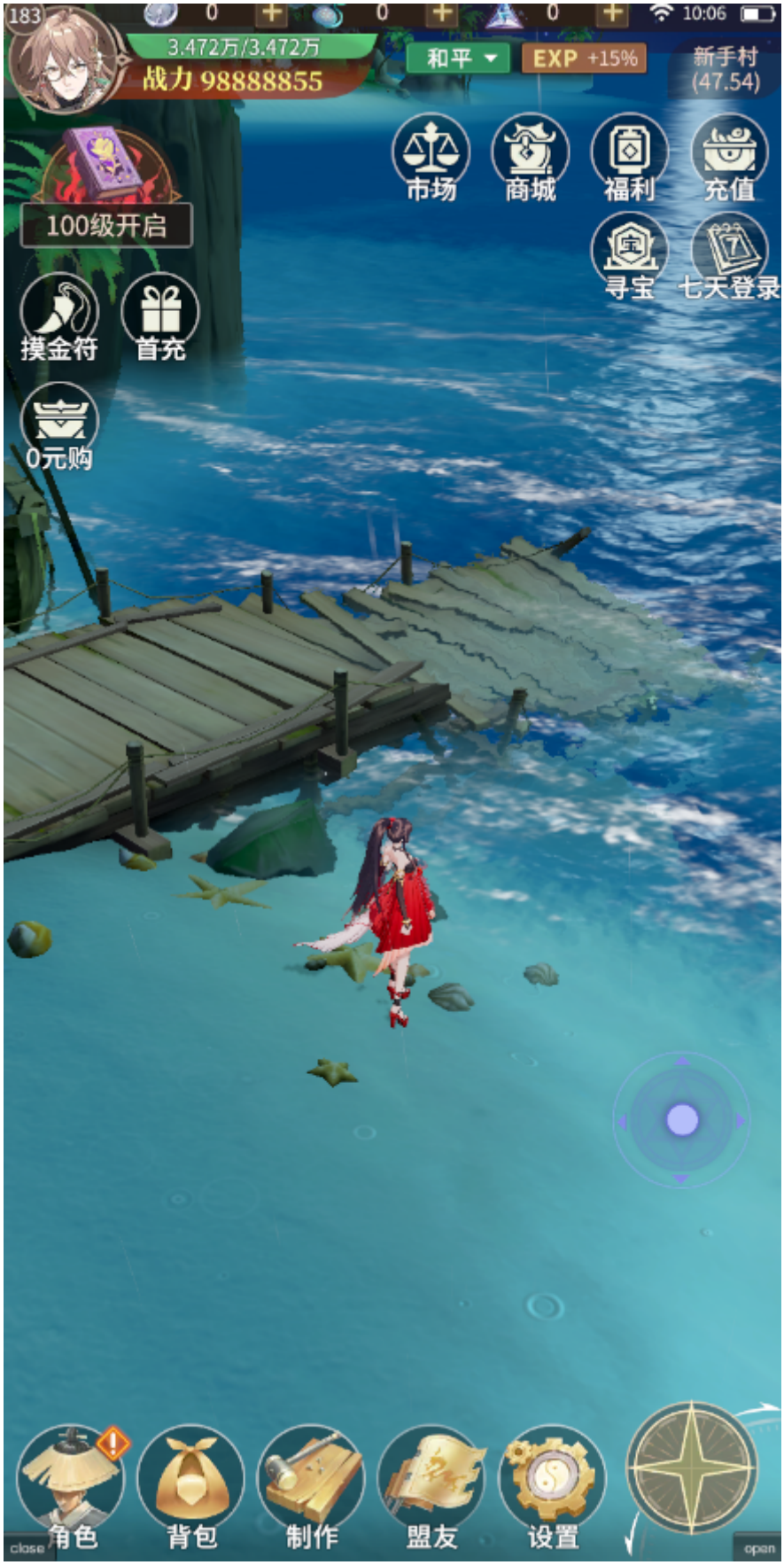
水面(WaterNew)

背景

原水面的水面法线计算有些trick做法，不知道原理是什么，作者也已离职，因此自己写了一版功能类似的水面。

效果展示



参数说明

1. 海水法线：使用逆FFT的海水法线或FBM生产的噪声生成法线。法线会从两个方向交叉流动。
2. 法线强度：法线强度会同时影响到反射的扭曲、折射的扭曲、光照计算
3. 风力方向：xyz分别代表世界坐标下的xyz方向。这里只使用x和z方向
4. 顶点波动（使用顶点波动需要高面数的平面）

    1. 开关
    2. 波动振幅：海面的起伏
    3. 波动速度：海面起伏的速冻
    4. 波动频率：海面起伏的密度
5. 伪光
    1. 开关
    2. 伪光方向：xyz代表xyz三个轴的欧拉角
    3. 伪光颜色
    4. 伪光强度
6. 实时反射
    1. 类型：
        1. cube(环境盒)：需要设置ReflectionProbe或Environment Reflections
        2. realtime（相机平面反射）：需要在水面挂上Planar Reflection脚本，设置反射layer，原则上只反射大件物体。避免高额的drawcall。
        3. Null：无反射
7. 水体颜色（随深浅变化）
    1. 浅水颜色
    2. 深水颜色
    3. 深度Scale：控制最浅最深的区间范围。值越小范围越大。
    4. 深度Power：深浅变化的指数参数。控制深浅变化。
    5. 深度Scale和Power请先置为1，然后调整scale，最后Power。
8. 边缘 （已屏蔽）
9. 漫反射强弱：海水漫反射或纯海水颜色的渐变
10. 高光
    1. 高光颜色
    2. 强度
    3. 光泽度
11. 模拟焦散
    1. 焦散显示距离：焦散距离水面的距离，浅或深都不会显示焦散。
    2. 焦散尺寸
    3. 焦散强度
12. 浪花：海岸边近处会看到浪花拍岸，远处则隐藏掉
    1. 开关
    2. 浪花贴图：
    3. 颜色
    4. 速度
    5. 扭曲
    6. 尺寸
13. 远处渐隐：远处的边缘会逐渐半透，用于水天一线
    1. 遮罩
    2. 强度

技术简介

1. 水面法线
    1. 使用两层交叉的水面法线混合得到，更复杂真实的水面需要用fbm的方式，将不同法线用不同振幅、频率混合。

```
// 取法线(考虑使用更复杂的计算使法线更自然)
wind = i.wind;
// 来点魔法数字，使法线随机些
half4 Rand = half4(0.2,0.5,.37,15);
// 两道uv交叉移动的法线叠加
float4 offset = 0.701 * float4(wind.x-
wind.y,wind.x+wind.y,wind.x+wind.y,-wind.x+wind.y);
float4 uvWind = uvNormal.xyxy -time * offset + Rand;

TN1 = UnpackNormalWithScale(tex2D(_NormalTex, uvWind.xy), _NormalScale);
TN2 = UnpackNormalWithScale(tex2D(_NormalTex, uvWind.zw * 0.4),
_NormalScale);
TN = lerp(TN1, TN2, 0.5);

worldNormal.x = dot(_unity_tbn_0, TN);
worldNormal.y = dot(_unity_tbn_1, TN);
worldNormal.z = dot(_unity_tbn_2, TN);
worldNormal = normalize(worldNormal);
```

2. 水面波动
    1. 简单的顶点正弦波

```
inline float4 WaterWave(in float4 vertex,in float time)
{
float4 ret = vertex;
float2 uv = vertex.xz * _WaveFrequency + time * _WaveSpeed;
ret.y += (sin(uv.x) + cos(uv.y)) * _WaveScale;
return ret;
}
```

3. 水体颜色
    1. 使用简单的双颜色，依据深度渐变

```
// 屏幕深度差 = 水底深度 - 水面深度
// 深度系数= pow( saturate ( 屏幕深度差/ 过渡深度), 过渡系数)
// 水体基础颜色= lerp( 浅水颜色, 深水颜色, 深度系数)
half t1 = saturate(pow(saturate(depthDiff * worldView.y * _DepthScale),
_DepthPower*1.5));
half alpha = lerp(_ShallowColor.a, _DeepColor.a, t1);
half t2 =  saturate(pow(saturate(0.65 * depthDiff * worldView.y *
_DepthScale), _DepthPower));
half4 baseColor = lerp(_ShallowColor, _DeepColor, t2);
```

    2. 如果需要更细腻的变化，应该使用ramp渐变。ramp渐变需要程序实现，可参考ShaderGraph的Gradient节点

4. 水面光照
    1. 特化的blinnphong光照

```
// 特化的漫反射 美术嫌弃原版漫反射不好看
baseColor.rgb *= saturate(dot(worldNormal,worldLight) *
_DiffuseIntensity + (1-_DiffuseIntensity)) ;
// 高光
col.rgb += pow(dotNH,_Gloss*128) * _SpecularPower * lightColor *
_SpecularColor * alpha;
```

5. 水面反射
    1. cube反射或相机平面反射

```
// 反射
inline half3 ReflectColor(float4 screenPos,half3 worldNormal, half3
worldView, half depthRatio){
half3 reflColor = 0;
// 平面反射
#if _REFLECTION_REALTIME
float4 reflPos = screenPos + float4( worldNormal.xz,0,0) * depthRatio ;
reflColor = tex2Dproj(_ReflectionTex, UNITY_PROJ_COORD(reflPos)).rgb;
#endif
// 静态cube反射
#if _REFLECTION_CUBE
half perceptualRoughness = 1 - _Gloss;
perceptualRoughness = perceptualRoughness*(1.7 -
0.7*perceptualRoughness);
half mip = perceptualRoughness * UNITY_SPECCUBE_LOD_STEPS;
half3 reflectionDir = reflect(-worldView,worldNormal);
reflectionDir.y = reflectionDir.y < 0? 0:reflectionDir.y;
half4 EnvReflData = UNITY_SAMPLE_TEXCUBE_LOD(unity_SpecCube0,
reflectionDir,mip);
reflColor = DecodeHDR (EnvReflData, unity_SpecCube0_HDR);
#endif
// 雷光
reflColor.rgb += _LightningColor;
return reflColor.rgb;
}
```

6. 模拟焦散
    1. [基于连续空间旋转min叠加，实现焦散]

```
// 基于连续空间旋转min叠加，实现焦散
[https://blog.csdn.net/tjw02241035621611/article/details/80135626]
float CausticRotateMin(float2 uv, float time){
float3x3 mat = float3x3(2,1,-2, 3,-2,1, 1,2,2);//1.2.操作矩阵
float3 vec1 = mul(mat*0.5,float3(uv,time));//3.对颜色空间进行操作
float3 vec2 = mul(mat*0.4,vec1);//4.重复2，3操作
float3 vec3 = mul(mat*0.3,vec2);
float val = min(length(frac(vec1)-0.5),length(frac(vec2)-0.5));//5.集合操
作 min
val = min(val,length(frac(vec3)-0.5));
val = pow(val,7.0)*25.;//6.亮度调整
return val;
}
```

```
// 焦散
inline half3 CausticColor(float2 uv, float time,half depth, half c,
half2 disortOffset){
half power = saturate(1-abs(c-depth)) * _CausticPower;
return CausticRotateMin(uv*_CausticScale + disortOffset, time).xxx *
power ;
}
```

7. 岸边浪花
    1. 梦幻西游3d的实现方式，浪花纹理加法线扰动

```
// 假的海浪
inline half3 FoamColor(float2 uv, half2 wind, half2 disortOffset,float
time){
    // uvNormal -= _Time.y * float2(1,0);
    uv += disortOffset.xy;
    float2 uvFoam = 0.701*float2(uv.x + uv.y, - uv.x + uv.y)  ;
    half windPower = length(wind.xy);
    float cosT = wind.x / windPower;
    float sinT = wind.y / windPower;
    uvFoam = float2(uvFoam.x * cosT+sinT*uvFoam.y,-sinT * uvFoam.x +
cosT*uvFoam.y);
    uvFoam -= time * float2(0.701,-0.701) * windPower * _FoamSpeed;
    // return tanT;
    // return half4(frac(uvFoam),0,1);
    half3 foamColor = tex2D(_WaterFoamTex, uvFoam * _FoamScale).rgb;
    foamColor *= foamColor;
    return foamColor * _FoamColor.rgb;
}
```

8. 远处渐隐
    1. 使用遮罩控制边缘半透，实现水天一线

```
// 远处渐隐
col.rgb = lerp(refract,col.rgb,saturate(tex2D(_FadeShape,i.uv.zw).r *
_FadeIntensity));
```

参考资料

1. 真实感水体渲染技术总结
2. JTRP风格化水体渲染