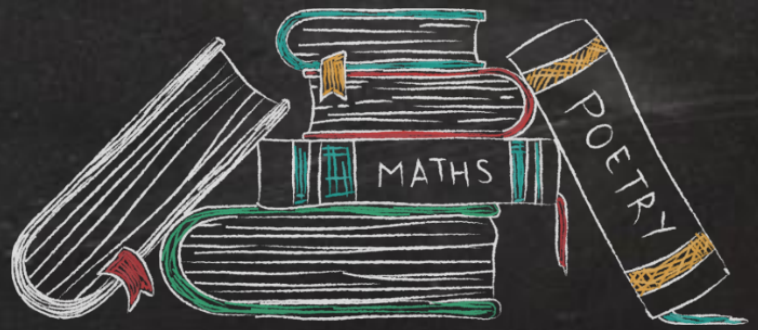
今日内容

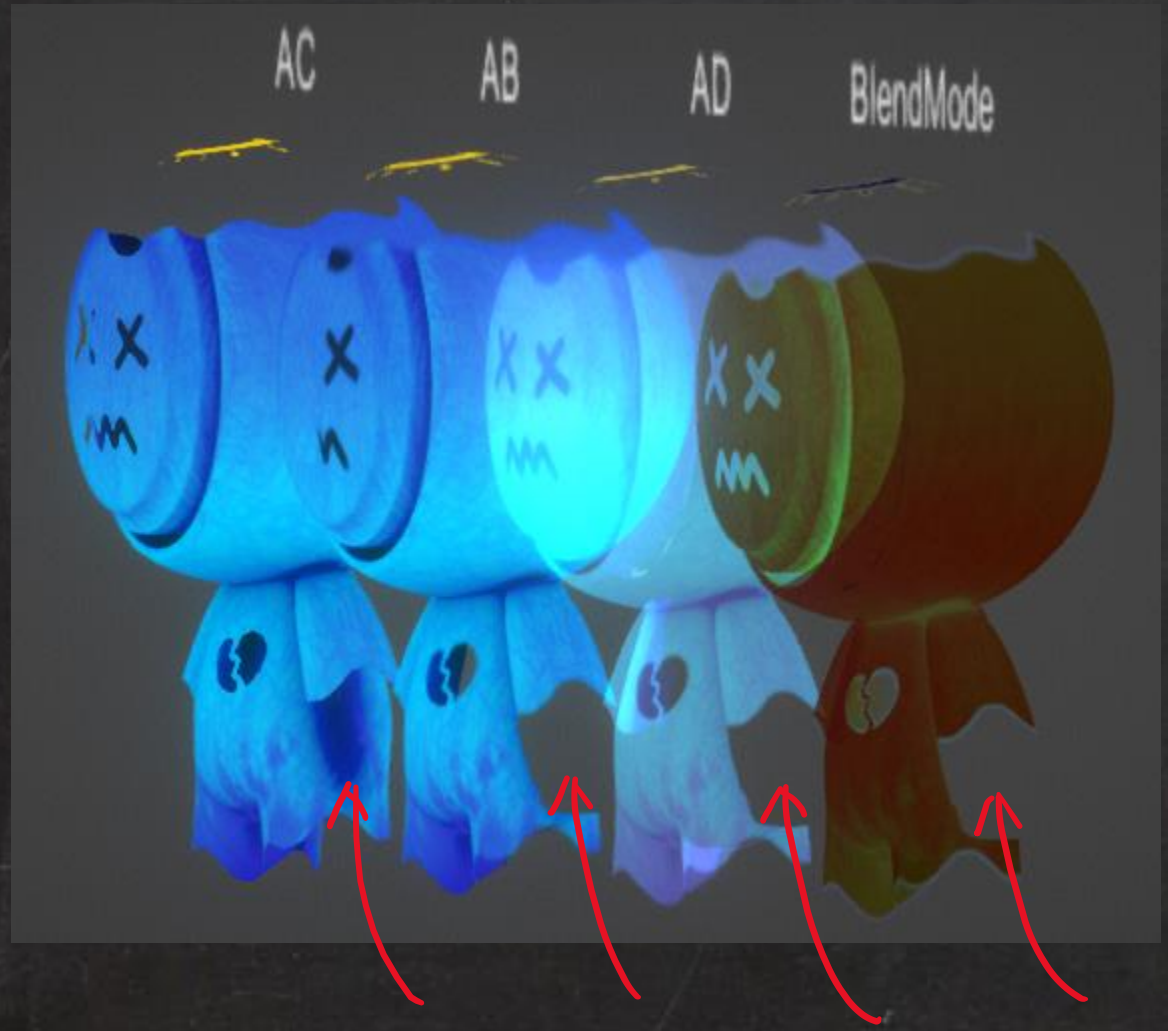
- 符文·增广·L13
- 符文·筑基·UV流动
- 符文·化神·UV扰动
- 任务委托

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

1 符文·增广·L13

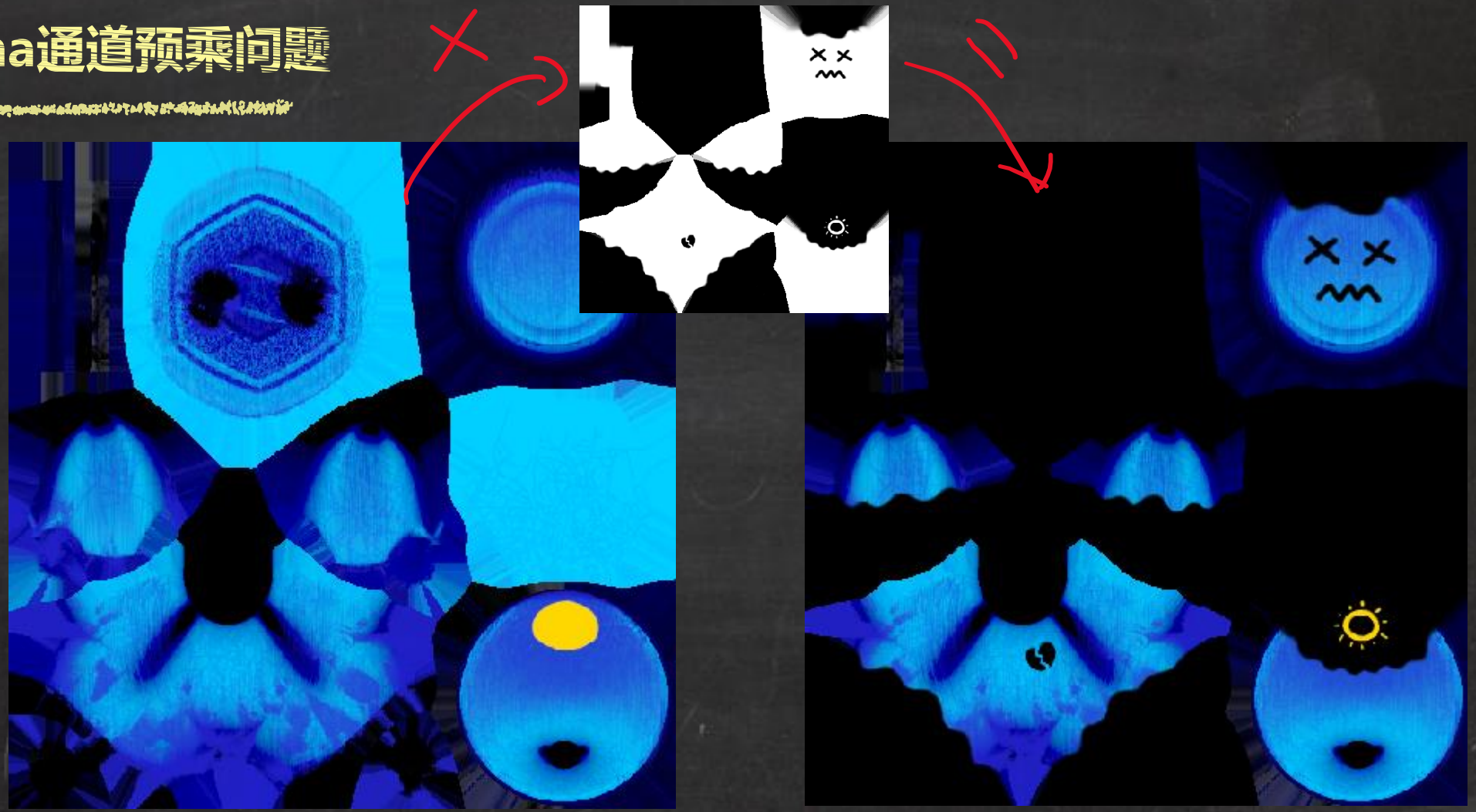


01 排序问题



1. 如图，AC没有排序问题；AB，AD有明显排序问题；
2. 简单的解决办法：
    1. Detach，Attach大法；
    2. 一些情况下可用Zwrite Off解决；

02 Alpha通道预乘问题



1. 左图没有预乘，右图做了预乘；
2. AB中预乘可用One OneMinusSrcAlpha；不预乘要用SrcAlpha OneMinusSrcAlpha，或在Shader里做乘法；
3. AD中预乘可以没有A通道；不预乘需要在Shader里做乘法；
4. 具体视项目情况而定；

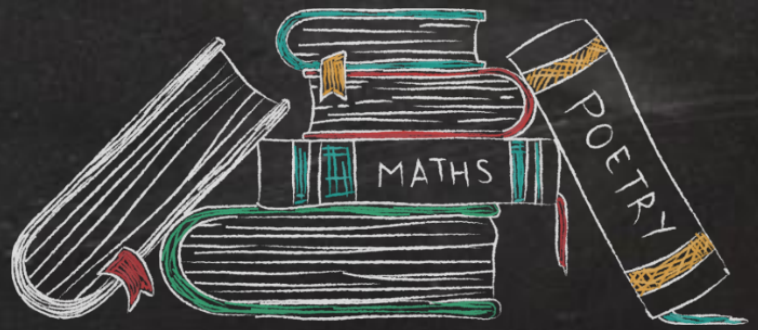
03 AB AD代码改进

1. 追加_Opacity参数，控制总体透明度；
2. 统一视纹理为不预乘，在Shader中做乘法；
3. 结果为AB，AD纹理形式统一；Shader代码内容统一；

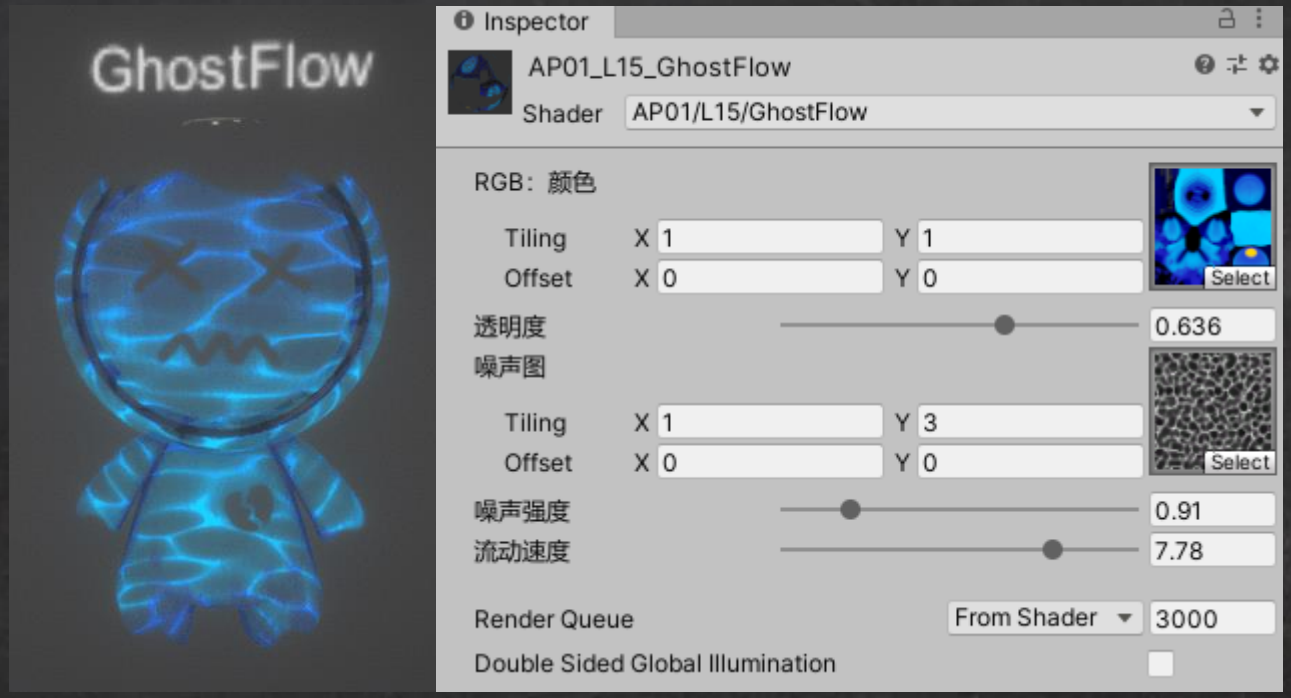
```
Shader "AP01/L13/AB" {
    Properties {
        _MainTex ("RGB：颜色 A：透贴", 2d) = "gray"{}
        _Opacity ("透明度", range(0, 1)) = 0.5
    }
    SubShader {
        Tags {
            "Queue"="Transparent"                // 调整渲染顺序
            "RenderType"="Transparent"           // 对应改为Cutout
            "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;
            uniform half _Opacity;
            // 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
                half3 finalRGB = var_MainTex.rgb;
                half opacity = var_MainTex.a * _Opacity;
                return half4(finalRGB * opacity, opacity);                  // 返回值
            }
            ENDCG
        }
    }
}
```

```
Shader "AP01/L13/AD" {
    Properties {
        _MainTex ("RGB：颜色 A：透贴", 2d) = "gray"{}
        _Opacity ("透明度", range(0, 1)) = 0.5
    }
    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;
            uniform half _Opacity;
            // 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>C
                    o.uv = TRANSFORM_TEX(v.uv, _MainTex);        // UV信息 支持Ti
                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
                half3 finalRGB = var_MainTex.rgb;
                half opacity = var_MainTex.a * _Opacity;
                return half4(finalRGB * opacity, opacity);                  // 返
            }
            ENDCG
        }
    }
}
```

2 符文·筑基·UV流动



01 演示·GhostFlow



02 代码·GhostFlow

1. 以L13_AB为模板，CtrlCV大法；
2. 修改路径名；
3. 定义材质面板：
    1. _MainTex：RGB：颜色 A：透贴
    2. _Opacity：总体不透明度
    3. _NoiseTex：混合用噪声图
    4. _NoiseInt：混合强度
    5. _FlowSpeed：噪声图流动速度
4. SubShaderTags，不用改；
5. 混合模式：Blend SrcAlpha OneMinusSrcAlpha；
6. 对应声明输入参数；

```
Shader "AP01/L15/GhostFlow" {
    Properties {
        _MainTex ("RGB：颜色 A：透贴", 2d) = "gray"{}
        _Opacity ("透明度", range(0, 1)) = 0.5
        _NoiseTex ("噪声图", 2d) = "gray"{}
        _NoiseInt ("噪声强度", range(0, 5)) = 0.5
        _FlowSpeed ("流动速度", range(0, 10)) = 5
    }
    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

            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 half _Opacity;
            uniform sampler2D _NoiseTex;    uniform float4 _NoiseTex_ST;
            uniform half _NoiseInt;
            uniform half _FlowSpeed;
```

03 代码·GhostFlow

7. 输入结构，不用改；

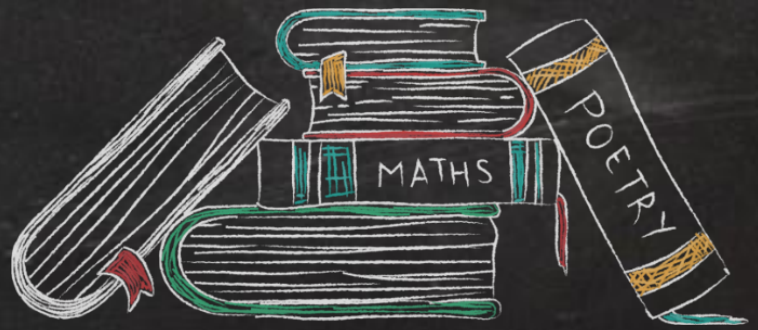
8. 输出结构声明有两套UV，uv0采样MainTex用，uv1采样NoiseTex用；

9. 顶点Shader对应追加对两套uv的赋值：
    1. uv0：传递即可；
    2. uv1：
        - TRANSFORM_TEX让其支持Tiling；
        - 对v轴加上随时间变化的偏移量，以实现流动；

10. 像素Shader：
    1. uv0，uv1分别对MainTex，NoiseTex采样；
    2. 获得finalRGB；
    3. 计算opacity：
        1. Remap噪声（0~1 > 0~2），以1为中间值；
        2. 截去噪声负值；
        3. 计算opacity = 透贴*总不透明度*噪声；
    4. 返回值：half4(finalRGB, opacity)；

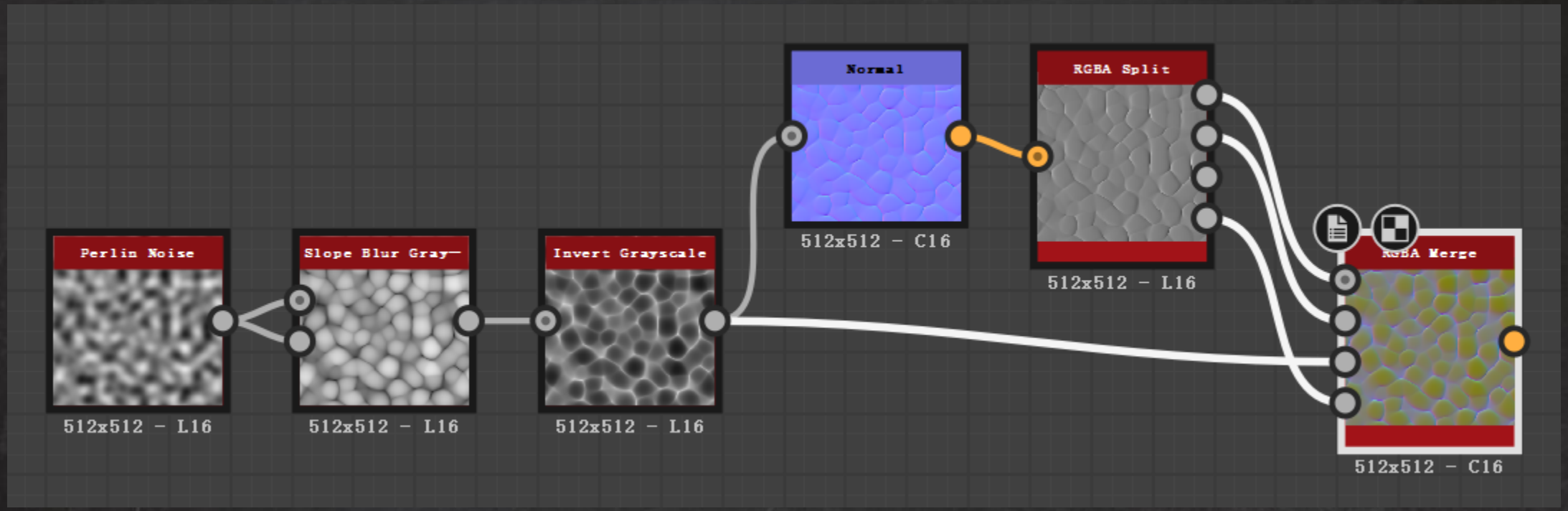
```
// 输入结构
struct VertexInput {
    float4 vertex : POSITION;       // 顶点位置 总是必要
    float2 uv : TEXCOORD0;          // UV信息 采样贴图用
};
// 输出结构
struct VertexOutput {
    float4 pos : SV_POSITION;       // 顶点位置 总是必要
    float2 uv0 : TEXCOORD0;         // UV 采样MainTex用
    float2 uv1 : TEXCOORD1;         // UV 采样NoiseTex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.uv0 = v.uv;
        o.uv1 = TRANSFORM_TEX(v.uv, _NoiseTex);         // UV1支持TilingOffset
        o.uv1.y = o.uv1.y + frac(-_Time.x * _FlowSpeed);// UV1 V轴流动
    return o;
}
// 输出结构>>>像素
half4 frag(VertexOutput i) : COLOR {
    half4 var_MainTex = tex2D(_MainTex, i.uv0);         // RGB颜色 A透贴
    half var_NoiseTex = tex2D(_NoiseTex, i.uv1).r;      // 噪声图

    half3 finalRGB = var_MainTex.rgb;
    half noise = lerp(1.0, var_NoiseTex * 2.0, _NoiseInt);  // Remap噪声
    noise = max(0.0, noise);                            // 截去负值
    half opacity = var_MainTex.a * _Opacity * noise;

    return half4(finalRGB * opacity, opacity);          // 返回值
}
ENDCG
    }
  }
}
```

3 符文·化神·UV扰动



01 演示·GhostWarp

02 准备WarpTex



1. 制作理想的Noise；
2. 以Noise为Height生成一张Normal；
3. 保留Normal的RG通道，将B通道替换为Noise，保存输出为WarpTex；

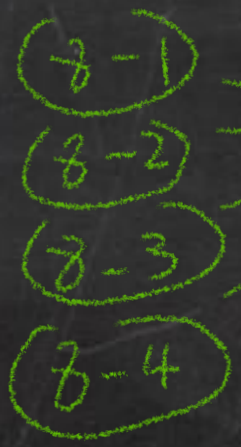
03 代码·GhostWarp

1. 以GhostFlow为模板，CtrlCV大法；
2. 修改路径名；
3. 定义材质面板：
    1. _MainTex：RGB：颜色 A：透贴
    2. _Opacity：总体不透明度
    3. _WarpTex：RG：扭曲图 B：噪声图
    4. _WarpInt：扭曲强度
    5. _NoiseInt：噪声混合强度
    6. _FlowSpeed：噪声图流动速度
4. SubShaderTags，不用改；
5. 混合模式，不用改；
6. 对应声明输入参数；

```
Shader "AP01/L15/GhostWarp" {
    Properties {
        _MainTex ("RGB: 颜色 A: 透贴", 2d) = "gray"{}
        _Opacity ("透明度", range(0, 1)) = 0.5
        _WarpTex ("扭曲图", 2d) = "gray"{}
        _WarpInt ("扭曲强度", range(0, 1)) = 0.5
        _NoiseInt ("噪声强度", range(0, 5)) = 0.5
        _FlowSpeed ("流动速度", range(0, 10)) = 5
    }
    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

            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 half _Opacity;
            uniform sampler2D _WarpTex;    uniform float4 _WarpTex_ST;
            uniform half _WarpInt;
            uniform half _NoiseInt;
            uniform half _FlowSpeed;
```

04 代码·GhostWarp

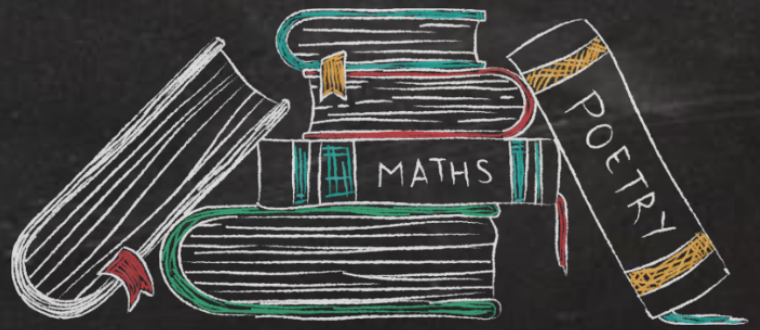
7. 输入，输出结构，顶点Shader；
8. 像素Shader：
    1. 先采样WarpTex；
    2. 计算uv偏移值：将WarpTex的RG通道值Remap到(-0.5, 0.5)；
    3. 将uv偏移值加给uv，得到偏移后的uv；
    4. 用偏移后的uv采样MainTex；
    5. 后续操作同GhostFlow；



```
// 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 uv0 : TEXCOORD0;          // UV 采样MainTex用
    float2 uv1 : TEXCOORD1;          // UV 采样NoiseTex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.uv0 = v.uv;
        o.uv1 = TRANSFORM_TEX(v.uv, _WarpTex);           // UV1支持TilingOffset
        o.uv1.y = o.uv1.y + frac(-_Time.x * _FlowSpeed);// UV1 V轴流动
    return o;
}
// 输出结构>>>像素
half4 frag(VertexOutput i) : COLOR {
    half3 var_WarpTex = tex2D(_WarpTex, i.uv1).rgb;      // 噪声图
    float2 uvBias = (var_WarpTex - 0.5) * _WarpInt;      // 计算UV偏移值
    float2 uv0 = i.uv0 + uvBias;                         // 应用UV偏移量
    half4 var_MainTex = tex2D(_MainTex, uv0);            // 偏移后UV采样MainTex

    half3 finalRGB = var_MainTex.rgb;
    half noise = lerp(1.0, var_WarpTex.b * 2.0, _NoiseInt); // Remap噪声
    noise = max(0.0, noise);                             // 截去负值
    half opacity = var_MainTex.a * _Opacity * noise;

    return half4(finalRGB * opacity, opacity);           // 返回值
}
ENDCG
            }
        }
    }
```

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

4 任务委托



01 作业

必做作业：

- 代码：GhostWarp，5月1日截止；

创意作业：

- 课内知识自由发挥；