响应投射器
        }
        Pass {
            Name "FORWARD"
            Tags {
                "LightMode"="ForwardBase"
            }
            BlendOp [_BlendOp]                   // 可自定义混合算符
            Blend [_BlendSrc] [_BlendDst]        // 可自定义混合模式

            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 _MainTex; uniform float4 _MainTex_ST;
            // 输入结构
            struct VertexInput {
                float4 vertex : POSITION;        // 顶点位置 总是必要
                float2 uv : TEXCOORD0;           // UV信息 采样贴图用
            };
            // 输出结构
            struct VertexOutput {
                float4 pos : SV_POSITION;        // 顶点位置 总是必要
                float2 uv : TEXCOORD0;           // UV信息 采样贴图用
            };
            // 输入结构>>>顶点Shader>>>输出结构
            VertexOutput vert (VertexInput v) {
                VertexOutput o = (VertexOutput)0;
                    o.pos = UnityObjectToClipPos( v.vertex);     // 顶点位置 OS>CS
                    o.uv = TRANSFORM_TEX(v.uv, _MainTex);        // UV信息 支持TilingOffset
                return o;
            }
            // 输出结构>>>像素
            half4 frag(VertexOutput i) : COLOR {
                half4 var_MainTex = tex2D(_MainTex, i.uv);       // 采样贴图 RGB颜色 A透贴
                return var_MainTex;                              // 返回值
            }
            ENDCG
        }
    }
}
```

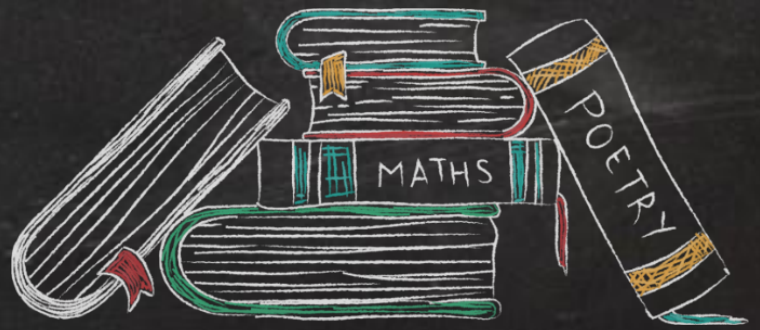
05 常用模式




```
Blend SrcAlpha OneMinusSrcAlpha  // Normal

Blend OneMinusDstColor One  // Soft Additive

Blend DstColor Zero  // Multiply

Blend DstColor SrcColor  // 2x Multiply

BlendOp Min    // Darken
Blend One One

BlendOp Max   // Lighten
Blend One One

Blend OneMinusDstColor One // Screen
Blend One OneMinusSrcColor // Same as above

Blend One One // Linear Dodge
```

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

6 任务委托



01 作业

必做作业：

- 代码：AB，AC，AD，BlendMode；

创意作业：

- 课内知识自由发挥；