计算机动画的算法与技术——基于 GPU 的碰撞检测算法

田正祺　2020080095　软件 01

1 背景

碰撞检测通常分为宽相位碰撞检测（broad phase collision detection）和窄相位碰撞检测（narrow phase collision detection）[1]。宽相位碰撞检测用于快速粗略地将完全没法碰撞的物体去除掉，然后对可能碰撞的物体进行更精确的但通常更慢的窄相位碰撞检测。由于预期仿真的物体（球，正/长方体，四面体）比较简单，此项目会使用简单的窄相位碰撞检测算法，而聚焦在宽相位碰撞检测。

2 算法概述

朴素的算法会将 n 个物体中的每一个物体与其他 $n-1$ 个物体进行碰撞检测，其时间复杂度为 $O(n^2)$。对于几万个物体，这个算法的速度不够快。

为了降低时间复杂度，此项目使用的数据结构是层级包围体树。一个包围体必需包含它的物体的所有点。为了加快计算，通常会使用最小轴对齐包围盒（minimum axis-aligned bounding box），即最小的，各个边与三个坐标轴平行的长方体。包围体可以形成树结构，约束条件是叶节点必须是物体本身，而所有其他节点都是包围体，并且任意一个内部节点必须是它的子节点的包围盒。在便利层级包围体树时，若一个物体与某一个节点表示的包围体没有碰撞，则物体不会与此包围体的子包围体/物体碰撞，因此可以跳过子节点。对于理想构建的树而言，时间复杂度可以降低到 $O(n \log n)$。

以下会讨论算法的实现细节以及可以利用 GPU 加速的地方。

3 树的构建

此项目的碰撞检测对象是动态的物体，即每次物体的位置更新后需要更新层级包围体树并做一次碰撞检测。目前存高效的算法，可以一次性构建一棵树并每次更新同一棵树[2]。但是考虑在此场景中，物体的位置可能会有巨大的变化，因此选择每次进行碰撞检测时重新构建全新的树。

此项目采用了线性层级包围体算法[3]，将所有物体排序，并划分区间，将同一个区间的物体放在同一个包围体中。由于更小的包围体能够更精确地拟合物体，则目标是将位置类似的物体放在同一个包围体中。将三维位置信息映射到一唯空间并且保留位置的局部性，可以使用 Z 阶曲线（Z-order curve，也称 Morton order/code）[4]。一个三维位置的 Z 值是位置的三个值的二进制表示中每个比特的交错。比如：

```
x = 0.00101010 -> 0.0  0  1  0  1  0  1  0
y = 0.10100101 -> 0. 1  0  1  0  0  1  0  1
z = 0.10101010 -> 0.  1  0  1  0  1  0  1  0
     Z(x, y, z) = 0.011000111000101010101010
```

计算到 Z 值后需要进行排序，而排序是一个可并行化的操作。TODO

排序后，根节点就覆盖 $[0, n-1]$ 区间内的所有物体，而它的左子树和右子树覆盖的区间分别为 $[0, i]$ 和 $[i+1, n-1]$，其中 i 是某一个合适的分割点。这样递归地计算区间，直到区间内只有一个物体，它就成为叶节点。

包围体的是自底向上计算的。叶节点的包围体是物体的包围体，而内部节点的包围体是它的两个子节点的包围体的并。如果使用包围盒，可以用长方体的对角线上的两个点表示。为了简化计算，可以取 x, y, z 分别最小和分别最大的两个角。在 GPU 上可以为每个叶节点创建一个线程，计算完了一个节点的包围体就计算父节点的包围体。为了避免重复地计算以及保证两个子节点的包围体也计算完毕，可以使用一个原子标志，使得第一个处理此节点的线程直接返回，而让第二个处理此节点的线程正真进行计算。

4 树的遍历

参考文献

[1] Thinking Parallel, Part I: Collision Detection on the GPU | NVIDIA Technical Blog — developer.nvidia.com[EB/OL]. https://developer.nvidia.com/blog/thinking-parallel-part-i-collision-detection-gpu/.

[2] WALD I, IZE T, PARKER S G. Fast, parallel, and asynchronous construction of bvhs for ray tracing animated scenes[J/OL]. Computers &; Graphics, 2008, 32(1): 3–13. http://dx.doi.org/10.1016/j.cag.2007.11.004.

[3] LAUTERBACH C, GARLAND M, SENGUPTA S, et al. Fast bvh construction on gpus[J/OL]. Computer Graphics Forum, 2009, 28(2): 375–384. http://dx.doi.org/10.1111/j.1467-8659.2009.01377.x.

[4] Z-order curve - Wikipedia — en.wikipedia.org[EB/OL]. https://en.wikipedia.org/wiki/Z-order_curve.

[5] Thinking Parallel, Part II: Tree Traversal on the GPU | NVIDIA Technical Blog — developer.nvidia.com[EB/OL]. https://developer.nvidia.com/blog/thinking-parallel-part-ii-tree-traversal-gpu/.

[6] Thinking Parallel, Part III: Tree Construction on the GPU | NVIDIA Technical Blog — developer.nvidia.com[EB/OL]. https://developer.nvidia.com/blog/thinking-parallel-part-iii-tree-construction-gpu/.