# 场景管理

场景管理是 3D 引擎的一个核心部分，用户看到的每一帧数据都是经过场景管理后渲染出来的，因此场景管理的效率非常重要。

一般来说，场景管理需要考虑三种结构关系：

**层次结构**：用于描述物体之间的子随父动关系，形成一个树形结构；

**空间结构**：用于描述物体之间在空间上的相邻关系，空间上相邻的物体组织在一起；

**状态结构**：用于描述不同渲染状态的物体，相同状态的组织在一起。

为了描述上述的三种结构，一般的通用图形引擎都是采用三种不同的视图，每种视图采用不同的数据结构来对整个场景进行描述，这三种视图如下：

**场景图（树）**：整个世界的一个层次结构视图，父节点可以创建多个孩子结点，父节点的位置变换影响孩子结点；

**空间图**：整个世界的一个空间视图，通过一个特定的数据结构对空间进行划分管理，方便对物体进行拣选；

**渲染图**：世界的一个最佳状态图，相同渲染状态（例如着色器，材质）的物体在同一组中，渲染顺序相邻，减少状态转换。

上述三种视图分别完成不同的工作，场景图用于构建场景，空间图用于拣选，渲染图用于渲染，这对于引擎的设计者来说是非常不错的想法，每种数据结构都为优化性能而设计。同时整个引擎呈现给用户的只有一个场景图，即用户只需考虑如何构建整个场景而无需考虑内部的优化。

在我们实现的引擎中，实现了两个版本的场景管理：SceneManager 和 OctreeSceneManager。SceneManager 只是单纯的实现了场景图，渲染场景则遍历整个场景图并进行适当的拣选，然后渲染。OctreeSceneManager 在 SceneManager 的基础上加上了空间图，这个空间是采用松散八叉树进行描述的。下面分别对场景图和空间图进行介绍，渲染图目前没有一个明确的实现思路（如何对可渲染物品进行分组排序）。

场景图的实现反映在 SceneNode 类中，每个 SceneManager 会默认创建一个根节点，然后可以在该根节点上创建子节点，同时每个节点上可以挂接（attach）一些实体，例如 NPC 等。这样就构建了整个场景。创建完成的整个场景图如下图-1 所示：

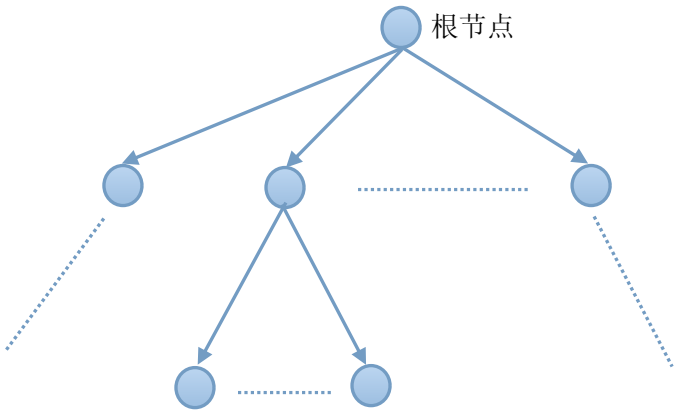


图 1 场景图示意图

场景图中每个节点都有个数据成员表示该节点占据的空间范围，这种数据有很多种表示方法，例如球形包围盒，轴对称包围盒，圆柱包围盒等。每个场景节点的包围盒即为它下面挂接的物体的包围盒，并和自身的位置进行组合得到。可以利用节点的包围盒的信息对场景图进行剪裁，得到需要渲染的场景节点，这是一种蛮力法的剪裁，时间复杂度是 O(n)，其中 n 为场景节点的数量。

空间图的即为了减少裁剪场景需要的时间复杂度而引入的，空间图的表示也有多种表示方法，例如 BSP 树，八叉树，KD 树等。每种表示方法都有适合使用的场景，因此不能简单的说哪个好哪个不好。BSP 树，叫做二分空间树，每次划分空间都是用一个平面将当前的空间一分为二；它比较适合于描述室内的场景，将房子内的各个墙壁作为划分的依据较为合理。八叉树则是采用非常正规的方式来划分空间，每个空间即将划分时总是以该控件的中心为划分点，划分成八个大小相等的子空间；这种空间划分方法比较适合描述室外的场景，即物体比较分散的场景。KD 树和 BSP 树的思路相似，但是它对划分空间的平面进行了一定的限制。

我们在实现中使用了八叉树进行空间的描述，因为这种数据结构的描述较为简单，同时采用了一个分散因子来进行优化，具体原因下面会有介绍。

首先介绍八叉树，在系统中我们使用的是 Octree 数据结构来表示的。每个 Octree 拥有八个孩子结点，这八个孩子的划分如下图-2 所示。

图 2 八叉树空间划分示意图

当向八叉树的根节点（Octree）中加入一个待管理的场景节点（SceneNode）时，如果该节点的空间大小在根节点的空间范围内，则将该根节点划分为八个大小相等的子空间，并递归进行，直到某个子孙节点不能再划分以包围这个节点为止或者整棵树达到一定深度。

当划分节点并递归进行时，可能一个场景节点的空间位置在这个八叉树节点的两个孩子结点之间，这时无论分给谁都不合理。例如下图 3 中，整个四边形是一个空间节点，现在有一个场景节点加入进来，该场景节点中包含了一个五角星的实体。

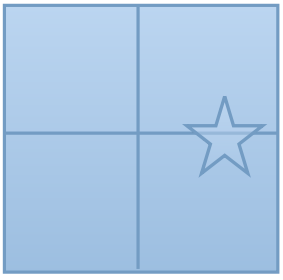

图 3 四叉树空间示意图

由于该五角星大小可以较小，但是没有一个子节点可以包含它，所以上述规则应该将该

场景节点放入根节点中。但是这将导致上层空间节点会包含大量的物体，影响了拣选的效率。还有另外一种解决方案，就是将五角星节点进行分割，分割成两个场景节点，但是对一个三维的物理模型进行分割复杂性太高，得不偿失。

因此引入了松散因子的概念，松散因子就是让每个节点扩大一定的管理范围，每个空间节点可以部分重合，这样当一个物体在两个空间节点之间时就可以明确的将其放入一个孩子结点中。我们的系统中采用的松散因子是 0.5，如下图所示。

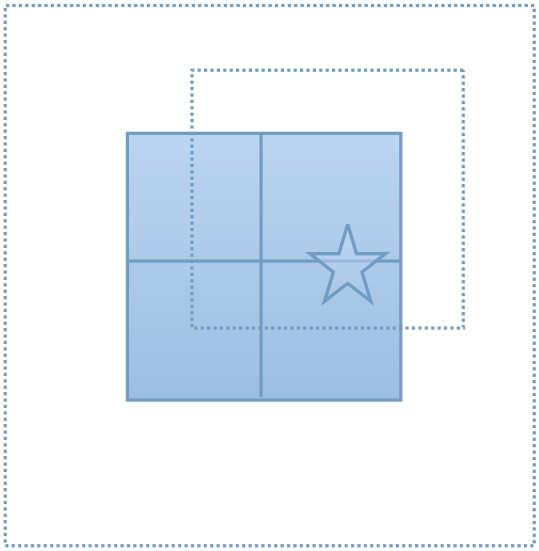

图 4 松散因子示意图

经过松散处理后，每个场景节点的空间范围在每个方向上都变成了原来的两倍。当一个场景节点加入进来到根节点时，根节点就会将该场景节点和根节点的孩子结点松散前的大小进行比较，若孩子结点松散前大小可以容纳，则该场景节点一定在该空间节点松散后的范围内。因此，可以将该场景节点直接划分给还孩子空间节点进行管理。当进行场景拣选时，场景节点使用其松散后的空间范围作为计算因子。

当使用了八叉树进行空间管理后，场景拣选的代价就会大幅减小，从原来的 O(n)减小到 O(lg n)。因为父空间节点的空间范围包括了子节点空间范围，因此若夫空间节点不可见，则无需再对子空间节点进行判断。

代码见 engine/src/octree 部分。