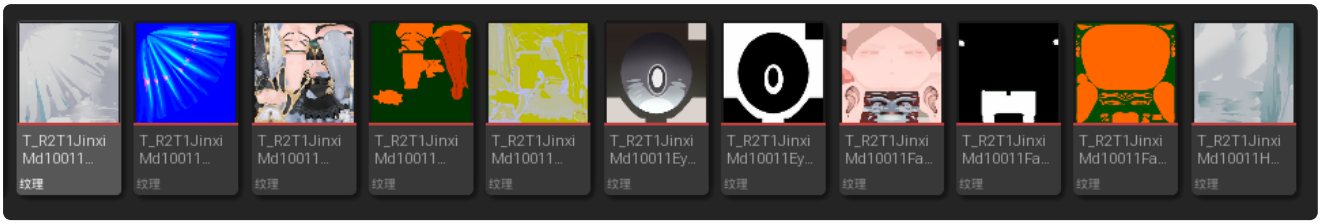
仿鸣潮人物渲染（思路和分析）

学习了半年人物卡渲，最近也是做了一个仿鸣潮人物渲染作为一个小小的目标暂时结束吧，也是我推今汐了，嘻嘻，主要是把学习的一些东西整合一下，这边也非常感谢各位知乎大神与B站大佬，当然，需要学习的还有太多太多了，不过在深入学下去的话就需要涉及更深入的内容了，需要修改UE代码和默认的材质管线，UE的终点是C++啊，这部分的洞很深，等未来的我慢慢来填吧，最近还是专注一下本行工作嘿嘿。

贴图资源分析阶段

在做渲染的时候，最重要的便是对贴图资源进行分析，因为要了解每一个贴图的作用，才能做出正确的渲染效果



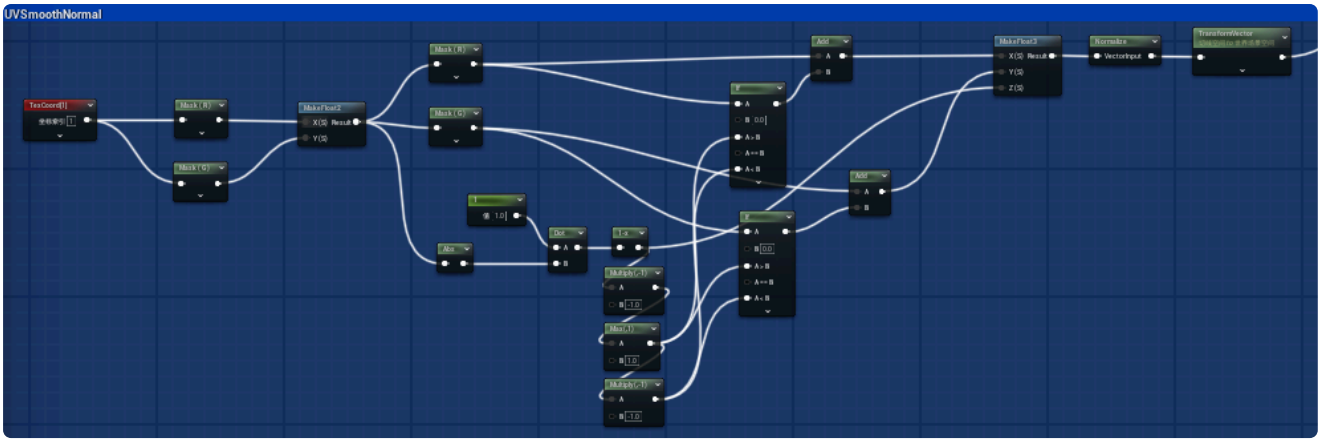


鸣潮模型主要分为上半，下半以及头发三个部分（还有一个eyes），头发又分了bangs和hair两个部分。鸣潮是没有把眉毛睫毛拆分出来的。
Diffuse贴图RGB存的就是漫反射的信息，在A通道是光照信息和常暗阴影的贴图。
Normal贴图RG通道存了法线信息B通道和A通道是Roughness和Metalic的信息。
ID贴图是区分区域信息的贴图。
HM贴图R通道是头发高光G通道是头发阴影光照信息。
还有一个skin的ramp图。

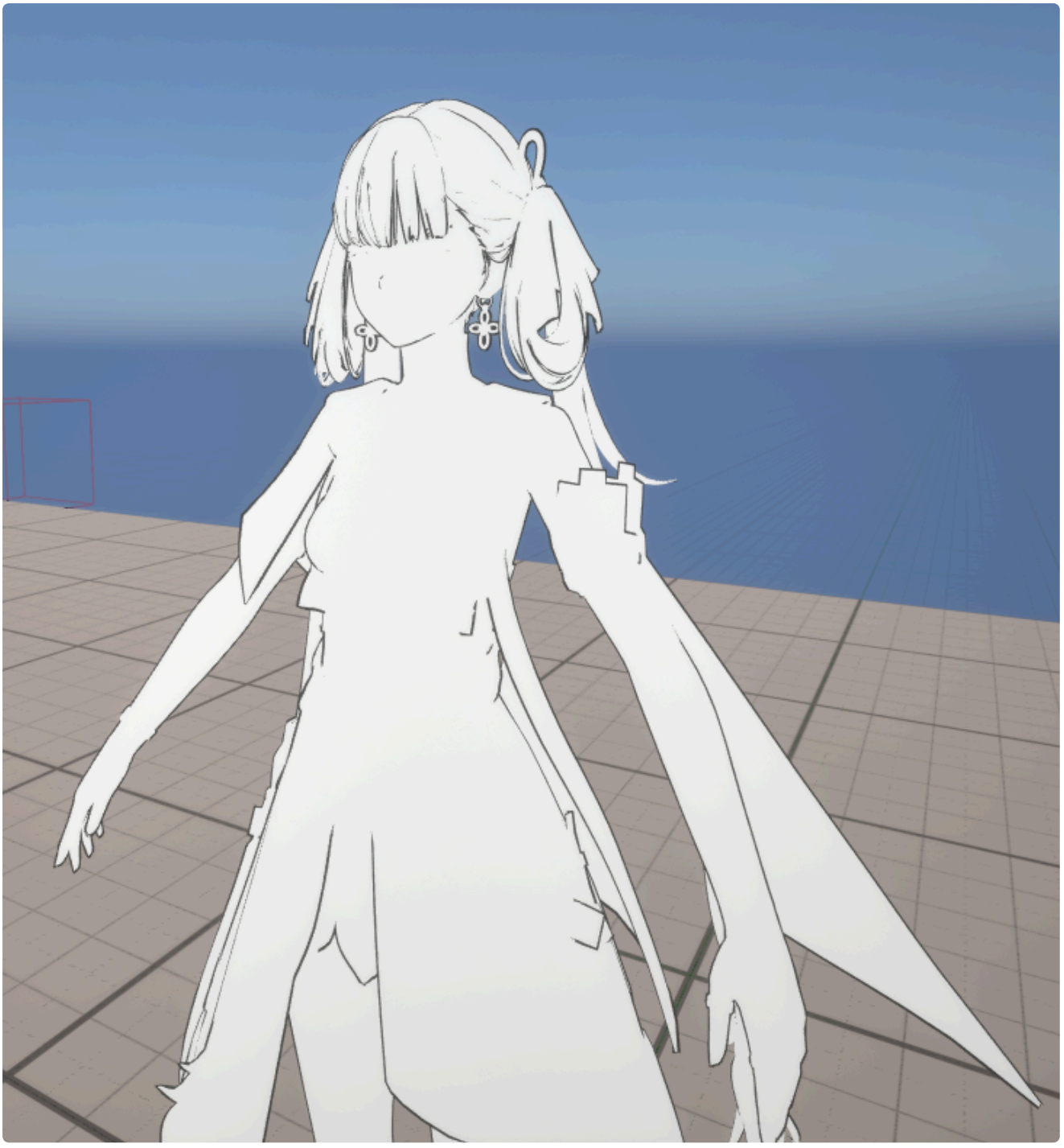
这边顺便说一下，鸣潮官方模屋给的mmd贴图和模型与游戏里的会有差别（至少今汐是这样，其他角色不知道），虽然不知道为什么要这么做，所以想完全复现游戏效果的话可能得自己想办法通过一些特殊方法获得资源。

描边

描边就是法线外扩即可，鸣潮的平滑法线存在了UV1里，鸣潮总共有4套UV，还有2套我也不知道是干什么用的（求大佬解答）

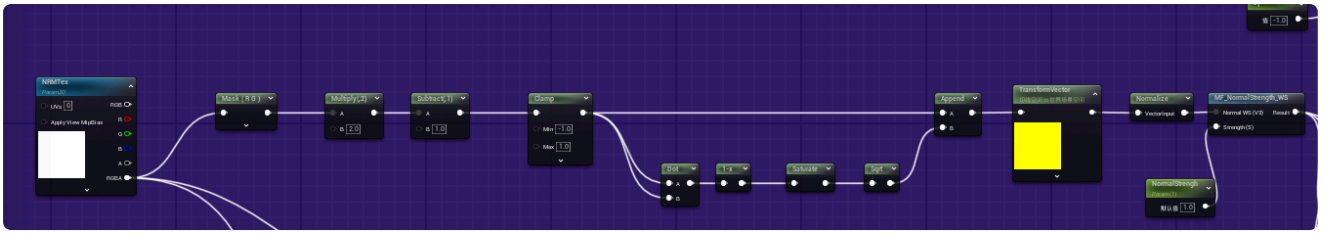


主要就是把uv里存的平滑法线二维转三维，上面图中的计算就是八面体法线解码。



法线

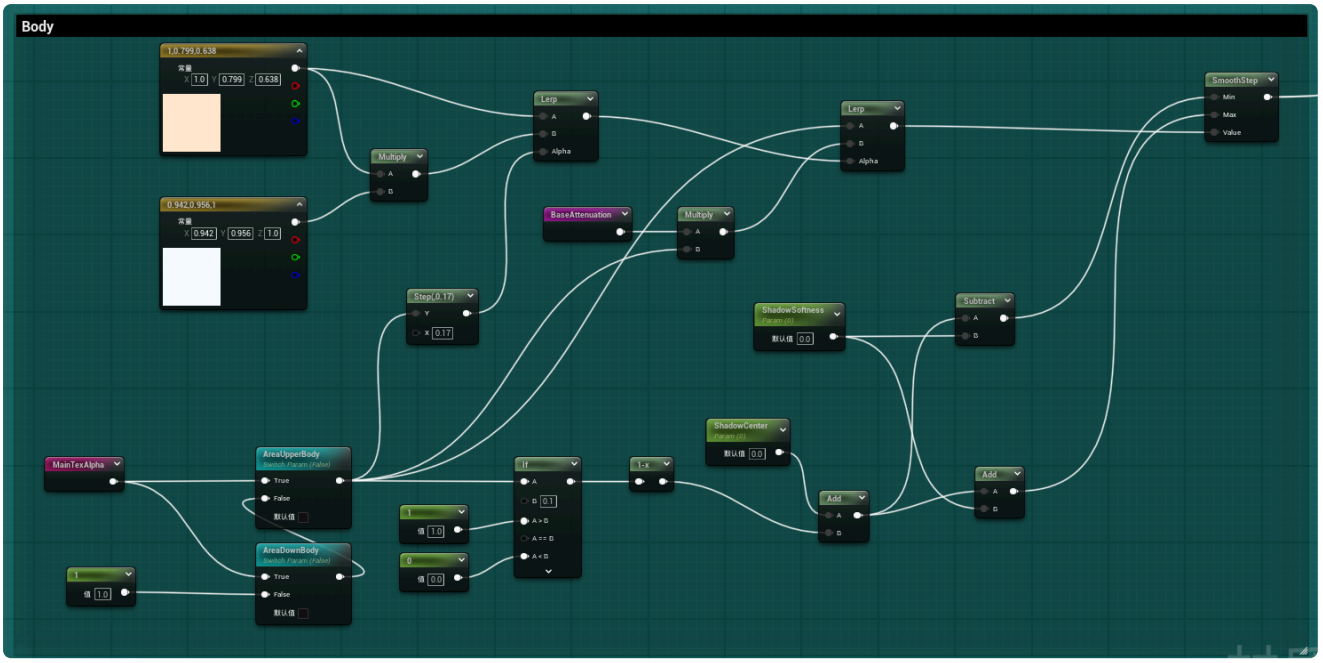
鸣潮的法线图是存在了RG通道当中，B通道是其他的信息，需要通过RG通道的法线信息，算出B通道的法线信息。



光照

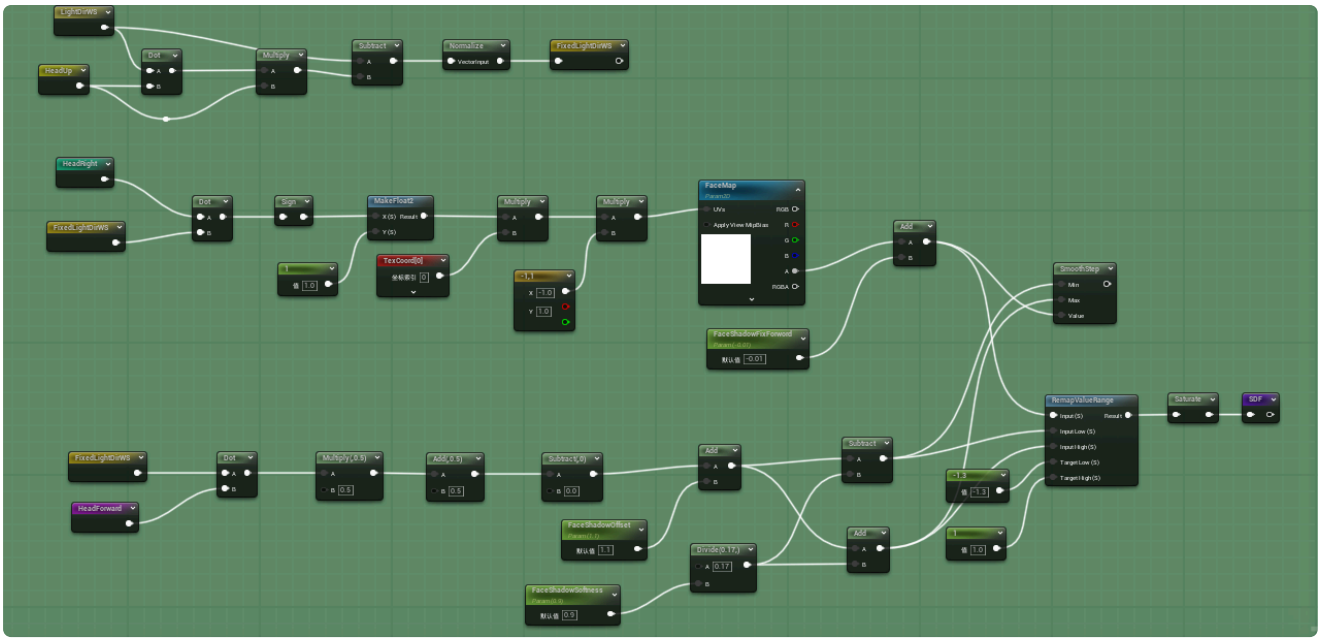
Body部分

基本上光照都离不开Lambert，在body上我的总体思路主要就是围绕lambert展开，利用smoothstep软化一下阴影，再用if（step同样效果，只是我觉得if更直观）计算一下光照贴图做一下Mask控制阴影的强度以及常暗（最暗）阴影的添加，就是body基本的阴影计算了。



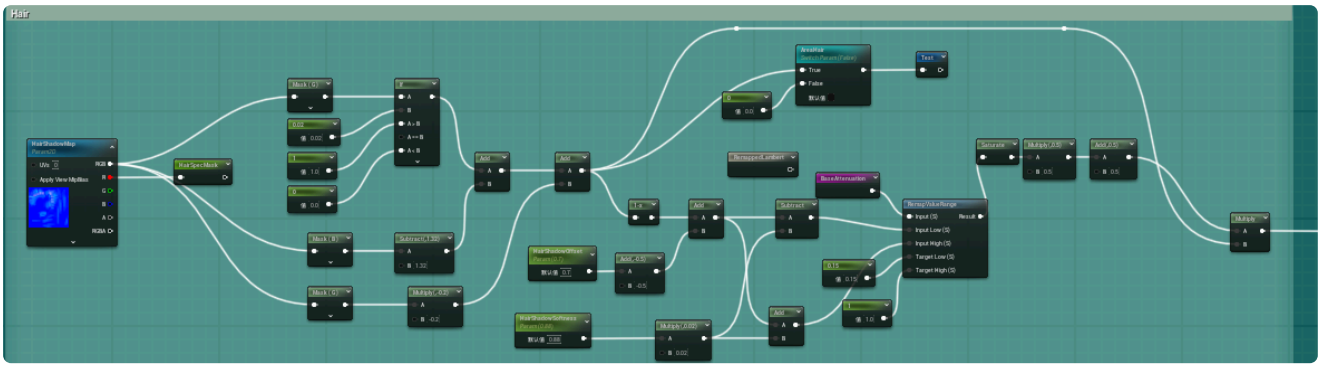
Face部分

脸部方面鸣潮用的也是现在用的最多的方案，距离场阴影SDF做face部分的阴影计算，通过采样一张处理好的sdf贴图，实现脸部阴影。



Hair部分

Hair部分用到了HM贴图的GB通道，总体思路就是先将几个部分的阴影单独计算出来然后在相加就完事了（感觉是废话），头发的常暗（最暗）阴影，发丝的浅阴影，以及最表层的阴影。

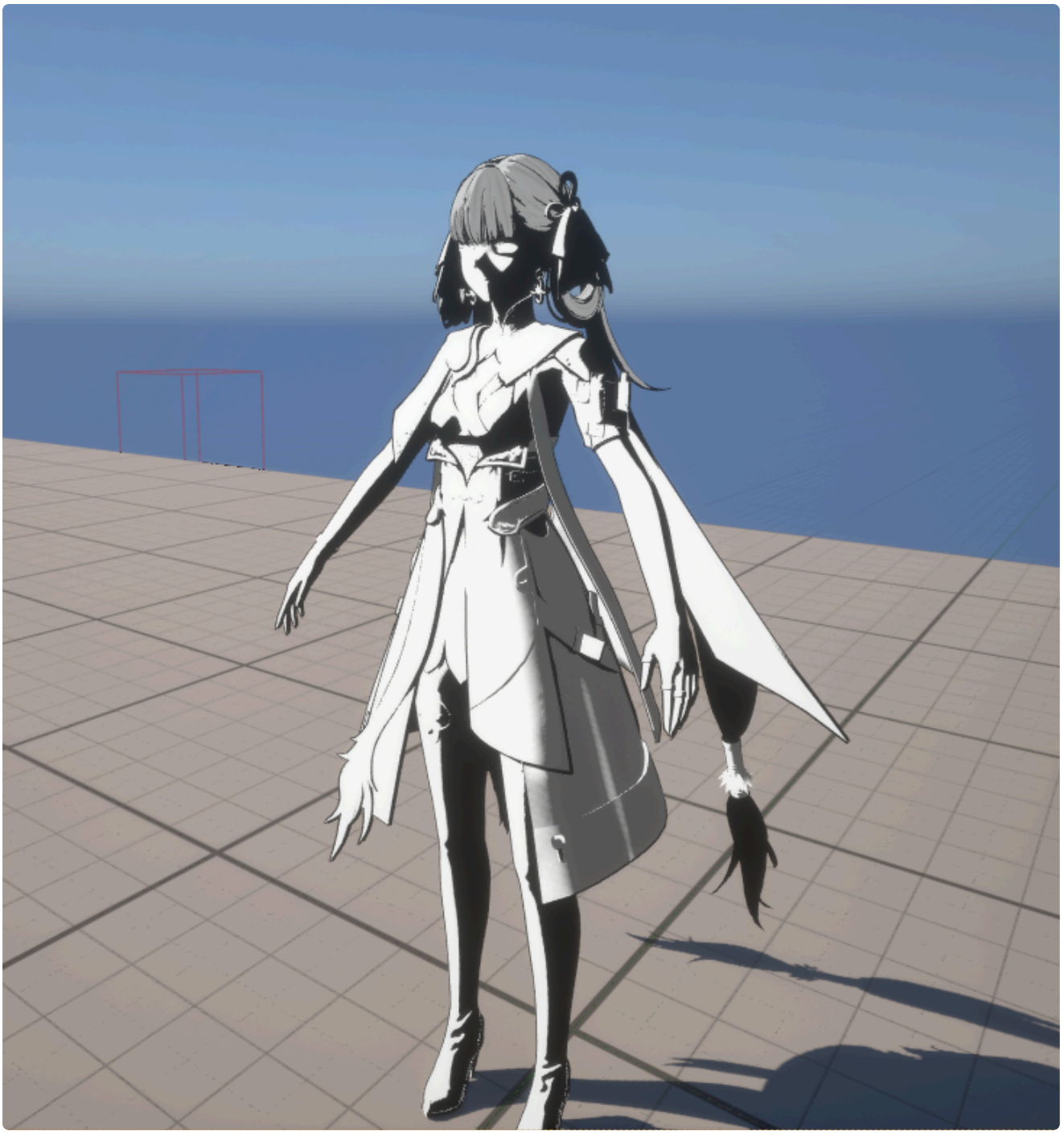


主要有几个计算吧，利用if（也可以是step）得出常暗（最暗）阴影，再利用B通道压暗一下亮面，然后在与发丝阴影相加，在做一个常规的过渡trick计算重映射一下，最后在与常暗阴影相乘。

头发阴影效果



最终光照效果展示

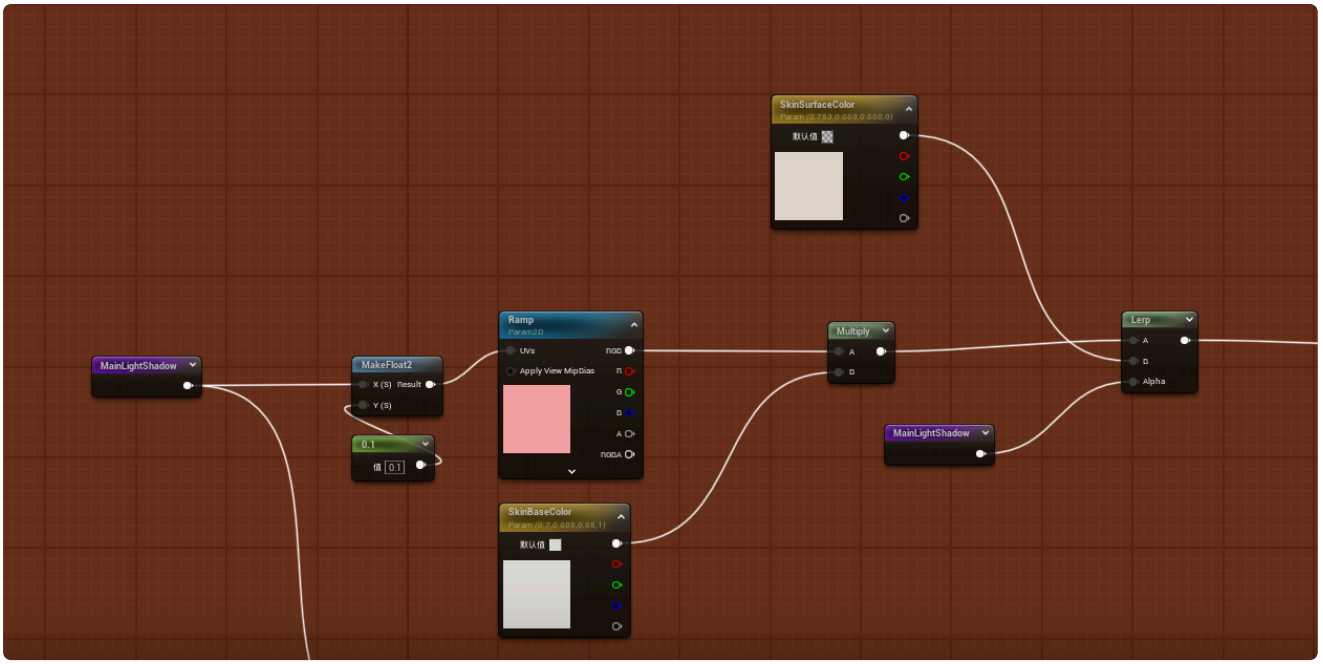


Ramp部分



鸣潮只有一个Ramp图，这个Ramp图总共有五条Ramp，第一条一看就是跟皮肤有关的ramp，第二条和第三条我猜也是跟皮肤相关的ramp，但在本次渲染并没有使用，我的推测是，鸣潮有自己的一套TOD系统（采访看见的），所以会根据时间的不同切换几个不同的ramp或者混合这几条的ramp来丰富光照效果，但这几条应该都是有关skin方面的，在衣服和

头发部分应该是有做一个可自定义动态调整的ramp来给不同角色做不同效果。

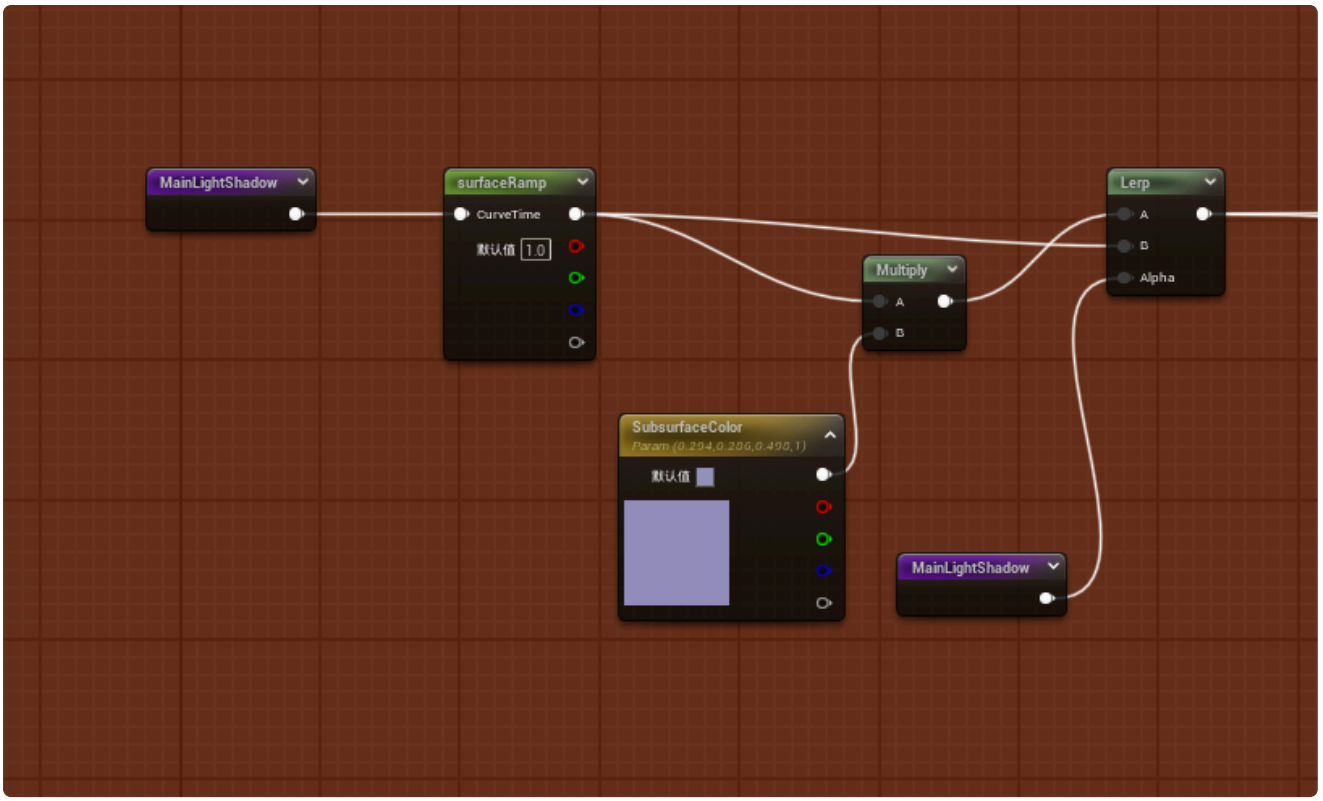


这边主要就是用光照结果去采样ramp图，然后再根据光照对亮面与暗面做lerp。再用typemask（ID图）做一个skin的区域划分。

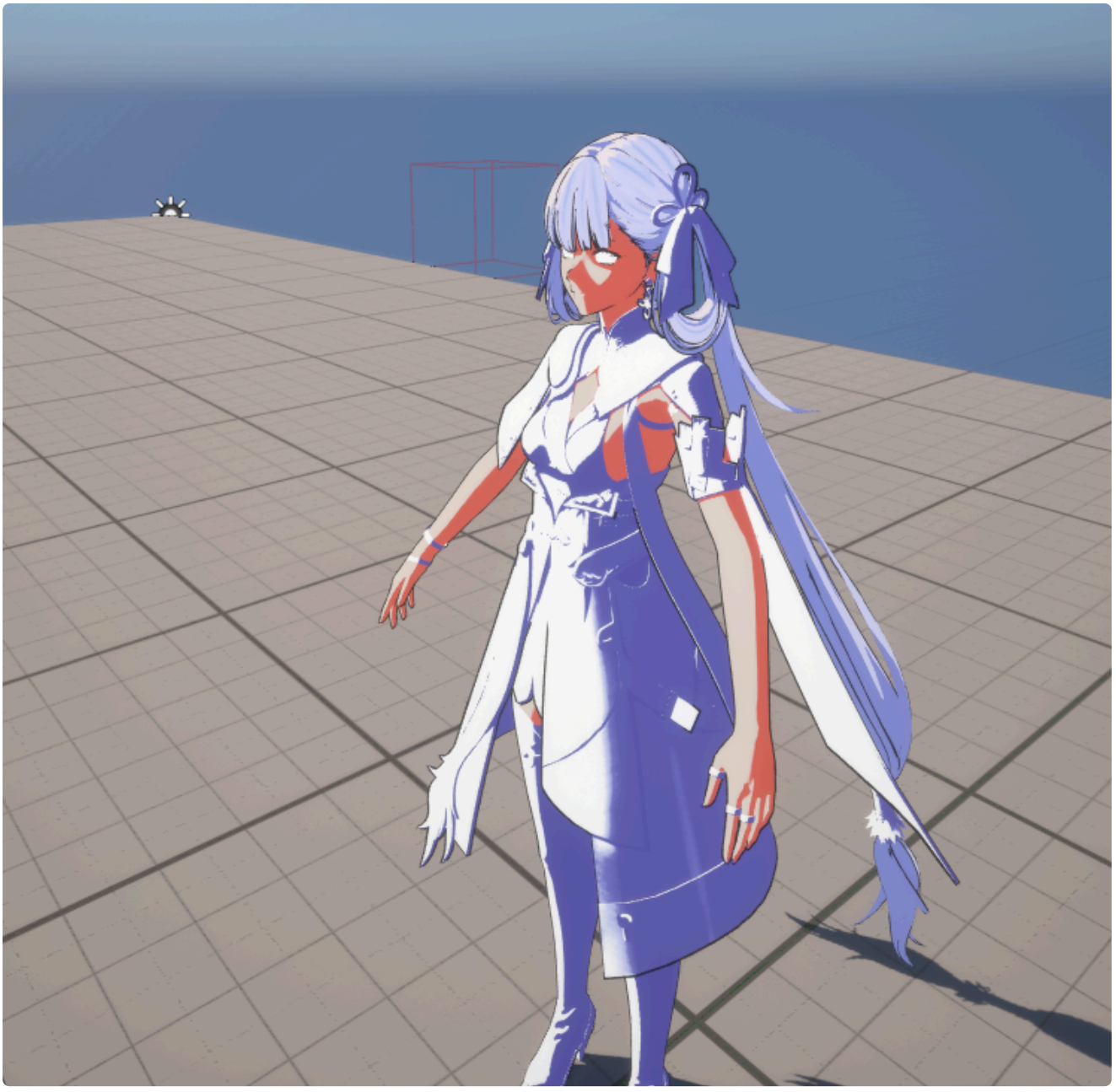
skin ramp效果



衣服与头发方面就用了UE自带的curve工具来分别各做一个渐变ramp。整体逻辑跟采样skinramp是一样的。

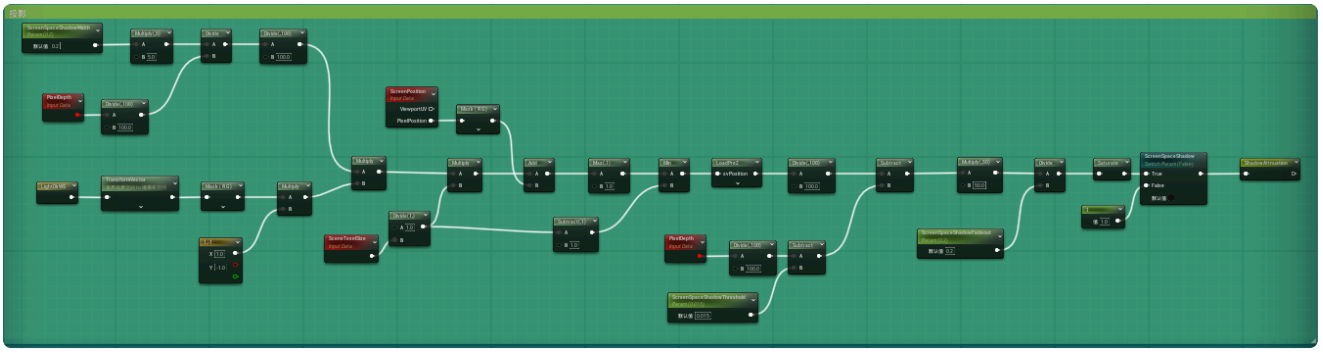

MainLightShadow
surfaceRamp
CurveTime
默认值 1.0
Multiply
A
B
Lerp
A
B
Alpha
SubsurfaceColor
Param (0.294,0.206,0.490,1)
默认值
MainLightShadow

Ramp总效果



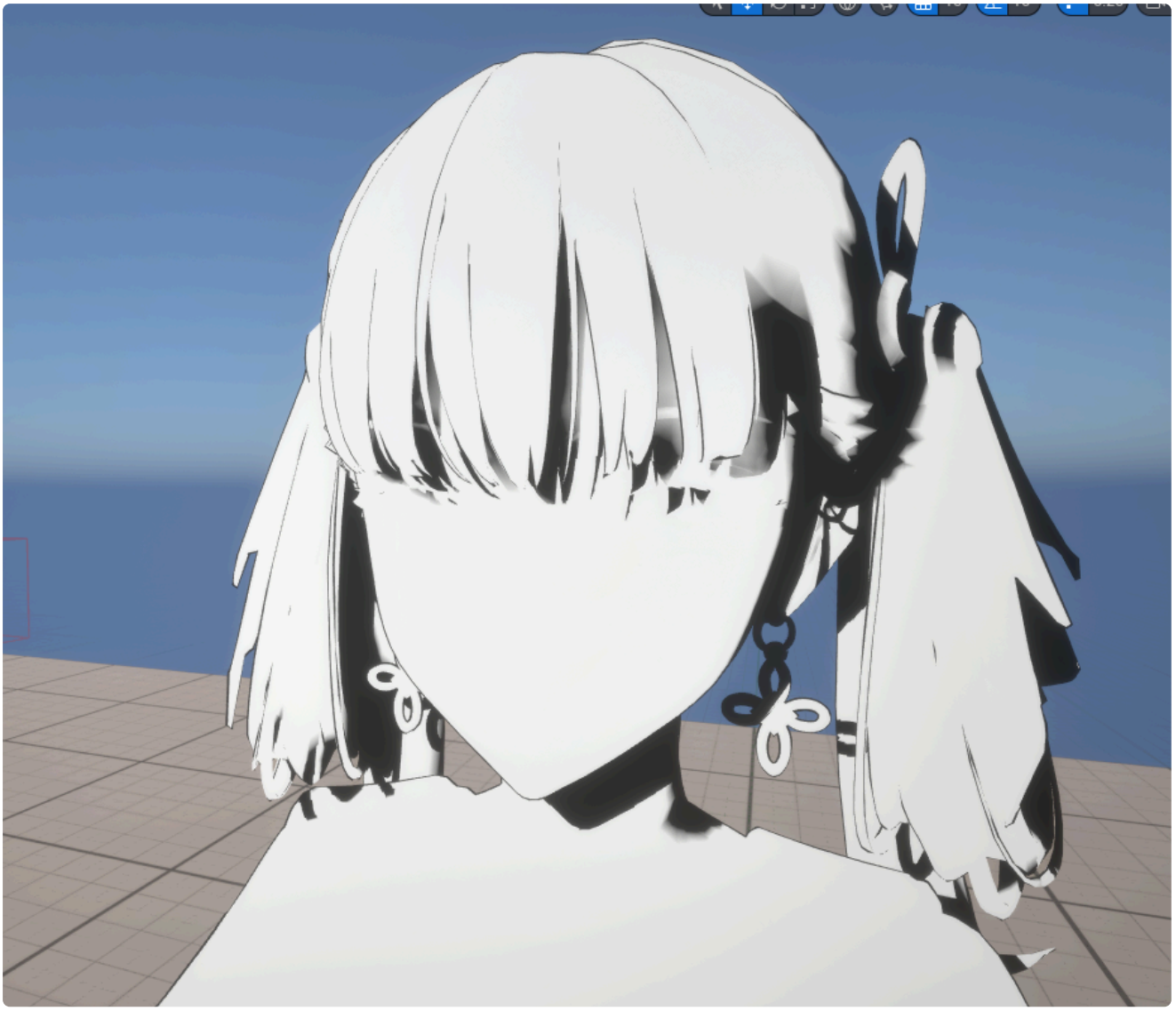
投影

投影方面主要使用的是屏幕空间下的深度实现的。鸣潮使用的方案应该是用模板测试实现的，这也是现在使用的最多的方案吧，这边没用是因为我还没搞懂UE的模板测试以及怎么用鸣潮的Het贴图来控制这个模板测试，目前我的的大致思路是复制一个相同的mesh，在新建一个专门写入模板值的材质，将材质blendmode设为mask，然后在里面进行操作与计算，但还没有跑通，可能是得改引擎才能实现了，蓝图节点还是太受限了，其实直接在模型上面操作一下也都能实现，就把刘海mesh直接拆分出来就行，但偷懒了，而且也想研究研究鸣潮是咋弄的，所以刘海投影部分就用这个方案过渡着用用了，仅用于角色展示还是没问题的。



主要思路来源于下面这篇大佬的文章。
https://zhuanlan.zhihu.com/p/552489625
核心思路就是将当前的深度图根据光照方向进行偏移，在与原本的深度图做对比。
需要注意的是深度图的规范问题即可。

投影效果



Specular高光

高光方面主要是PBR，用的方法是cook-torrance brdf的GGX高光部分。

Cook-Torrance BRDF 的高光公式通常表示为：

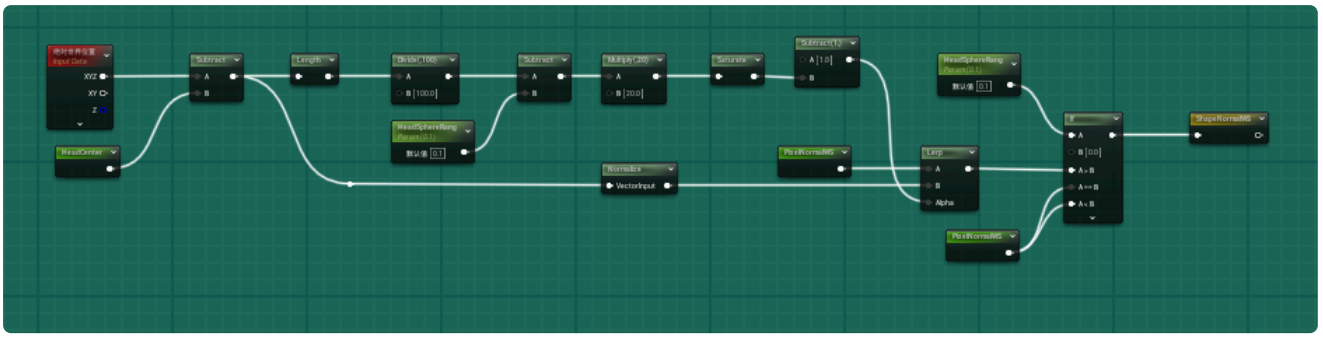
$$f_{specular} = \frac{D(h)F(v,h)G(l,v,h)}{4(n \cdot l)(n \cdot v)}$$

整体公式大概如下：

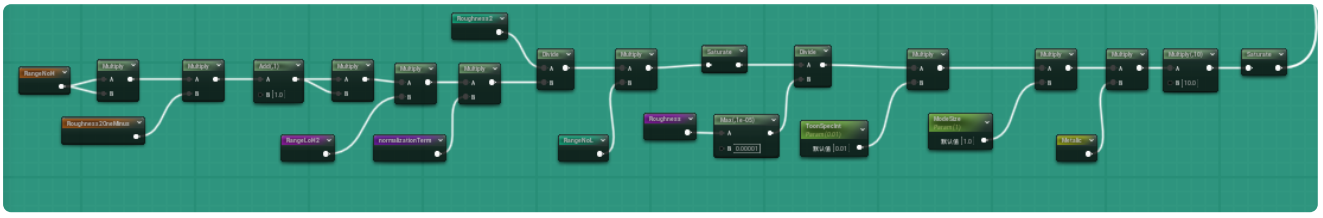
```
D = roughness^2 / (NoH^2 * (roughness^2 - 1) + 1 )^2
V * F = 1.0 / (LoH^2 * ( roughness + 0.5 ))
Spec = (D * V * F) / 4.0 * NdotL
```

上面是unity官方优化过的GGX。主要就是将G项（几何遮蔽）进行了简化。

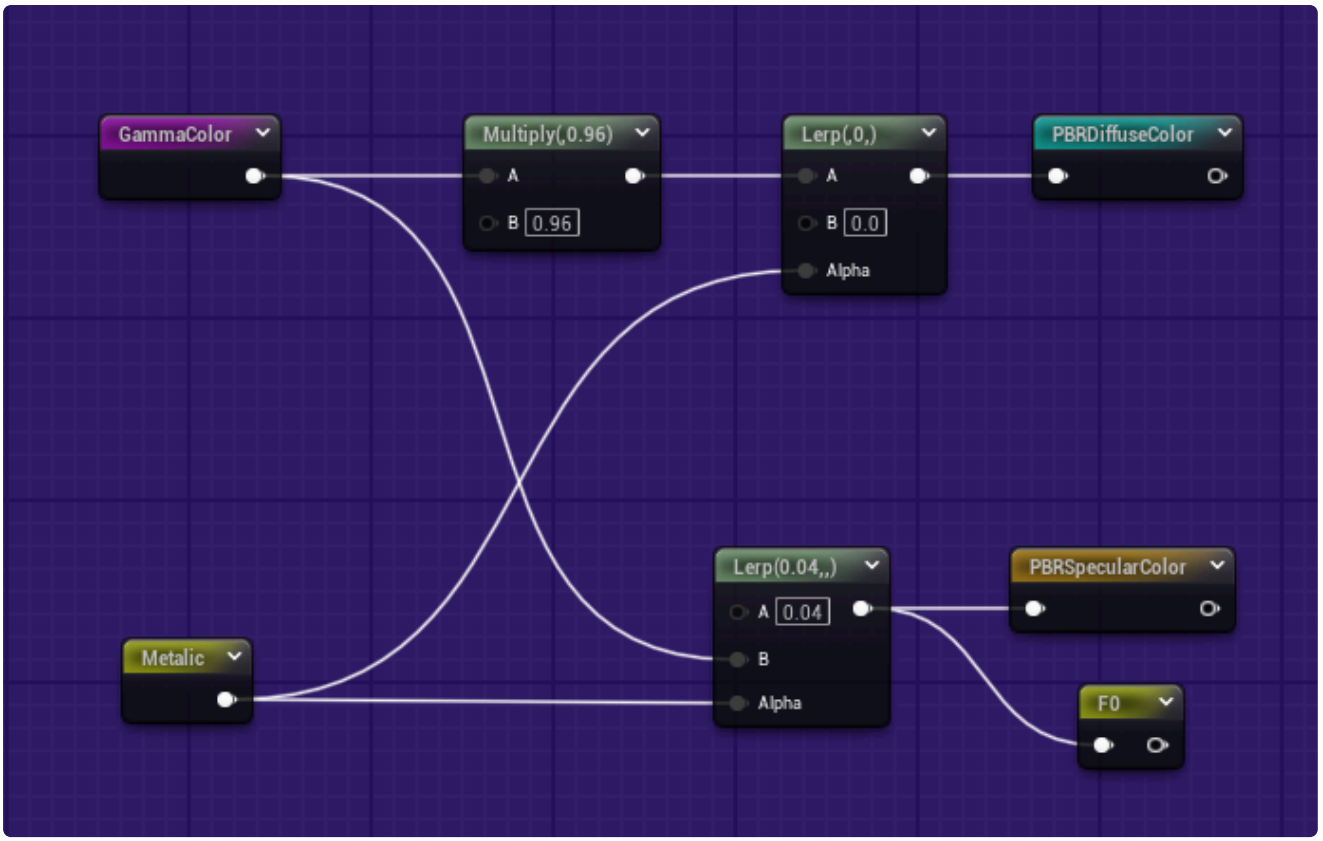
在做卡渲高光的时候头发的法线要单独去做一个球形法线，使计算出来的高光是一大块平滑的高光，而不是松散零碎的高光。



这边是单独算了个遮罩然后lerp了球形法线与贴图法线。



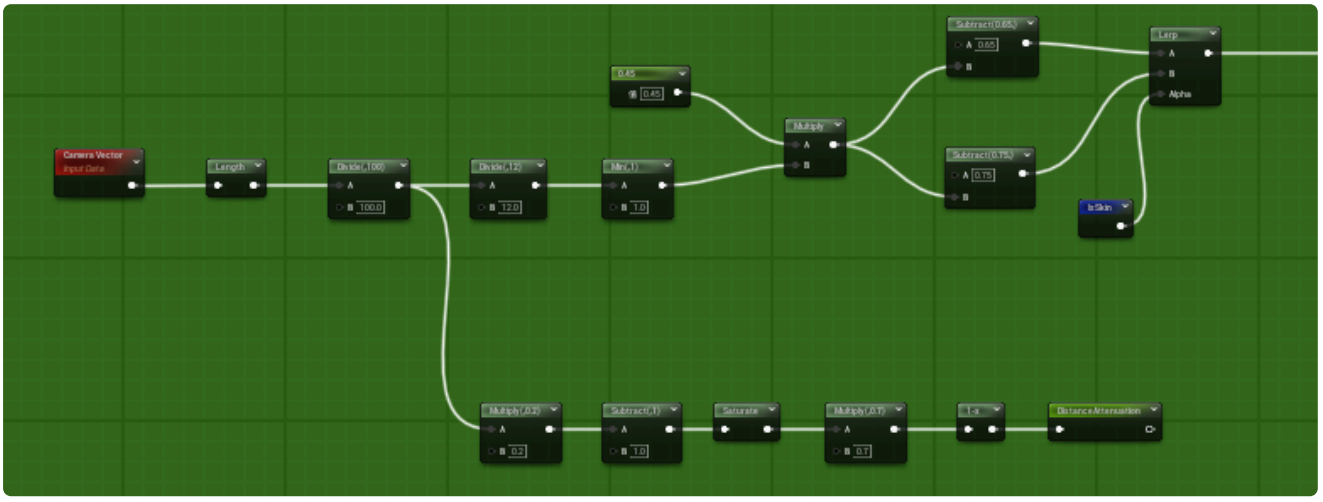
跟着ggx的公式计算高光即可。然后在利用两个参数来控制高光。

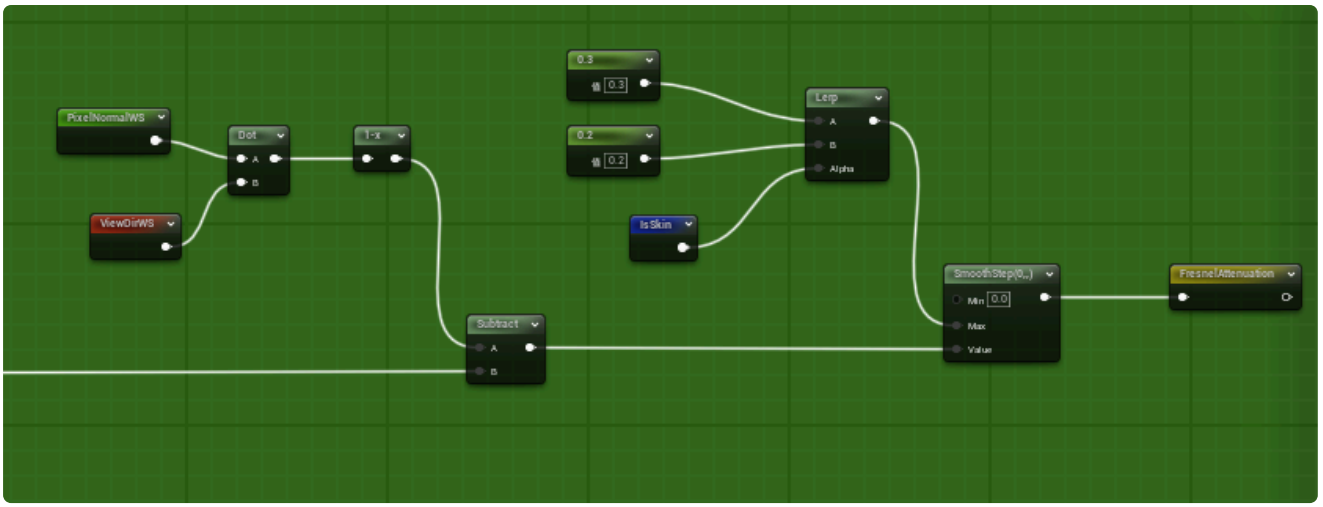


需要注意的是要对金属与非金属之间进行划分，非金属的最大漫反射比例是0.96。非金属的镜面反射比例就是1-0.96=0.04。

边缘光

边缘光主要就是用菲涅尔进行实现 1-（N dot V）



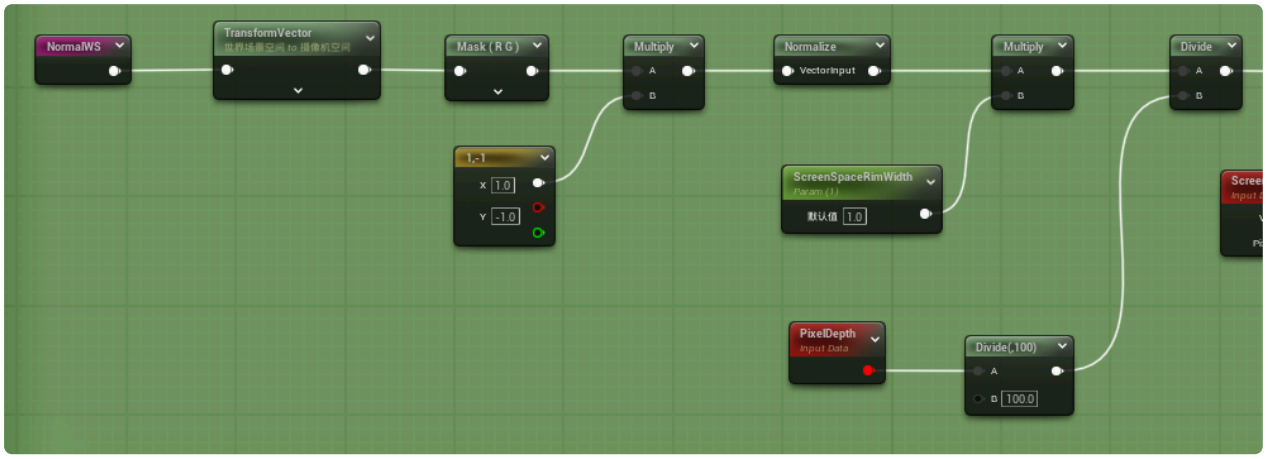


在计算菲涅尔的时候引入相机距离，并根据皮肤进行判断计算min(1, camera / 12.0)最大值为1，当相机距离为12米时为最大。

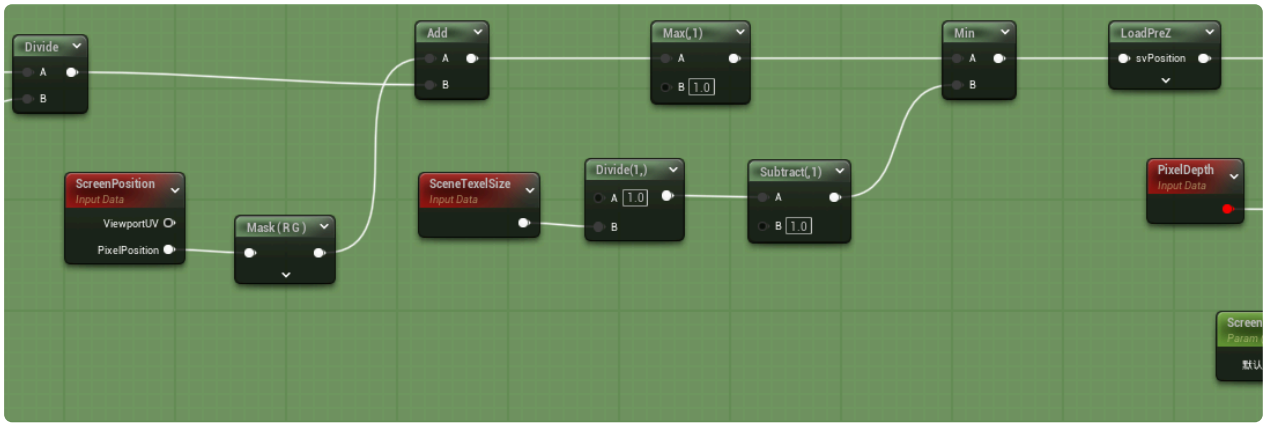
屏幕空间边缘光

这边也计算了屏幕空间边缘光，主要是基于深度边缘检测的屏幕空间边缘光与上面计算投影的思路相像
主要的思路是：

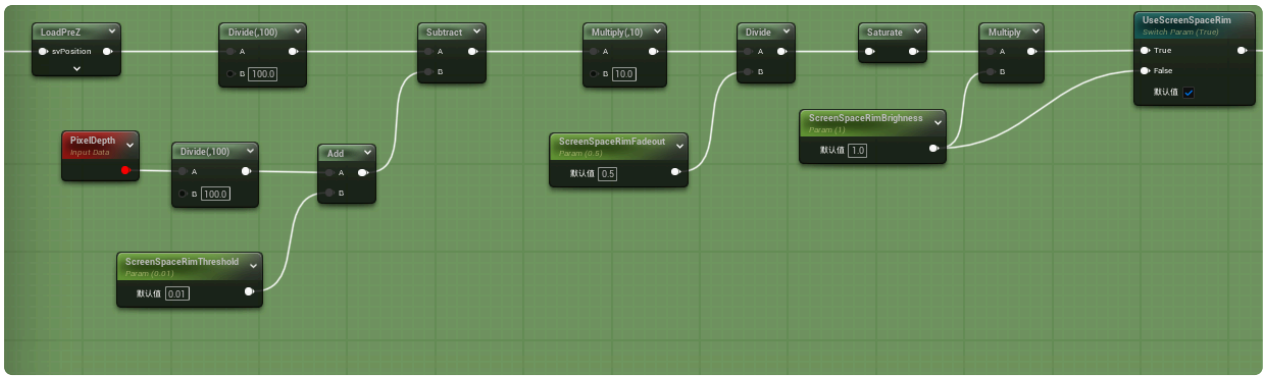
- **在屏幕空间中，沿着法线方向进行一个小小的位移。**
- **比较位移后采样到的深度与当前像素的深度。**
- **如果两者深度差异很大，则说明这个像素位于物体的边缘。**



法线偏移的UV计算



计算采样相邻像素的纹理坐标以及限制采样坐标在屏幕范围内，防止越界。



采样偏移后的深度并且转换到线性的线性观察空间深度，在进行边缘检测计算也就是相减，如果偏移位置的深度比当前像素深得多 ，那么这个差值会大。然后在用两个参数进行控制 FadeOut是缩放因子， brightness则是亮度。

最终效果



未能实现的部分

眼睛着色部分

目前眼睛部分只是单独贴了个贴图上去，但其实鸣潮的渲染当中有对眼睛做蛮多的细节计算的。像是视差效果，眼睛的虹膜还有高光。

模板测试部分

像是利用模板测试做的眉透，眼透，刘海投影等。