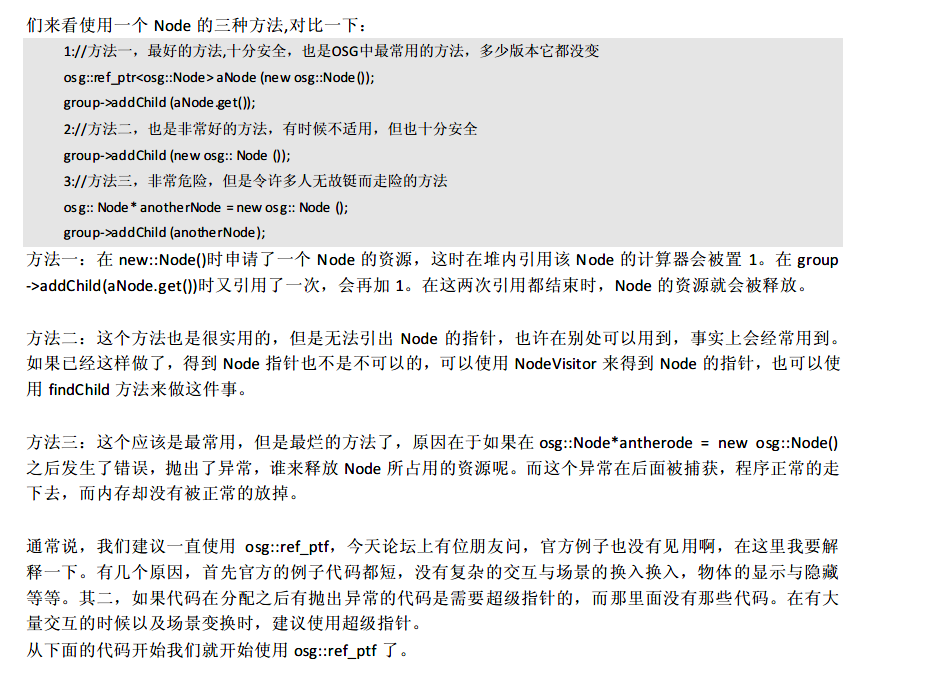
1. 有一点比较重要然后容易忽略的是，在osg程序开发中，请尽量使用osg::ref\_ptr<T\*>；这个智能指针构建起osg的对象树（绝大部分osg类都是继承于osg::Referenced，即可计数对象），类似于Qt的对象树，负责其内存的管理。当然其内存管理策略不是我们关心的，我们使用它的原因是基于其内存管理策略，大部分类的析构函数都是受保护的，意味着我们并不能直接维持对象的内存，就必须用到osg::ref\_ptr（当然这个必须是相对的，举个例子，如果我们new了一个osg::Node，然后把它addChild()进mapNode，那当removeChild的时候，这块内存其实也是会被释放掉的。但这责任分配很奇怪不是吗？）根据部分教材的说法：

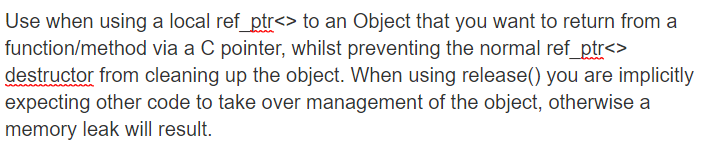


这种时候就必须由我们来负责内存释放，而这是必须通过osg::ref\_ptr完成。

至于我们怎么释放？很简单，举例：osg::ref\_ptr<osg::Node> myNode;

直接myNode = nullptr;这块内存就会被释放（基于计数原则），同时把myNode reset为空。

Osg::ref\_ptr本身非常简单，除了上面所说基本没有其他内容（而正是基于这么简单的内容就构建起整个内存管理系统）。有一个小地方可能比较引人注目，就是osg::ref\_ptr<T\*>::release()，这个函数的名字非常让人迷惑，但是相信我，基本上用不到这个函数的。根据官方的注释：



和具体实现：



重点在于“local”，经过实验，它唯一会使用到的地方就是类似于下面的代码：



也就是注释中的，局部的osg::ref\_ptr,以及必须返回C风格的指针（而不是ref\_ptr）。稍微解释一下，因为这种情况下createGemoetry域返回时就是调用geom的析构函数，这个时候就会导致引用数为0，即释放掉geom的内存，即使返回了geom的指针，也只会是野指针，所以必须返回gemo.release。

总结，总之用到osg相关的，加上osg::ref\_ptr就对了。（当然说是智能指针，最终内存还是要自己管理，谨慎小心！）原本没用到的也不是一定要改，只是不好而不是不能用。