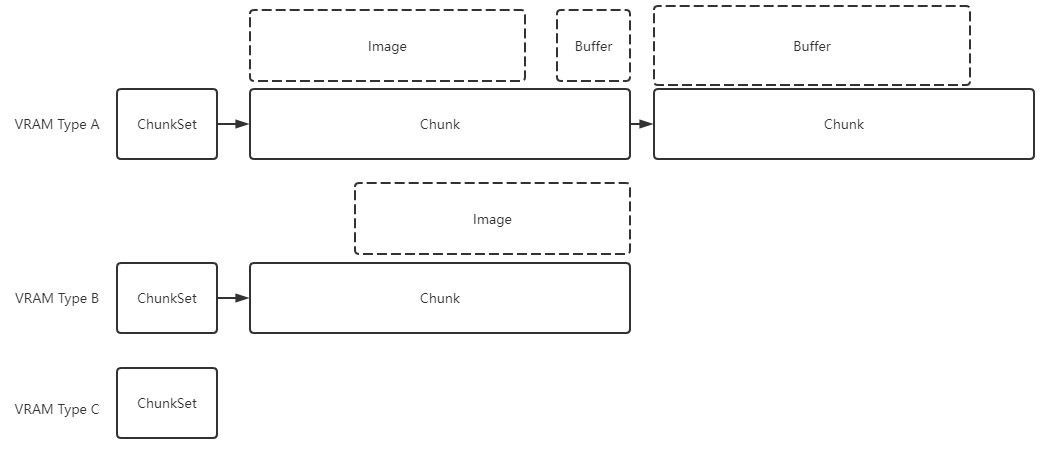
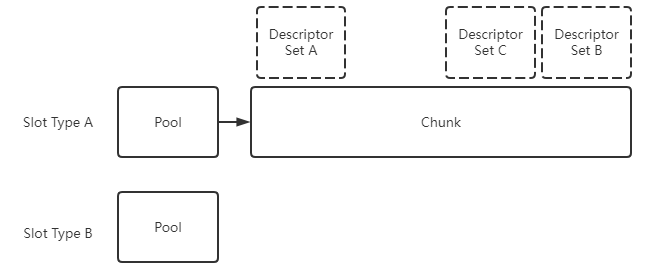
# 显存管理

由于存在分配显存次数的限制与性能的要求，使用了首次适应算法的显存池，根据显存类型自动创建分配并收集。



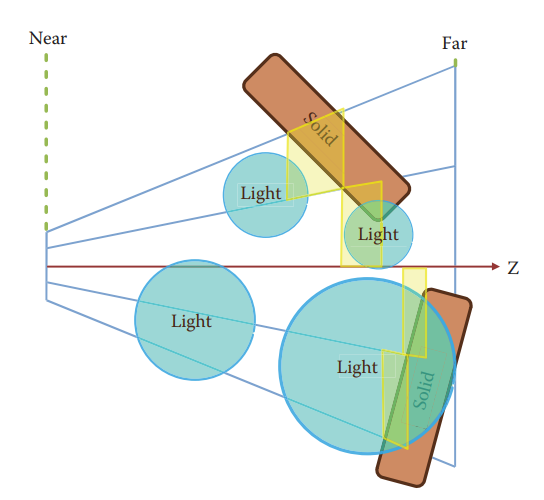
# 描述符管理

描述符管理使用了ARM的持久化方案，在加载Shader时自动解析生成DescriptorSetLayout，而Material实例实际持有DescriptorSet用于保存资产信息，只在逻辑线程修改资产时更新DescriptorSet，在渲染线程只负责向CommandBufer绑定。同时使用了池化技术。



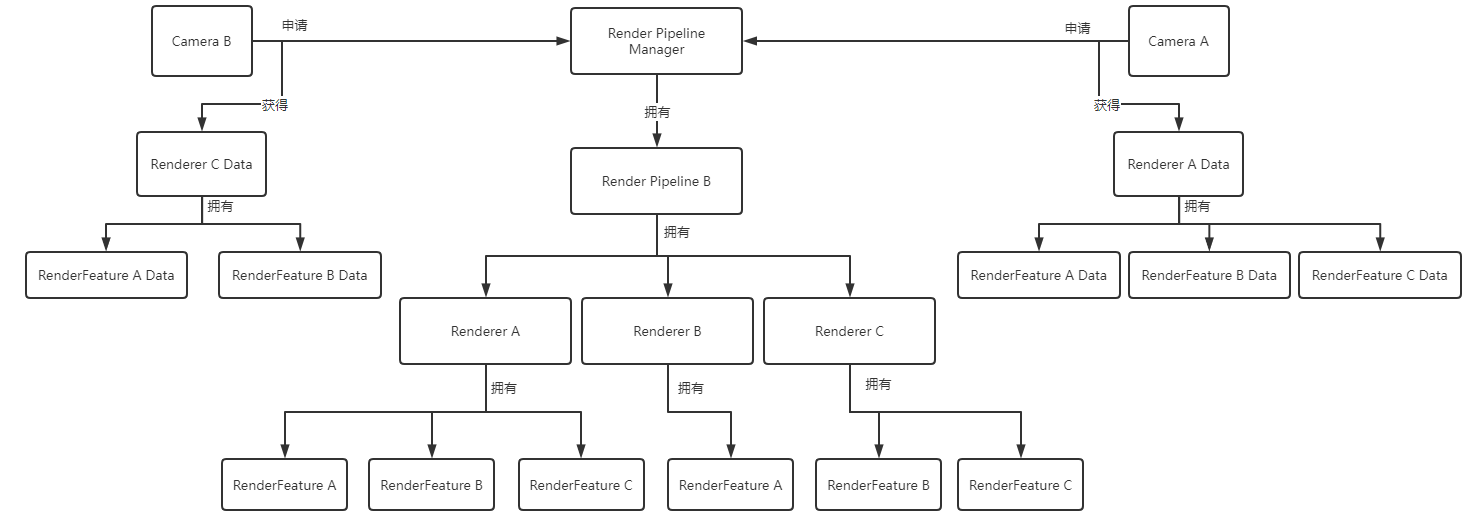
# Forward+包围盒压缩

在计算分块灯光表时使用深度图对包围盒进行了压缩。

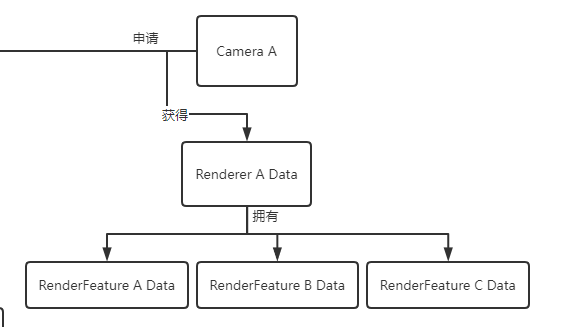


# 渲染特性管理

仿照unity srp的对象，对渲染特性分类管理。全局只有一个Pipeline，往下分类为多个Renderer，Renderer内部的RenderFeature可实现复用。

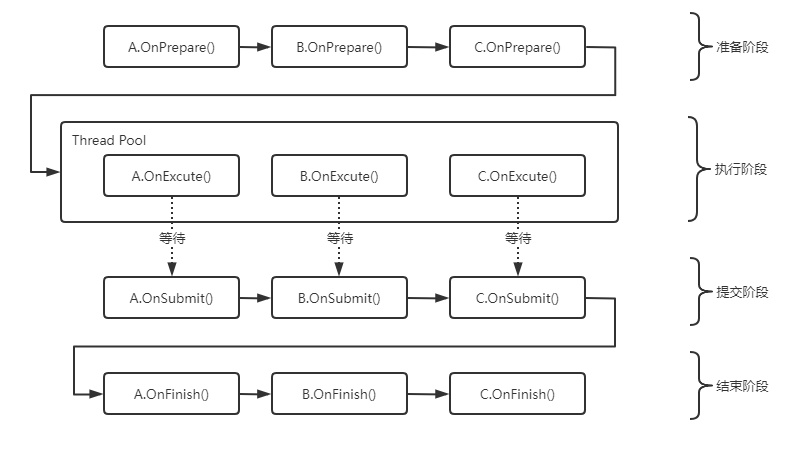


Camera只持有渲染数据，并没有管线信息。



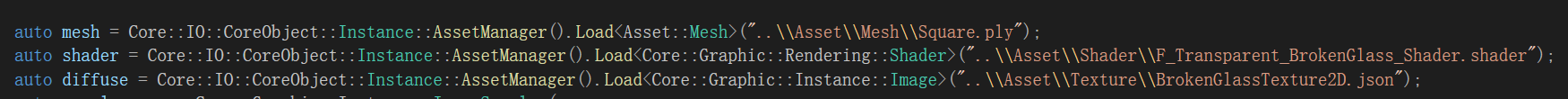
# 多线程渲染

在RenderFeature内部暴露了多个接口，在保证一定顺序的情况下供渲染线程池并行生成CommandBuffer。



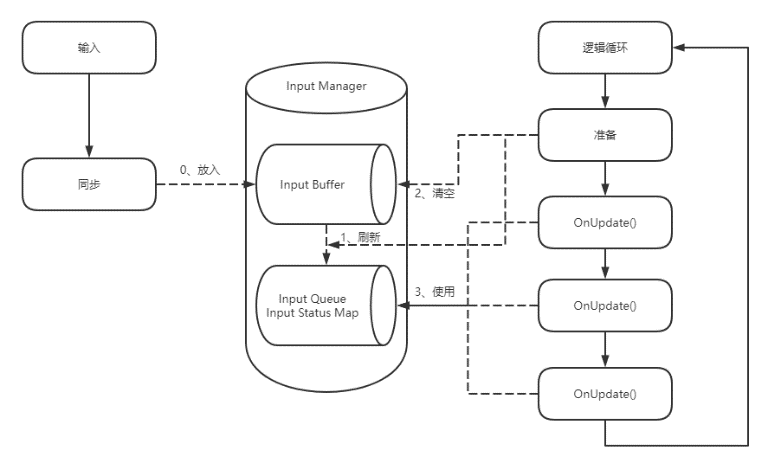
# 资产管理

构建了一个使用引用技术的资产管理类，全部资源都需要通过接口加载卸载。



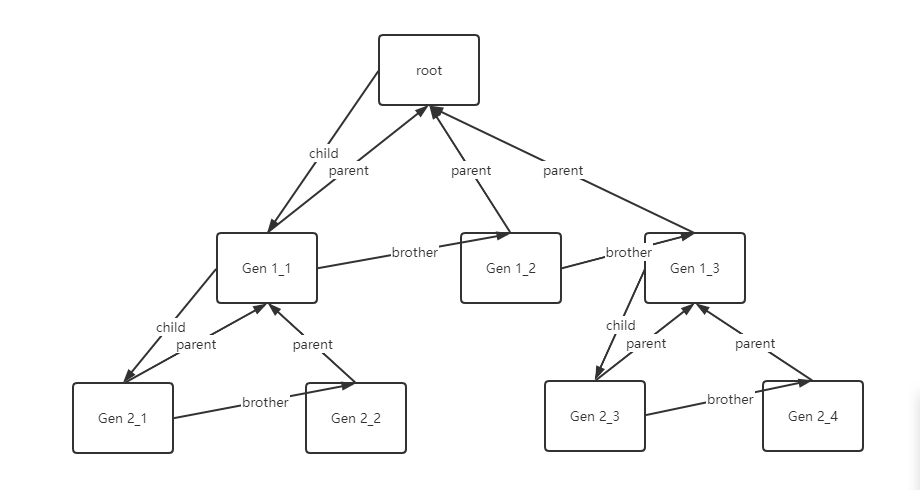
# 输入管理

输入管理队QT窗口的信息进行了缓存整理，方便使用。

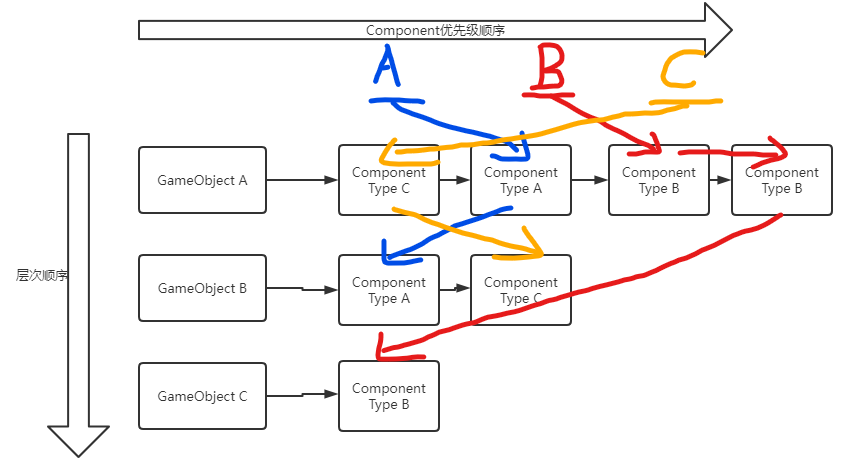
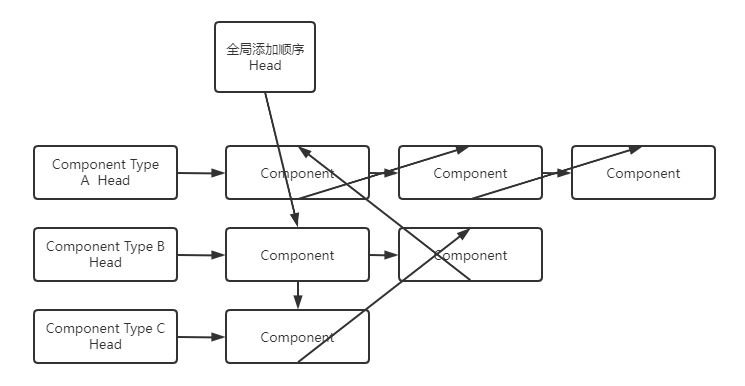


# 场景管理

使用了孩子兄弟树来存储树状GameObject，方便层次遍历。



Component使用十字链表存储，进行隐式Component类型分类与添加顺序分类，便于Component遍历。



同时集成了反射功能，可根据类型名或类型获取全部相关的Component

