**点光源和聚光灯**

**点光源**

点光源发出的光线在某点的亮度值与该点到光源距离的平方成反比。

其他光源类型的数据

1. 我们定义非定向光源的数量最大值为64。

2. 定义光源数量，颜色以及光源位置发送到GPU。

光源随距离衰减

光照越远，亮度越低。应遵循公式：i / d2。

其中i为光照强度，d为光照距离，这被称为反平方定律。

限制光照范围

限制光照的最大范围，超过这个范围就将光照强度设为0。​max(0, 1 – (d2/ r2)2)2

**聚光灯光源**

聚光角度

聚光灯有一个角度用来控制光锥的宽度，这个角度是从中间测量的。除此之外，还有一个单独的内角，控线光线以及何时开始衰减公式如下：

saturate(da + b)2

a = 1 / (cos(ri / 2) – cos(ro / 2))

b = -cos(ro / 2)a

配置内角

聚光灯始终可以配置外角角度，但在URP被引入之前是没有单独的内角的。所以灯光的Inspector面板中没有暴露内角角度，渲染管线可以通过覆盖灯光的Inspector面板来修改灯光，这是通过创建编辑器脚本来扩展LightEditor，且给它CustomEditorForRenderPipeline属性完成这个操作。该属性第一个参数必须是Light类型，第二个参数是我们希望覆盖Inspector面板的渲染管线资产类型。

1. 在CustomRP的Editor子文件夹下创建CustomLightEditor脚本。

2. 要替换Inspector面板，首先重写OnInspectorGUI方法，我们需要做的额外操作就是首先检查是否仅选择了聚光灯，通过settings中的属性可以进行光源类型的判断，然后调用DrawInnerAndOuterSpotAngle方法绘制一个调节内外聚光角度滑块，最后调用ApplyModifiedProperties应用该滑块所做的修改即可。

烘焙光照和阴影

烘焙光照

只需要将点光源和聚光灯的灯光组件Mode属性改为Baked，进行烘焙即可（若要烘焙阴影，修改Shadow Type选项）。然后会发现烘焙后光照比较亮，因为Unity默认使用了错误的灯光衰减，和旧版渲染管线的结果相匹配。

1. 我们可以告诉Unity使用不同的衰减，通过在Unity编辑器中执行光照烘焙之前提供一个委托方法。

2. 仅对于编辑器，需要重写lightmapper设置光照数据，通过提供一个委托方法，来传入一个Light数组。最后输出一个NativeArray<LightDataGI>结构委托的类型是 Lightmaing.RequestLightsDelegate。

3. 根据不同的光源类型，创建一个LightDataGI结构，调用LightmapperUtils的Extract方法，参数是光源和光源引用结构，然后调用光源数据的Init方法。

4. 对所有的灯光数据的衰减类型设置为FalloffType.InverseSquared。

5. 调用Lightmapping.SetDelegate方法，把我们定义的委托作为参数传递过去。

6. 当我们的渲染管线被处理时我们还需要清理和重置委托，通过重写Dispose方法，先进行清理，然后调用Lightmapping.ResetDelegate来重置委托。

阴影蒙版

把点光源和聚光灯的Mode设置为Mixed也能将阴影烘焙到ShadowMask中。每个光源都使用一个通道，就像方向光一样。但由于其范围有限，因此多个光源可以使用同一通道，只要它们不重叠。因此，阴影蒙版可以支持任意数量的光，但每个纹素最多只能支持四个。如果多个光源在尝试声明同一通道时重叠，那么最不重要的灯将强制设置为Baked模式，直到不再发生冲突。

(代码实现)