# **手机端次时代海水**

作者：日音

链接：<http://gad.qq.com/article/detail/253634>

先上最终效果：<http://gad.qq.com/article/detail/253634>

随着海岛奇兵海面的实现到上一个移动海面的改进，收到了不错的反响，这促使我开始思考，有没有可能在牺牲部分性能的情况下实现更加贴近pc端表现的海水。

话不多说，开始执行。

首先，我们要知道目光射入海水的方向：

float3 worldView = (IN.worldPos - \_WorldSpaceCameraPos);

然后要获得海水的法线：

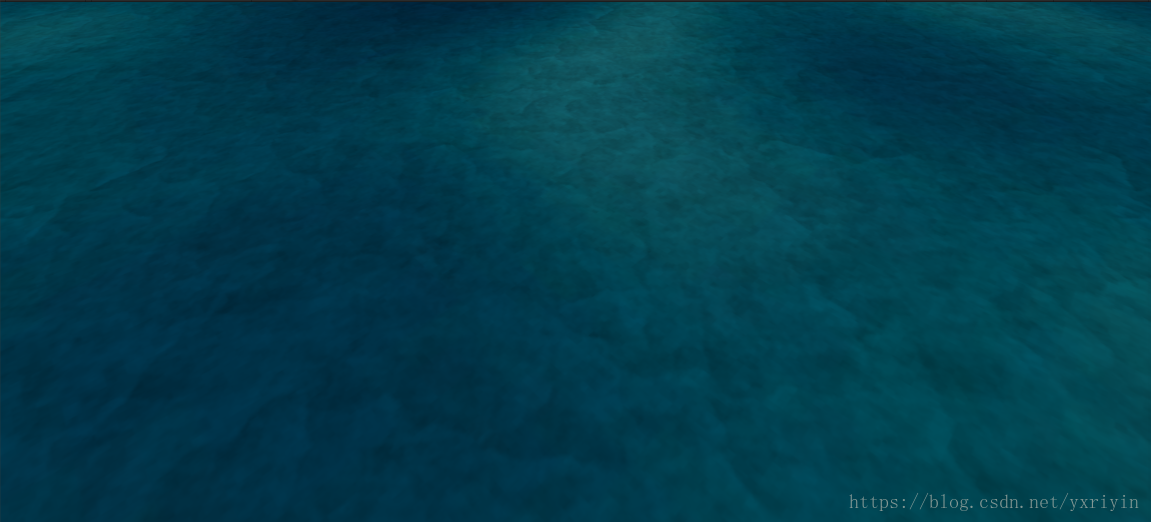
half4 nmap = (tex2D(\_BumpTex, i.worldPos.xz + offset) + tex2D(\_BumpTex, half2(-i.worldPos.z, i.worldPos.x) - offset)) \* 0.5;

这里为了让海水有流动的效果，offset随着时间变化，而取法线贴图则是用了xz和z来出里扰动。

有了法线和眼睛视角，就可以根据菲涅尔公式计算点积：

half fresnel = sqrt(1.0 - dot(-normalize(worldView), worldNormal));

这样就有了初步效果：



然后就是海面的高光，我们打算模拟下光照，如下：

half reflectiveFactor = max(0.0, dot(viewdir, reflect(lightDir, worldNormal)));

half shininess = \_Strength \* 100;

half specularFactor = pow(reflectiveFactor, shininess);

half diffuseFactor = max(0.0, dot(worldNormal, lightDir));

color = tex2D(\_MainTex, i.texcoord);

color.rgb \*= diffuseFactor;

color.rgb = \_Specular.rgb \* specularFactor;

这里我整理下光照的部分，首先我们理解下光照的方向，可以认为应该是负的xyz，这样才像从上面照下来。

于是我把光照设置成-1,-1,0,而计算物体光照原色的时候，利用法线和光照方向的点积，这里要注意，你需要反向一下光照的方向。而计算高光的时候，则是真的用光线的方向了。

这里还有一个非常有意思的点，就是法线里面存的是物体本身的坐标系，我们法线大部分的法线的值都是Z轴朝向的，但水面肯定是朝y轴的才对，所以这里用了一个小技巧，就是把y和z调换。

经过这样的处理，效果变成了这样：



接下来处理海岸的部分，先使用自带的深度图试试：

o.args = ComputeScreenPos(o.pos);

COMPUTE\_EYEDEPTH(o.args.z);

half4 foam = (tex2D(\_FoamTex, uv1) + tex2D(\_FoamTex, uv2)) \* 0.5;

half3 worldNormal = (normal.xyz \* 2 - 1).xzy;

color = tex2D(\_MainTex, i.texcoord);

float depth = tex2Dproj(\_CameraDepthTexture, i.args).r;

depth = LinearEyeDepth(depth);

depth = depth - i.args.z;

float depth1 = saturate(depth \* \_Range.x);

float depth2 = saturate(depth \* \_Range.x \* 0.9);

根据视角看过去的深度，计算出深度差。根据深度差，显示出海岸泡沫的效果。这里有一小技巧，我本来使用depth的时候，发现有泡沫的地方和没有泡沫的地方有很明显的分界线，于是用了两层泡沫，第一层浓，第二层淡，弱化泡沫边界。

效果如下：



接下来就是增加光照和深浅变化的控制，这部分就很简单了：

half3 lightDir = \_WorldSpaceLightPos0.xyz;

// Phong shading model

half reflectiveFactor = max(0.0, dot(viewdir, normalize(reflect(-lightDir, worldNormal)))) \* 0.995;

half shininess = \_Strength \* 200.0;

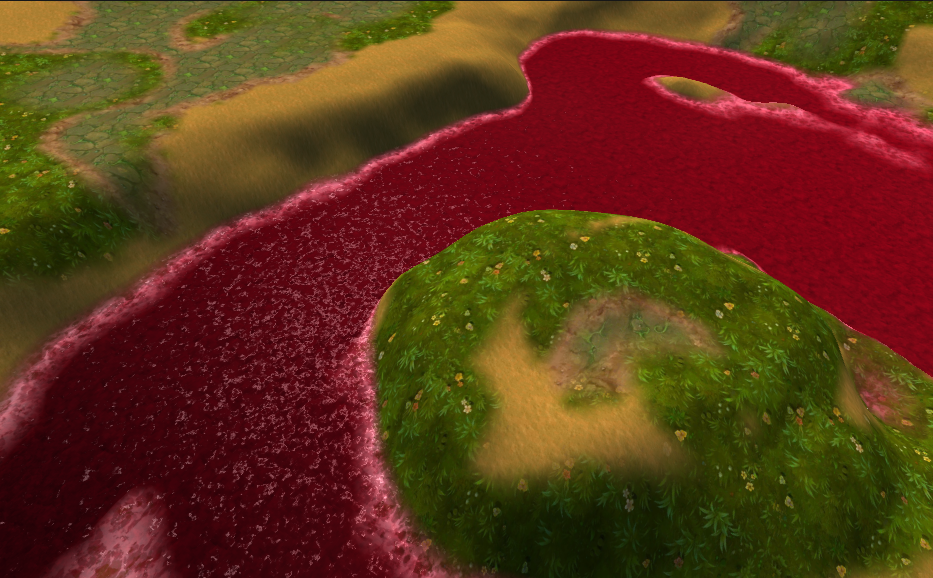
half specularFactor = pow(reflectiveFactor, shininess);

float deep = 1 - saturate(depth1 \* \_Range.y);

float4 deepColor = lerp(\_Bright, \_Dark, deep \* 2)\* \_Range.y;

color.rgb = color.rgb \* \_Range.z + deepColor \* (1 - \_Range.z);

配了一个血海：



目前已经提交给商店更新了，审核大概需要1-2周。

商店插件地址：[https://www.assetstore.unity3d.com/en/#!/content/120466](https://www.assetstore.unity3d.com/en/" \l "!/content/120466)