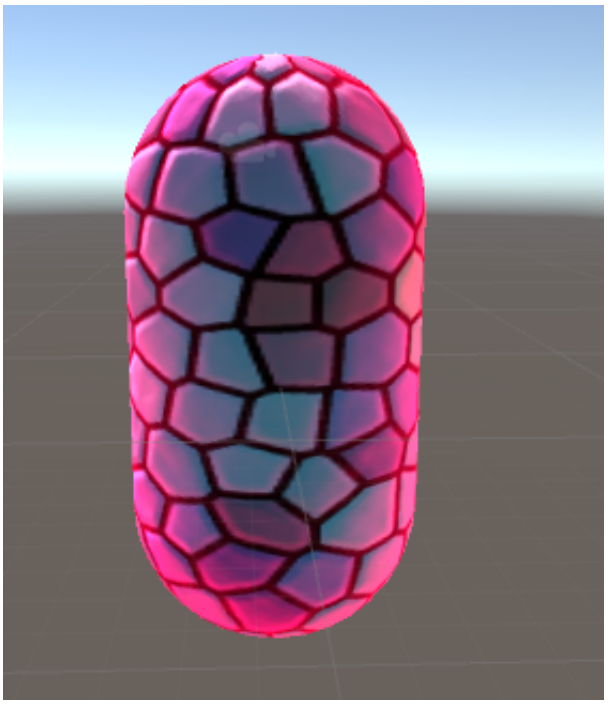
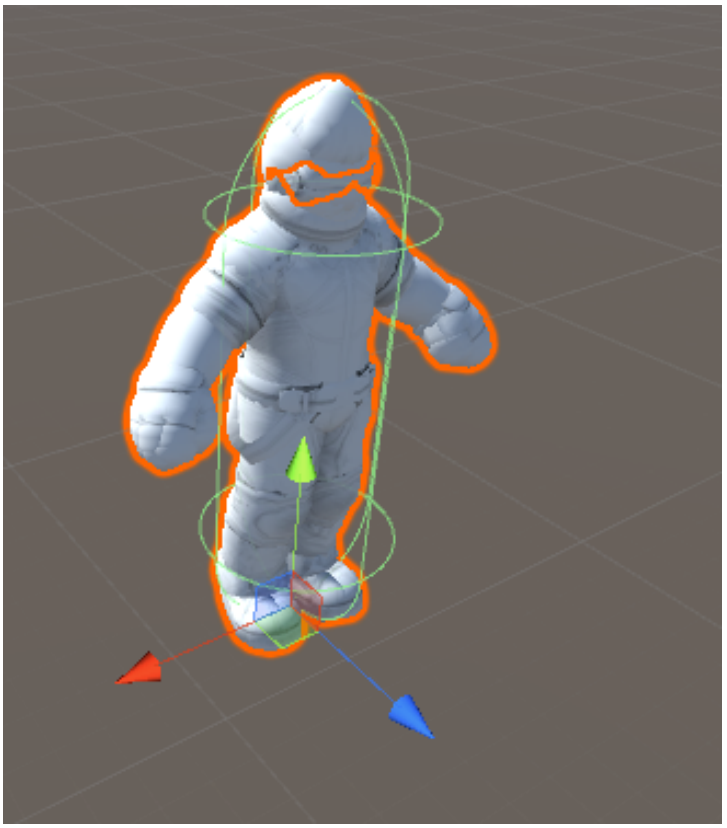
漫反射+凹凸+边缘光



```
Shader "Custom/NewSurfaceShader" {
    Properties {
        _MainTex ("Albedo (RGB)", 2D) = "white" {}
        _BumpMap("Normal Map",2D)="bump"{}
        _Color("_Color",Color)=(1.0,1.0,1.0,1.0)
        _ColorTint("色泽",Color)=(0.5,0.6,0.5,1)
        _RimPower("边缘颜色强度",Range(0.5,10))=3.0
    }
    SubShader {
        Tags { "RenderType"="Opaque" }
        LOD 200

        CGPROGRAM
        #pragma surface surf Lambert

        sampler2D _MainTex;
        sampler2D _BumpMap;
        fixed4 _Color;
        fixed4 _ColorTint;
        float _RimPower;

        struct Input {
            float2 uv_MainTex;
```

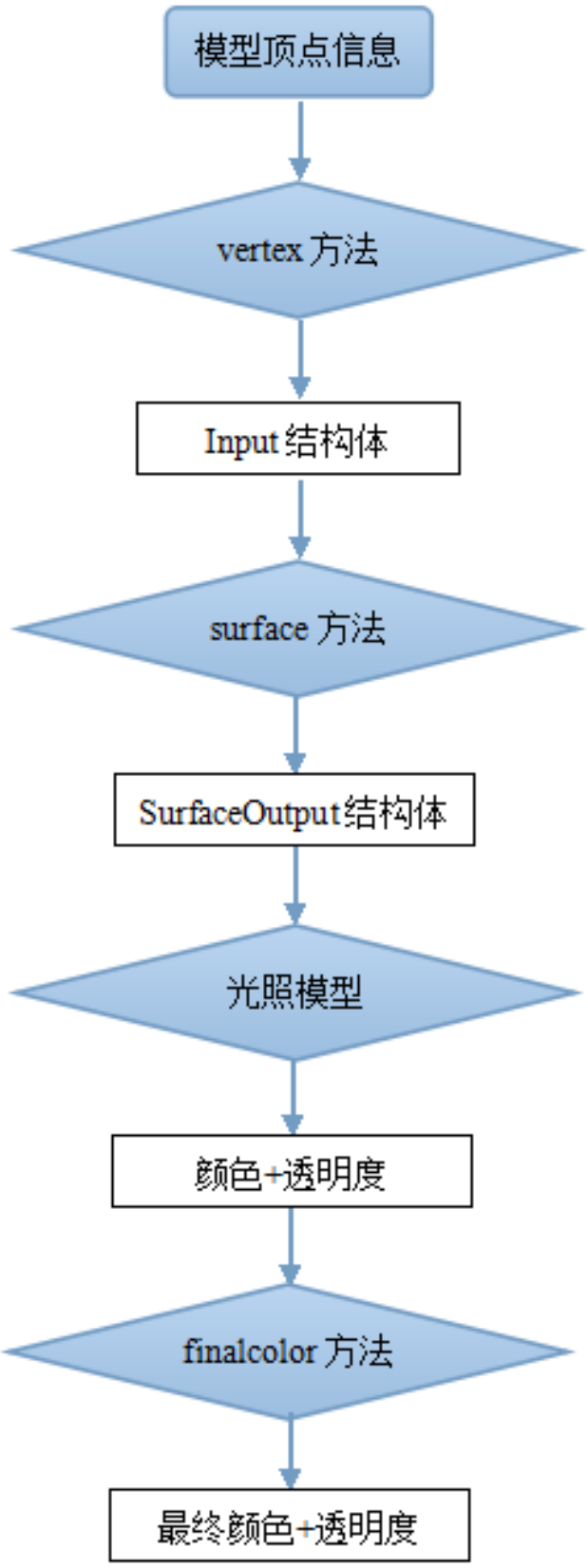
```
            float2 uv_BumpMap;
            float3 viewDir;//当前坐标的视线方向
        };

        void surf (Input IN, inout SurfaceOutput o) {

            fixed4 c = tex2D (_MainTex, IN.uv_MainTex);
            o.Albedo = c.rgb;
            //o.Alpha = c.a;
            o.Normal=UnpackNormal(tex2D(_BumpMap,IN.uv_BumpMap));

            //获得发光的强度系数
            //获取视角方向向量和法线向量的点积 求点积就是求夹角的cos值
            //normalize 获取单位化向量    dot 获得点积    saturate 限制0-1之间
            half rim=1.0-saturate(dot(normalize(IN.viewDir),o.Normal));
            //把发光系数赋值给Emission属性 当rim值越大 发光效果越强  pow求rim的_RimPower次幂
            o.Emission=_ColorTint*pow(rim,_RimPower);
        }
        ENDCG
    }
    FallBack "Diffuse"
}
```

利用顶点函数 改变模型定点位置信息 修改人物胖瘦效果

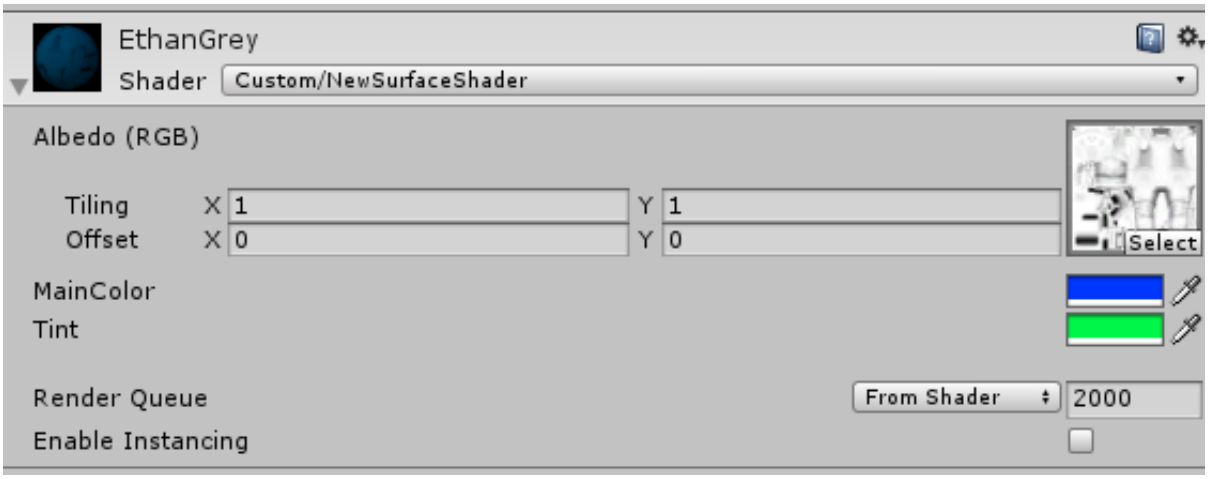


```
Shader "Custom/NewSurfaceShader" {
    Properties {
        _MainTex ("Albedo (RGB)", 2D) = "white" {}
        _Amount("Amount",range(-1,1))=0
    }
    SubShader {
        Tags { "RenderType"="Opaque" }
        LOD 200

        CGPROGRAM
        #pragma surface surf Lambert vertex:vert

        sampler2D _MainTex;
        float _Amount;

        struct Input {
            float2 uv_MainTex;
        };

        void vert(inout appdata_full v)
        {
            v.vertex.xyz+=v.normal*_Amount;
        }
```

```
24
25          void surf (Input IN, inout SurfaceOutput o) {
26
27              fixed4 c = tex2D (_MainTex, IN.uv_MainTex);
28              o.Albedo = c.rgb;
29
30          }
31          ENDCG
32      }
33      FallBack "Diffuse"
34  }
35
```



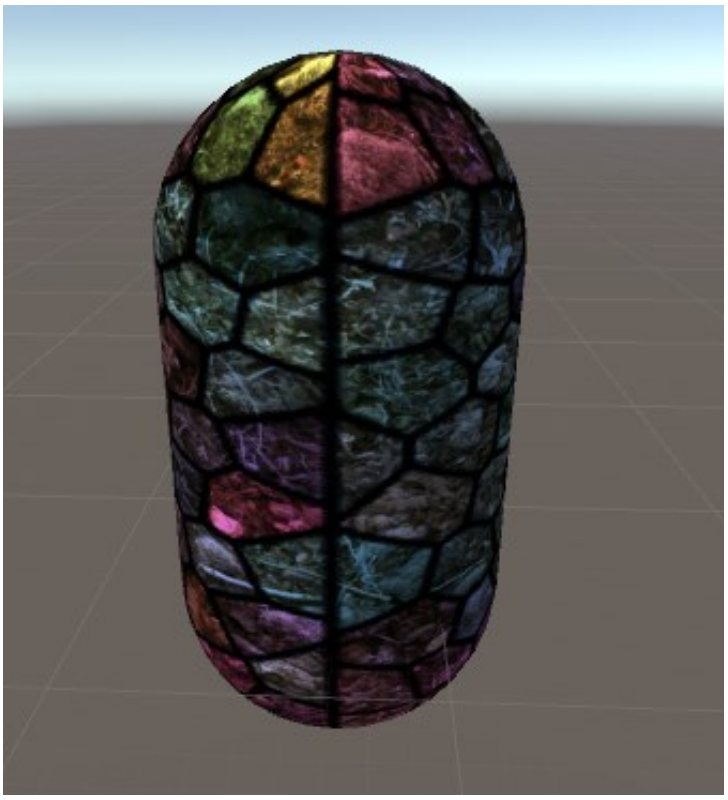
使用最终颜色函数实现颜色的混合




```
Shader "Custom/NewSurfaceShader" {
    Properties {
        _MainTex ("Albedo (RGB)", 2D) = "white" {}
        _MainColor("MainColor",color)=(1.0,1.0,1.0,1.0)
        _ColorTint("Tint",color)=(1.0,0.6,0.6,1.0)
    }
    SubShader {
        Tags { "RenderType"="Opaque" }
        LOD 200

        CGPROGRAM

        #pragma surface surf Lambert finalcolor:mycolor


        sampler2D _MainTex;
        fixed4 _ColorTint;
        fixed4 _MainColor;


        struct Input {
            float2 uv_MainTex;
        };

        void mycolor(Input IN, SurfaceOutput o,inout fixed4 _color)
        {
            //最终的颜色混合
            _color*=_ColorTint;
        }

```

```
31          void surf (Input IN, inout SurfaceOutput o) {
32
33              fixed4 c = tex2D (_MainTex, IN.uv_MainTex)*_MainColor;
34              o.Albedo = c.rgb;
35
36          }
37
38
39          ENDCG
40      }
41      FallBack "Diffuse"
42  }
43
```

使用两张贴图纹理实现颜色叠加 细节纹理效果



```
1  Shader "Custom/NewSurfaceShader" {
2      Properties {
3          _MainTex ("Albedo (RGB)", 2D) = "white" {}
4          _Detail("细节纹理",2D)="gray"{}
5      }
```

```
    SubShader {
        Tags { "RenderType"="Opaque" }
        LOD 200

        CGPROGRAM

        #pragma surface surf Lambert

        sampler2D _MainTex;
        sampler2D _Detail;

        struct Input {
            float2 uv_MainTex;
            float2 uv_Detail;
        };

        void surf (Input IN, inout SurfaceOutput o) {
            o.Albedo= tex2D (_MainTex, IN.uv_MainTex).rgb;
            o.Albedo*=tex2D(_Detail,IN.uv_Detail).rgb*2;
        }

        ENDCG
    }
    FallBack "Diffuse"
}

```

积雪效果

```
Shader "Custom/NewSurfaceShader" {
    Properties {
        _MainTex ("Albedo (RGB)", 2D) = "white" {}
        _Bump("Normal",2D)="bump"{}
        _Snow("Snow level",range(0,0.5))=0
        _Color("MainColor",color)=(1.0,1.0,1.0,1.0)
        _SnowColor("SnowColor",color)=(1.0,1.0,1.0,1.0)

        //雪的方向是垂直落下的反方向
        //理论上上从天空落下的向量是0,-1,0    此处为了方便和法线运算 用的是0,1,0
        _SnowDirection("SnowDirection",Vector)=(0,1,0)

    }
    SubShader {
        Tags { "RenderType"="Opaque" }
        LOD 200

        CGPROGRAM

        #pragma surface surf Lambert


        sampler2D _MainTex;
        sampler2D _Bump;
```

```
25          float4 _Color;
26          float _Snow;
27          float4 _SnowColor;
28          float4 _SnowDirection;
29
30          struct Input {
31              float2 uv_MainTex;
32              float2 uv_Bump;
33              float3 worldNormal;INTERNAL_DATA
34          };
35
36
37          void surf (Input IN, inout SurfaceOutput o) {
38              half4 c= tex2D (_MainTex, IN.uv_MainTex);
39              o.Normal=UnpackNormal(tex2D(_Bump,IN.uv_Bump));
40
41              //得到基于世界坐标系下的真正的法向量(并非凹凸贴图产生的法向量 而是要做一个空间到世界的转换)
42              //用法向量和下雪的方向进行点积运算 得到是否满足积雪效果
43
if(dot(WorldNormalVector(IN,o.Normal),_SnowDirection.xyz)>lerp(1,-1,_Snow))
44                  o.Albedo=_SnowColor.rgb;
45              else
46                  o.Albedo=c.rgb*_Color;
47              o.Alpha=c.a;
48          }
49          ENDCG
50      }
51      FallBack "Diffuse"
52  }
```