计算几何实验报告

张维（024920） 李行武（024939） 戴侃斐（024924）

问题背景：

碰撞检测是交互场景模拟的瓶颈问题之一。对物体进行层次化表示的预处理，是进行碰撞检测经常用到的处理方法。

任何利用层次结构进行碰撞检测的算法必须满足以下三个要求：

- 层次结构对于物体的近似是一种保守近似，必须对于物体进行完全覆盖。
- 层次结构中，低一层（子节点层）对于物体近似的精确程度要比上一层的结点（父节点层）要高。
- 对于层次结构中的每一个节点所覆盖的物体，都必须被这个节点的子节点完全覆盖。

现有的层次表示法利用不同的基本几何形体对场景中的物体进行近似构造：sphere, bounding box等。一般而言，对于用于进行近似构造的几何形体而言，都存在着效率和精确性之间的折衷。对于简单的几何形体：sphere和bounding box等，虽然可以用很小的计算代价进行碰撞检测，但是在对于物体构造的近似程度（正确性）方面存在着不足。但是对于交互场景而言，时间上的限制使得碰撞检测都是在一定误差范围之内进行的，所以可以利用简单几何形体构造的近似物体在交互场景中进行碰撞检测计算。

同其他几何形体相比，利用sphere 进行层次结构的构造有以下几个方面的优点：

1. 碰撞检测的低计算代价。
2. 对物体的近似构造具有较高的正确性(尤其在利用adaptive generation算法以后)，利用较低层数的sphere tree可以达到对物体近似的很高程度。

利用sphere进行层次构造所得到的数据结构成为sphere tree。

算法及原理：

1、算法思想：

用Adaptive Medial Axis Approximation的方法进行sphere tree的构造。

1) 对输入的Object表面进行均匀、密集的采样。

2) 构造初始的sphere tree的根结点：利用求最小包围球的算法，得到最初的sphere，并建立对应的4个sites的3D voronoi diagram。

3) 算法的递归过程：

- Adaptive algorithm：更新voronoi diagram，使得每次新加入的sites都是原来采样点中距球面距离最远的点，并生成对应的球心在vertex地球体。重复进行此过程直到sphere的个数达到规定的阈值，在该过程中声称的sphere 称之为medial sphere；
- Sphere reducer：对于上述生成的medial sphere set进行归并，消除冗余的spheres，使得生成的spheres的个数不超过设定的 treeBranchFactor （最大子节点个数）；
- Optimize processing：对于经过归并以后的spheres set，进行优化处理，进一步减小生成的sphere的误差；
- Insert into sphere tree：将经过优化处理的spheres加入到sphere tree中。

2、 核心算法的具体描述：

1) 3-D voronoi diagram construction algorithm:

处理在原有的 voronoi diagram 中加入新的 site 时，对于 voronoi diagram 的更新。

2) 球的误差（m_fError）的计算公式：

假如球心c在物体表面的外面，则：

$$d = \sqrt{r^2 - d_2^2} + d_1$$ （r，球的半径；d2，采样点到球心在表面投影c’的距离；d1，c到c’的距离）

假如球心c在物体表面的里面，则：

$$d = \sqrt{r^2 - d_2^2} - d_1$$

所得到的球的误差就是球所包含的所有的采样点的最大误差d。

3) Sphere reduction algorithm（Merge or Expand）：

主要是用来进行对于冗余的sphere的剔除，主要利用归并或者贪心算法，生成最少数目的sphere sets来覆盖原来的object。并且使得最后生成的sphere的差错率在指定的范围内。

Merge算法：在循环中，先构造出所有的能够合并的球的对（mergers）。在reduce过程中，每一次挑选出合并代价最小的一对包围球进行合并，同时重新计算球的误差，并且更新他们的邻接关系，直到球的数目不大于 treeBranchFactor。

Expand算法：在循环中，每次根据Max Uncovered Area 或者 Max Reduction Area 原则

选出一个球，并适当扩大球的大小，将这个球加入到reduced sphere set中，然后继续选择新的球，直到所有的采样点都被包括。

4) Sphere optimize algorithm：

Optimize算法：

定义sphere set的总误差TotalError = WORST_WEIGHT×单个小球最大误差＋RMS_WEIGHT×所有小球均方差，其中WORST_WEIGHT和RMS_WEIGHT是事先定义好的权重。

该算法的目标是对于sphere reduction algorithm产生的sphere set进行优化计算，使得最终sphere set的总误差最小且每个小球所覆盖的区域尽可能的平均。

首先对于sphere set中的每一个小球，确定其所包含的物体的区域（即所包含的采样点的集合），要求小球所包含的采样点不能重复以避免将来重复计算。

接着使用阿米巴算法（无约束条件的非线性优化算法）依次（从误差最大的开始）调整各个小球直到总误差达到最小。

Balance算法：

每一层的小球数量越少，则生成下一层小球的工作量以及将来碰撞检测时的计算量也会越小。但是小球数量减少时，误差通常也会增加。

该算法的目标是平衡上面两者的关系，即在保证误差在某个范围内的情况下尽可能减少小球个数。

给定小球个数，首先计算此时允许的最大误差。然后调用前面的Reduce算法将小球个数减少到给定的数目，接着使用上面的Optimize算法优化小球集合。

如果优化后的总误差满足条件则继续减少小球个数并重复上述过程，否则停止。

系统设计：

系统采用 Microsoft Visual C++ 7.0 进行设计和开发，全部程序在 Microsoft Windows XP Professional、256MB RAM 下调试通过。

- 系统流程（如图 1 所示）：

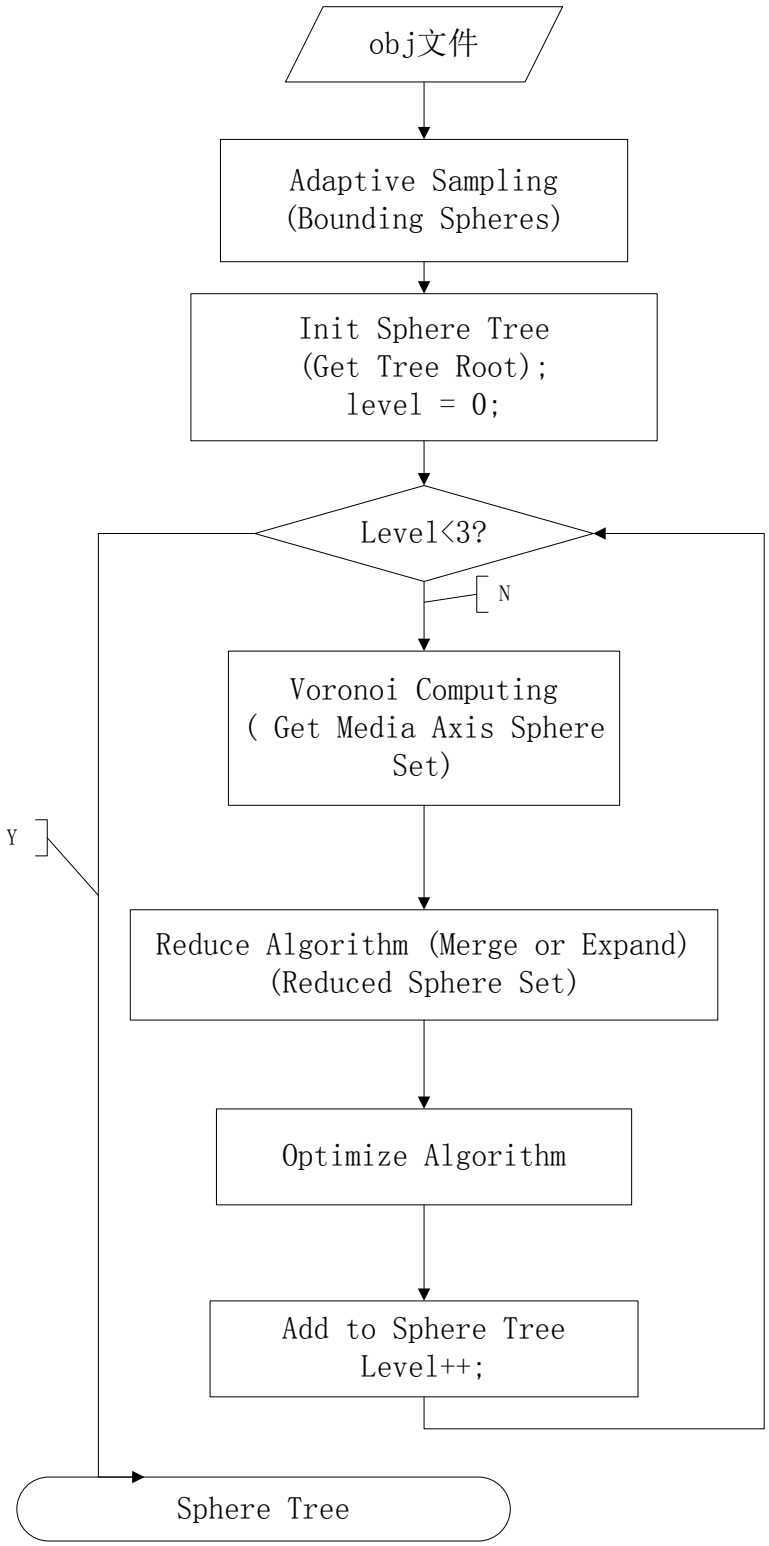

图 1

- 整个系统中所用的主要的数据结构：

Model = { // 物体数据

CGAL Polyhedron_3; (DCEL)

```
}
Point array= {                          // 采样点的数据结构
    CGAL Vector<Point_3>;
}
Sphere = {                              // 球的数据结构
    int    m_nIndex;                    // 球的索引
    Point_3 m_Center;                   // 球心
    double     m_fRadius;               // 球的半径
    double     m_fErr;                  // 误差
    Vector<int> m_PtsIndex;             // 球所包含的采样点的索引
}
Medial Sphere : public Sphere = {
    double  m_fError;                   // 误差
    vector<int> m_PtsIndex;             // 采样点索引
}
Sphere Tree = {                         // 生成的树
    Vector<Sphere>;
}
Sphere Set = {
    Vector<Medial Sphere>
}
struct MergeSphere{                     //merge 过程中用到的球的数据结构
        bool valid;                     // 判断球是否已经被 merge 过的标志
        Sphere s;                       //
        vector<int> neighbours;         // 邻接球的索引表
        double     m_fError;            // 球的误差
        vector<int> m_PtsIndex;         // 球所包含的点的索引
    };
```

实验结果：

实现了参考文献中提到的 adaptive medial axis approximation，并用 merge 和 expand 两种方法实现了 sphere-tree 的构造。并实现了 Banlance 和 Simplex 优化算法。

- 实验效果：

系统主界面如图 2 所示：

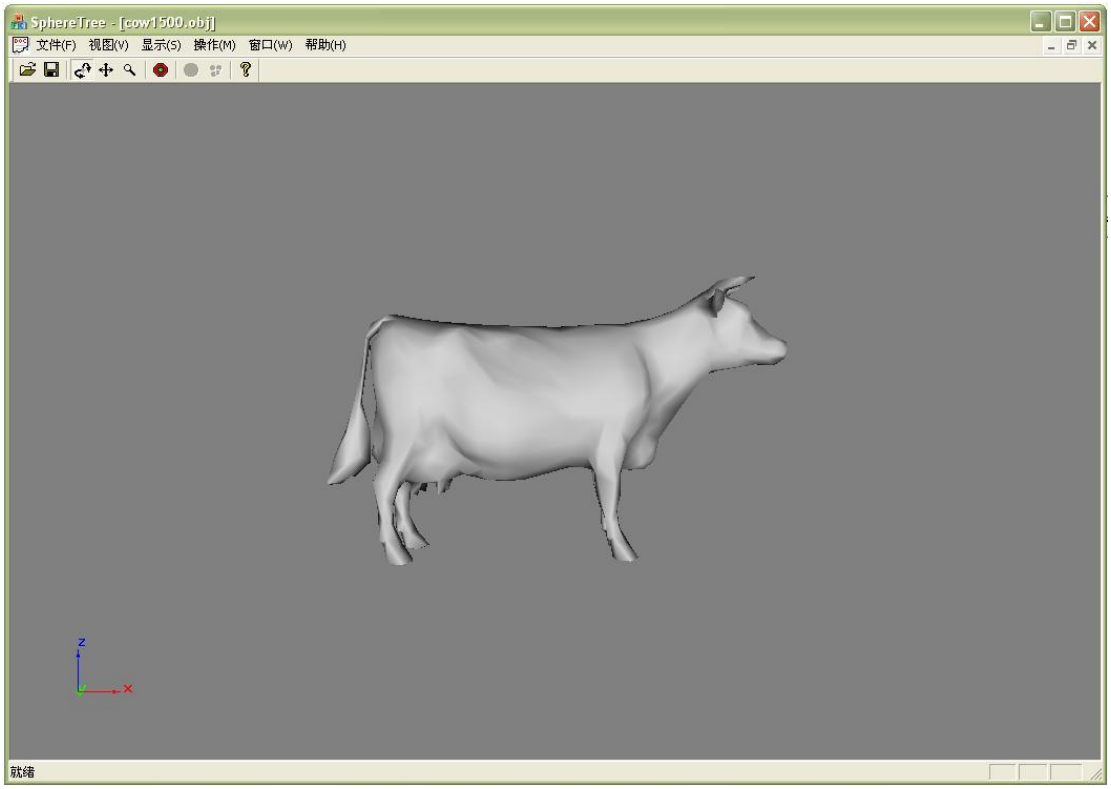

图 2

所生成的 sphere tree 如图 3 所示：（用的是 cow1500.obj 的例子，5000 个采样点，整个运行时间大约是 3 分钟。）

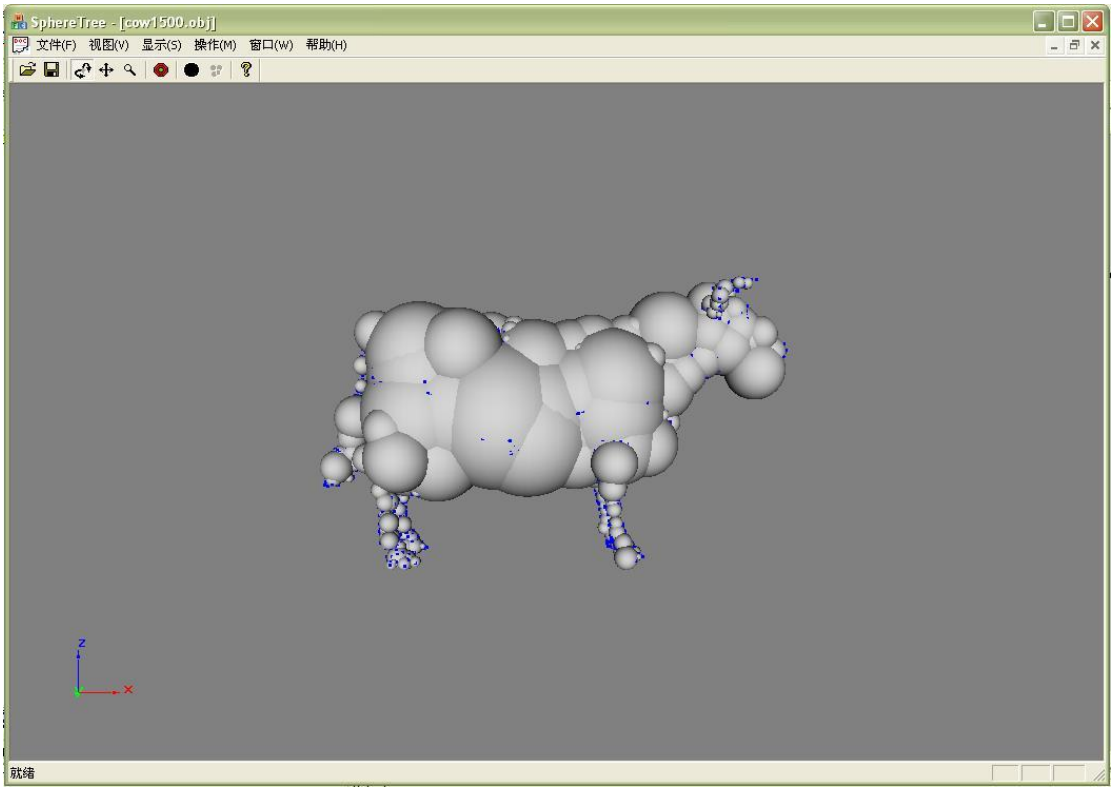

图 3

- 测试以及对比结果：

1、我们调用了 CGAL 库中的一些函数，[G. Bradshaw 2003]中的所有函数功能都是自己用 VC 实现的。

2、在输入一致的情况下，整个算法的速度明显由于原作者的实现结果，例如对于 cow1500.obj，在计算三层 SphereTree（最大 8 分支）时，原实现需要 15～20 分钟，我们的实现只需要两分钟。但是我们的实现结果明显比原实现要差，而且层数越高时，差距越明显。由此推测原实现中采用了大量的细节优化措施。

问题以及解决方法：

1. 对于输入物体表面的均匀采样的算法。参考文献并没有给出一个具体的实现算法，在我们的实现过程中，我们利用随机生成的采样点集去代替原先算法中的均匀采样点集。
2. 在确定 object segmentation 时，如何确定每个 parent sphere 的区域，这个区域对于和其他的 parent sphere 所确定的区域的重叠最小，即如何设计对 parent sphere set 中重复覆盖的区域进行剖分的算法（行武添加）。
3. 对于该算法中所用到的voronoi diagram，实际上只用到了其中的vertex，如何设计数据结

构使得对于vertex快速的遍历也是要解决的难点之一。在实现的过程中，我们利用了现有CGAL中的3DTriangulation数据结构，这个数据结构最大的特点是对于3D DelaunayTriangluation的支持，而且很好地支持了对于Vertex和Site地遍历操作。

4. 对sphere reduction algorithm产生的sphere set进行优化计算的方法，参考文献并没有给出具体的实现算法，而且这一部分的处理时间在整个算法中所占比例较大，是否存在一个较好的解决方案是值得研究的一个问题。（行武和侃斐能不能加一些东西）？

尚未解决的问题：

1. SphereTree Generation 算法的一个重要应用，表现在复杂物体进行碰撞检测时输入模型的简化处理。由于实现时间的限制，我们并没有实现我们原先的开题报告中所提出的第二部分的内容，即实现利用 SphereTree 算法的进行实时碰撞检测系统。
2. 在已经实现的系统中，程序运行所耗的大部分时间在于 sphere set 的优化过程，对于 sphere set 的优化无疑会优化程序的执行。

结束语：

碰撞检测中存在许多值得研究的问题，而且许多都是有现实意义的。

可以看到我们所完成的程序在许多方面有值得改进的地发。比如说，程序的运行速度还是比较慢。

在整个的作业过程中，我们学到最多的是团队合作精神，分工的思想，问题在讨论中产生，并在讨论中得到解决。

我们要感谢邓俊辉老师对我们的帮助，在计算几何课程中，我们学到了很多，不仅是几个概念，还有很多算法的思想，以及多种角度思考问题的模式。而在整个的作业过程中，邓老师对我们的指点也使我们受益匪浅。

代码来源：

算法参考了[G. Bradshaw 2003]中的源代码（http://isg.cs.tcd.ie/spheretree）。

算法中使用了 CGAL 函数库。

整体框架和核心的算法实现都是我们自己完成的。

参考文献：

[G. Bradshaw 2003] G. Gradshaw, Carol O'Sullivan.
Adaptive Medial-Axis Approximation for Sphere-Tree Construction, ACM Transaction on Graphics, to be appeared on 2004.01

[G. Bradshaw 2002] G. Gradshaw, Carol O'Sullivan.
Sphere-Tree construction using adaptive medial axis approximation,
Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, July 2002.

[CGAL 2002] CGAL Developers' Manual Release 2.3, November 2003

[CGAL 2003] CGAL Basic Library Release 3.0, November 2003