inline float4 **ComputeNonStereoScreenPos**(float4 pos) {

float4 o = pos \* 0.5f;

#if defined(UNITY\_HALF\_TEXEL\_OFFSET)

o.xy = float2(o.x, o.y\*\_ProjectionParams.x) + o.w \* \_ScreenParams.zw;

#else

o.xy = float2(o.x, o.y\*\_ProjectionParams.x) + o.w;

#endif

o.zw = pos.zw;

return o;

}

inline float4 **ComputeScreenPos** (float4 pos) {

float4 o = ComputeNonStereoScreenPos(pos);

#ifdef UNITY\_SINGLE\_PASS\_STEREO

o.xy = TransformStereoScreenSpaceTex(o.xy, pos.w);

#endif

return o;

}

其中UNITY\_SINGLE\_PASS\_STEREO用于控制在不同平台上计算方式的差异，而UNITY\_HALF\_TEXEL\_OFFSET用于支持DirectX 9的像素偏移。该函数返回的是齐次坐标下的屏幕坐标值

剔除不必要的代码，核心代码为：

inline float4 ComputeScreenPos (float4 pos)

{

float4 o = pos \* 0.5f;

o.xy = float2(o.x, o.y\*\_ProjectionParams.x) + o.w;

o.zw = pos.zw; return o;

}

首先该函数传入的参数pos为顶点变换到齐次坐标系下的坐标，也就是说我们在调用该函数的时候需要先将模型空间下的顶点坐标转换为齐次坐标系下的顶点坐标

我们先看正常流程的屏幕坐标是如何计算的

ScreenX = ((x / w) \* 0.5 + 0.5) \* width

ScreenY = ((y / w) \* 0.5 + 0.5) \* height

我们再看看ComputeScreenPos是如何做的，它的计算方法为

o.x = (pos.x \* 0.5 + pos.w \* 0.5)

o.y = (pos.y \* 0.5 \* \_ProjectionParams.x + pos.w \* 0.5)

其中\_ProjectionParams.x用于在使用翻转投影矩阵时（此时其值为-1.0）翻转y的坐标值，现在我们将正常屏幕坐标计算通过乘以w

W \* screenX = (x \* 0.5 + w \* 0.5) \* width

W \* screenY = (y \* 0.5 + w \* 0.5) \* height

现在一目了然了，ComputeScreenPos返回的值是齐次坐标系下的屏幕坐标值，其范围为[0,w]。

那么unity为什么要这么做呢？

这是因为如果在顶点着色器中除的话，会破坏插值结果，从顶点着色器到偏远着色器会有一个插值过程，如果我们队x/w、y/w进行插值，结果会不准确，因为投影空间不是线性空间，插值往往是线性的，所以不要在投影空间进行插值

Unity本意是希望你把该坐标值用作tex2Dproj指令的参数值，tex2Dproj会在对纹理采样前除以w分量。当然你也可以自己除以w分量后进行采样，但是效率不如内置指令tex2Dproj

pos = UnityObjectToClipPos(v.vertex);

screenPos = ComputeScreenPos(o.pos);

tex2D(\_ScreenTex, float2(screenPos.xy / screenPos.w))