PDF-XCHANGE Click to buy NOW! www.docu-track.com

BSP 技术研究报告

翟永强

2007-10-12

(一)BSP 的相关概念

BSP 用一个平面将多边形场景空间分割成前后两个半空间,然后递归分割所得到的两个半空间,直到不能分割为止.结果是整个场景空间被分割成紧密连接的凸状空间,这些凸状空间要么是封闭的多边形凸包,要么是一个不包含多边形的凸状空间.

注意:尽管一个凸包在分割时有可能一分为二,但是处理的时候是作为两个凸包处理的,就是说一个凸包被一分为二后,缺口要用一个侧面将其堵上的,目的是使后续的分割操作仍然是在一组凸包让继续分割.请不要担心这样会增加多边形,这里只是划分空间而已,等到空间划分成格子后,再将多边形表面分割填入每个格子.

有必要解释一下凸包:准确的定义是凸包的所有定点要么在侧面的线框上,要么在这个侧面之后,决不能出现在侧面之前的情况.或者可以理解为一系列不共面的平面围成一个(注意是一个!)封闭空间.每一个平面都被其他的平面裁切,形成一组共面的点,这些点顺序连接成一个线框,在 QUAKE 系列,(包括 HALF-LIFE) 中,将这个线框叫做"winding".这样讲或许有点难以想象出凸包是什么样子,可以想象一下足球的样子,把足球的每个侧面看成平的,一个平面的前面不会再有其他的点,边,或者侧面.每个侧面以外的顶点和边只能位于侧面之后.所有侧面都是法线向外的.

分割后的所有凸包,是紧密连接的,没有空隙,可以想象一下切西瓜,一刀切成两块,然后两个大块在分别切成小块,这些小块仍然能够无缝的拼合到一起的. BSP 分割场景与此大体相似,不同之处是场景中会有两种空间,一种是封闭多边形包围的 SOLID(实心)空间,一种是封闭多边形之外的空间,分割结果必然是两种空间小块的集合.这些空间小块应该是有序的:从顶层开始,分成两块,就是两个 node,顶层是根 node,然后每个 node 继续分割,最底层的 node 是 leaf_node. 这种分割层次的初衷是 z 排序,已经被现在的 z buffer 技术替代了.另外的意义就是 Hidden surface removal (隐藏面消除),和快速碰撞检测,因为我们能够预先确定出在所有 leaf_node 相互之间的可见性,也就是说当你处在一个空心的(相对于多边形的封闭空间)leaf_node 中的时候,就可以知道哪些实心 leaf_node 是看得到的或者看不到的，然后将看得到的 leaf_node 送入渲染管道，这样来提高实时渲染的效率。对于碰撞检测我们能够确定出移动的两点之间，观察者包围体（通常是一个球体或者盒子）可能会和哪些实心的 leaf_node 发生碰撞，因为这个时候已经锁定了一个很小的监测范围，所以可以显著的提高碰撞检测的效率。关于上面两点，还有一个前提步骤就是如何定位到所在的是哪一个空心的空间，这是很简单的，根据视点位置相对于每个 node 的分割平面进行二分查找，分割的方式到达叶子，就是所在的空间，是非常高效的。

从上面的叙述中可以看到，BSP 的场景实时处理过程很简单，包括消除隐藏面和碰撞检测，所以也很高效。场景的处理重心被移到了离线处理（预处理）过程中了。离线处理通常不会在意几个小时的时间来处理一个场景来换取每帧 33.3ms（30 fps）的渲染效率（这里考虑了一帧中包括场景渲染的所有开销）。问题的关键是在离线处理过程中做好 bsp 层次结构。

PDF-XCHANGE Click to buy NOW! www.docu-track.com

（二）BSP 技术细节

1） 离线处理过程的输入约束：必须是凸包组成的场景，允许凸包和凸包交叉（交叉部分可以消除掉的，消除的方法是处理两遍，后文介绍）。在 quake 系列中，凸包被称作 brush(凸包和刷子有什么关系么？还是不应该理解成刷子？)，凸包的侧面叫做 brush_side.每个 brush _side 在编辑器的输出定义中是三个共面但不共线的点。即便是一个立方体的一个侧面定义也是 3 个点，可见 brush _side 仅仅定义的是所在的平面(仅仅是为了节省空间么？如果定义一个法向量和一个 dist 不也是更节省空间么？)。每个 brush 由一组 brush_side 组成。

Quake3 实现约束的方法是，提供了一个只能添加图多边形的场景编辑器（q3radiant202）. 这就要求有建模功能。现代通常都使用强大的建模工具，可以在 bsp 之前进行预处理，如果有不符合约束的数据，可以处理一下。

2） 分割面如何选取？

主要的选择范围是凸包的所有侧面，共面的侧面使用一个平面。在分割之前，将所有的侧面的三点组成一个 plane 数组，quake3 的做法是添加每个平面时都一正一反两个。这些平面就是分割平面的全部的备选平面。选取的方法是使用的一个评分方案.每次分割一个半空间是的备选平面是从这个半空间中的所有凸包的侧面中选取，使用所选平面对这个个半空间中所有凸包进行切分测试，看看有多少个凸包被切开了，有多少个凸包侧面在屏幕之上，每切开一个凸包减 5 分，每多一个在平面上的侧面就加五分。注意这里没有考虑两侧凸包数目的平衡问题，quake3 的源码中加入了平衡测试，但是后来注掉了，只考虑位于平面上的侧面和切开了的凸包的数目。我进行了一个测试，如果考虑平衡问题的话，会导致切分出很细小的空间来，反而会降低效率。（为什么这样反而会导致分割的过于散碎呢？目前我只能从感觉上感觉出来，也无法定性的解释）。这样，对于待分割的空间中的所有侧面所在的备选平面，每一个都有一个分割评分，评分最高的就是最优分割平面。

在 quake3 中的分割平面的选取范围是每个待分割子空间中的所有侧面。Quake3 还考虑的一个方面，就是每跨过 1k 距离，要强制分割，这样防止出来特别大的空间。这个条件是最优先的。

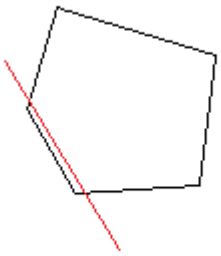
3） 分割过程的结束条件：

选取分割平面的时候是在待分割子空间的侧面中选取的，如果侧面所在的平面已经用过了，就不能再用，按照这个规则，如果所有平面都用过，（跨过 1k 距离的条件比这个条件更优先，就是说当开始选取分割平面的时候，这是一定是没有跨过 1k 距离的）已经没有可测试的分割平面了，这个时候就结束了，这就是 leaf_node.由此也可看到，只要还有没有参加分割的侧面，就会继续用它继续分割，所以每个 leaf_node 的结果是 leaf_node 空间中不存在东西，是中空的。或者是封闭的凸包围空间，或者是什么也没有的凸包围空间

4） 递归的分割过程：在根层次，如果超过 1k 边界，会先进行强制分割的。如果没有就直

PDF-XCHANGE Click to buy NOW! www.docu-track.com

接从全部侧面中选取最优的分割平面，分割平面将所以得凸包分成两组。一分为二后，再重复递归上述过程，递归的再输入就是每一组凸包，每次都从待分割凸包中选取分割平面，跨过 1k 空间强制分割最优先。递归的过程中，每个 node 的只需要记下分割平面。递归的结束条件是待选侧面都被用过了。分割的最后结果是形成了一个 bsp 树，每个 node 只包含分割面的信息，如果是 leaf_node 还会包含是不是封闭的空间，还是空心什么也没有的空间。这是为了做 PVS 而准备的。在 quake3 的 node 中用 opaque 变量来指示是否是透明空间，这里的透明不是指的是包含 alpha 半透明的材质，而是指的是不是能够封闭视线的空间。就是说即便使用了 alpha 材质，如果你不指定它是 opaque 属性，那么包含其中的东西就看不到了，一团漆黑。比如一个透明的罩子里面有一个物体，虽然罩子是透明的，但是里面的东西画不出来。另外分割 node 时，位于两侧的凸体自然就分为到两边了，跨过平面的凸体一分为二，截面用一个追加的侧面将其堵上。问题是什么才叫跨过呢，如果跨过 0.001 个单位算不算跨过？答案是不算，因为这样会分出很薄的叶子空间来。Quake3 的容差是 0.1，如果跨过平面的凸体中某一个侧面跨过了不到 0.1 个单位，就不视为跨过，而是放到大块的一侧。如图所示，红线是分割平面，凸体会被分出一个薄薄的片来，这样是不允许的，只要这个薄片位于平面两侧各 0.1 个单位之内就不算作跨过了，而是直接放到体积大的一侧。



5） 分割场景：请记住，分割空间的过程中之划分空间，尽管中间也会保留分割的凸包组，但是分割完毕后就会丢掉。所以即便分割开的凸体中间追加了一个截面，只是为了继续分割而准备得，全部分割完毕就都不要了。真正的分割场景多边形是在建成 bsp 树以后，然后再用格子空间来裁切所有的面，按照差值方法生成新的点和 uv 坐标。

6） 前面提到过，leaf_node 会纪录它是一个封闭的实心空间还是开放的空心空间，中间切开了的算作两个实心空间。这是为左 PORTAL PVS 而准备的。两个 leaf_node 两两之间无缝连接，连接的部分就是入口（PORTAL），PORTAL 两侧，要么是一个不能穿过视线的侧面（或许是分割了的碎片），要么是中空的空间，只有两侧都是中空空间的 PORTAL 才是视线的一个通路（PASS）。对于一个 LEAF_NODE,找出四周围所有的可以通过 PASS，连成一个链，这就是一个可以 FLOOD 的通路，很形象，如果一个房间中有很多隔段，隔段的底部有洞，那么一倒水，所有的隔段空间中都能被淹到，就是这个意思。淹不到的地方肯定看不到，但是能淹到的地方不一定是看不到的，因为水是四处流动的，而光线或者视线是视线传播的，对于能反射光线的镜子，看作一个前向的 portal，就是说人站在镜子前面，相当于能从视线能从镜子后面的对称位置进入镜框框住的空间。那么，如何判定每个中空的空间能看到那些 leaf_node 呢？当前所在 leaf_node 中周围包围的可以通过的 portal 都是可见的，然后沿着一个 portal，找到另一端连着的叶子，检查沿着入口方向直线传播方向一直能看到那些 PROTAL，那么那个 PORTAL 所在的 LEAF 就是可见

PDF-XCHANGE Click to buy NOW! www.docu-track.com

的。注意检查的范围，是能淹到的 LEAF NODE。将每个空心的 LEAF NODE 中能看到的实心 LEAF NODE 组成一个集合，就是这个 LEAF NODE 的 PVS 信息。每次在观察者找到所在的 LEAF NODE （注意是中空的空间，不可能走到实心空间中的，当你在ＣＳ中死亡后可以走进墙内，会发现有很多黑的，就是这个原因。），然后根据 PVS 测试那些实心ＬＥＡＦ是可见的，然后再经过 FRUSTUM CULLING(视锥裁减) 送入渲染管线。

7) 一个要注意的问题：很多资料上介绍 bsp 时的图示中画的多边形是一个线段，请注意这是一种抽象，实际的是一个包围着的凸壳。一条线段很难理解实心的封闭空间和空心的空间的结合情况，但是图示的东西难以表达三维空间的不同视角，所有要注意这一点。好一点的情况是画一个平面凸多边形，但是仍然难以表达一个封闭的凸包，我们可以看作是一个截面。所以我尽可能用凸包来表达一个封闭凸多边形,而不直接称之为凸多边形,免得总让人忽略掉这个封闭包围的特性.所以很少画图,上面只有一个图示.

7） 以上介绍了 BSP 分割算法的基本方法和要点，与此相关的 PORTAL PVS 没有详细介绍，LIGHT MAP 也没有介绍，BSP 分割是 PVS 计算和 LIGHTMAP 计算的基础。（完）