1、我做了什么东西

1.在shadowpass中插入了自己的绘制片段，其将会在其他当前需要绘制的层级阴影绘制完成后调用，采用的方法是设定相机参数后调用上下文的drawrenderers（不去自己绘制）

```
private void RenderLeadShadow(CommandBuffer cmd, ref ScriptableRenderContext context,
    ref ShadowSliceData shadowSliceData,ref ShadowDrawingSettings settings, Camera camera, Lig
{
    Vector3 lightDir = light.transform.rotation * Vector3.forward;
    SaveMainCameraSettings(ref camera);
    ConfigCameraToShadowSpace(ref camera, lightDir);
    context.SetupCameraProperties(camera);
    cmd.SetRenderTarget( (RenderTargetIdentifier) m_MainLightShadowmapTexture);
    cmd.SetViewport(new Rect(shadowSliceData.offsetX, shadowSliceData.offsetY, width: shadowSli
    cmd.EnableScissorRect(new Rect( x: shadowSliceData.offsetX + 4,  y: shadowSliceData.offsetY +
    context.ExecuteCommandBuffer(cmd);
    cmd.Clear();
    ShaderTagId shaderTagId = new ShaderTagId( name: "depthonly");
    SortingSettings sortingSettings = new SortingSettings(camera);
    DrawingSettings drawingSettings = new DrawingSettings(shaderTagId, sortingSettings);
    FilteringSettings filteringSettings = FilteringSettings.defaultValue;
    context.DrawRenderers(settings.cullingResults, ref drawingSettings, ref filteringSettings
    context.Submit();
    cmd.DisableScissorRect();
    context.ExecuteCommandBuffer(cmd);
    cmd.Clear();
    RevertMainCameraSettings(ref camera);
}
```

```
public void SaveMainCameraSettings(ref Camera camera)
{
    settings.position = camera.transform.position;
    settings.rotation = camera.transform.rotation;
    settings.farClipPlane = camera.farClipPlane;
    settings.nearClipPlane = camera.nearClipPlane;
    settings.aspect = camera.aspect;
    camera.orthographic = true;
}

// 还原相机参数，更改为透视投影
1 usage  Insulation
public void RevertMainCameraSettings(ref Camera camera)
{
    camera.transform.position = settings.position;
    camera.transform.rotation = settings.rotation;
    camera.farClipPlane = settings.farClipPlane;
    camera.nearClipPlane = settings.nearClipPlane;
    camera.aspect = settings.aspect;
    camera.orthographic = false;
}
```

这样操作的话不仅仅在任何情况下均可正常绘制，而且之前写的剔除等设置也完全正常，这里有个问题就是为什么会这样，可能跟引擎里面的看不到的代码有关

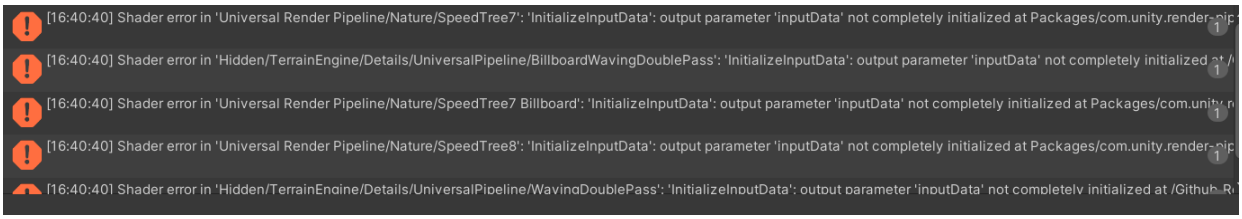
2.在Lighting.hlsl,Input.hlsl,shadow.hlsl等文件中修改，增加一次额外的阴影采样，在处理上，采用了两次采样的结果相乘来当作最终结果

3.添加各种宏定义，制作控制面板，无论怎么设置（开不开cascades，做不做主角的单独渲染，开几个cascades都能正常运行）

2、我还需要做什么东西

现在在修改一次shader源码后，会报个错但是正常运行（添加在input里的一个变量没有初始化，但是在其他的引擎内置shader中没有用到）

```
struct InputData
{
    float3  positionWS;
    half3   normalWS;
    half3   viewDirectionWS;
    float4  shadowCoord;
    half    fogCoord;
    half3   vertexLighting;
    half3   bakedGI;
    float4  shadowCoordLead ;
};
```



因为这个地方读不到我设置的宏开关，所以暂时没法根据面板上的设置进行控制，不知道这个问题是否重要

3、目前在做的是分享

框架是：1.背景介绍2.基础原理3.改造方案4.结果展示

先对自己做个简单的介绍，然后介绍这个问题的背景，从主角的阴影模糊切入，讲述模糊的原因（分辨率低）
接着讲述cascades shadow mapping的基本原理，讲述如何分割，有哪些视锥的计算方法（包围盒），有什么区别，为什么用包围球更好可以避免抖动，怎么计算这个包围球的信息，最后怎么用这个来算矩阵。然后延伸到主角的包围球和视锥的计算。
然后讲怎么对urp的渲染管线进行改造，管线中一个阴影pass是什么样的结构，怎么插入新的绘制命令来绘制主角区域。原先管线中怎么对级联阴影图进行采样，怎么修改能进行两次采样。最后讲讲中间遇见的一些困难。
最后展示一下成果的截图/视频前后对比作为结束。