我们的代码里用到了两个kdtree，一个是存整个场景里所有object的所有face的，比如triangle这样的几何体，是在render开始之前用场景里的object来初始化的。另一个是用来存hitpoints的，在每个round进行了ray tracing之后，在photon mapping进行之前，用ray tracing中记录的non specular的hitpoints来创建的。这两个kdtree在我们的代码里分别是在scene.cpp和hitpoint.cpp这两个文件里实现的。

那先来看一下objectkdtree，这个tree的treenode主要存储了treenode的bounding box， face的vector，以及child nodes的地址。我在构建这个objectkdtree的时候会把所有face唯一地存在所有treenodes上，当我在分割空间的时候，如果一个face同时出现在两个子空间里，那就把它存在父节点上。当我在做intersection的时候，我从root node开始查询，如果child nodes的bounding box和我当下正在考虑的ray相交的话，那再进入相应的子节点查询，是一个递归的过程。

另外一个kdtree是用来存储hitpoints的，在photon mapping的时候用来更新亮度。这个tree的tree node主要存储了以这个node为root下的kdtree的boundingbox，它需要考虑的photon的范围就是这个radius，一个hitpoint，以及child nodes的地址。在构建这个kdtree的时候，我们分别从xyz三个方向用中位数切分，每个父节点记录中位数的hitpoint，剩下的分别再在左右子节点上构建子树，bounding box记录的是以该父节点为根的所有hitpoint的范围。

那么我们在photon mapping的时候，photon没有被反射的情况下，我们需要根据这个photon对它周围的hitpoint进行update。我们是从root开始，根据每个节点的bounding box找到它附近的hitpoints。Update的主要是相关的hitpoint的radius，以及hitpoint的亮度。