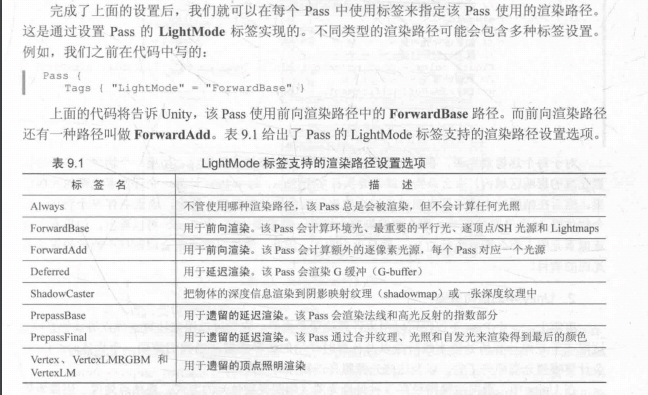
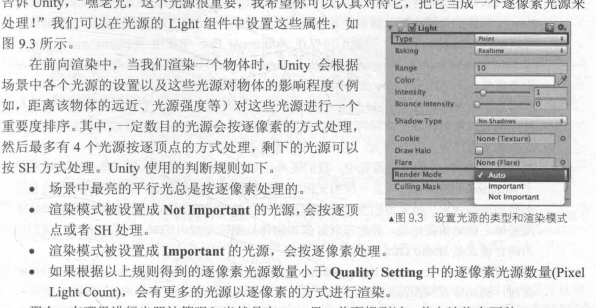
1. 渲染路径：

主要分为前向渲染和延迟渲染，需要在shader的pass中指定，例如



不同的渲染路径会告诉unity应该在什么变量中填充数据，比如延迟渲染gbuffer中就会被赋值相应的数据，如果指定错误，则可能会造成错误的赋值，导致计算结果出错。

1. 前向渲染
2. 每个pass都计算一个逐像素光源(render mode设为important就为像素光源，否则为顶点光源，Quality setting里有最大的像素光源数)的光照结果，然后在帧缓冲中把这些光照结果混合起来得到最终的颜色值。即N\*M的时间复杂度



1. 延迟渲染
2. 延迟渲染主要包含了两个pass

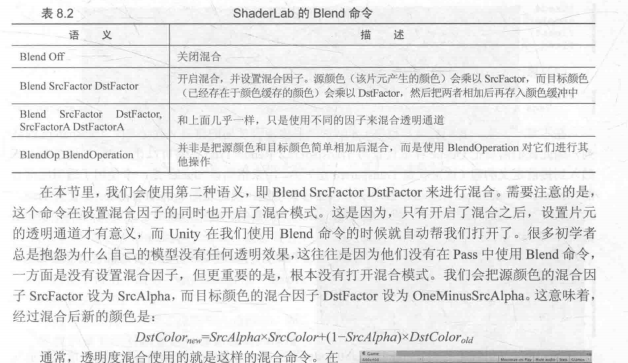
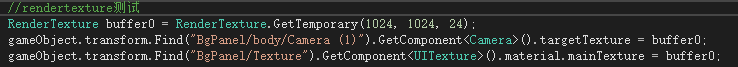
第一个pass中，不进行任何光照计算，而是仅仅计算哪些片元是可见的，这主要是通过深度缓冲技术来实现的，当发现一个片元是可见的，我们就把它的相关信息存储到Gbuffer中；

第二个pass中，利用Gbuffer的各个片元信息，例如表面法线、视角方向、漫反射系数等进行真正的光照计算

ii)

1. 渲染纹理的东西，GrabPass OnRenderImage RenderTexture的区别，<https://forum.unity.com/threads/graphics-blit-vs-grabpass-vs-rendertexture.523825/>

目前的理解是，grabpass和renderimage都是需要额外拷贝一份的，因为只有一个摄像机，又要输出到屏幕，又要输出到纹理，但是RenderTexture就不需要，因为设置了rt的摄像机就不会输出到屏幕了，所以其实还是一份，这也是为什么不那么耗的原因，但是要显示rt，就还是会拷贝一份输出到屏幕…写着写着又不知道了,又看了看，效率应该是差不多的，应该是rt可以自定义图片大小，这会使效率降低一些

1. GPU架构<https://zhuanlan.zhihu.com/p/112120206?utm_source=wechat_session&utm_medium=social&utm_oi=646005941838024704>
2. 透明叠加，写的时候命令是完全懂了，但是实现出什么样的效果，还得看看
   1. 
   2. 顺便说一下遇到的问题，研究了两天，唉，问题是看到网上说把模型渲染到RenderTexture中，然后把RT放到UITexture中，这样就可以和UI一样的层级管理，步骤打开是这样的：
      1. 摆好摄像机和模型，摄像机设为Solid Color，颜色都设为0
      2. 新建一个UITexture用于放RT
      3. 
      4. 注意24是深度，如果不加的话深度会有问题
      5. 上面都简单，但是这个主要问题出在透明物体以及叠加Shader中，由于颜色混合需要两步：1.Model->rt 2.RT->UITexture 所以颜色混合会有问题，解决办法是
         1. 
3. 阴影的实现：
   1. 首先，获取摄像机位置的深度图
   2. 获取光源位置的深度图
   3. 如果两个向量在同一点结束，则相机和灯光都可以看到该点，则是亮的
   4. 如果光的矢量到达该点之前结束，则光被遮挡，有阴影
   5. 如果摄像机看不到就不处理了