第四章 整体渲染框架的实现

4.1 概述

此部分内容是本文的后处理系统实现的核心。本文在第二章和第三章中提到了两种最主要的效果的渲染算法，即粒子表示的模型的表面重构和粒子表示的碎片云等介质体效果的渲染方法。然而，对于一套后处理渲染系统来说，光有每一部分的实现算法是不够的。该后处理系统的基本目标是，用户读取仿真结果的数据文件，并进行简单的用户参数设置（如光照信息，模型位置，观察视角等）便可自动地生成渲染结果图形。

本系统主要针对清华大学航天航空学院计算动力学研究室的物质点法计算软件MPM3DPP的仿真计算结果,其仿真结果的数据文件为VTK非结构网格数据类型（.vtu文件）。该文件结构只存储顶点信息和与各顶点相关联的单元信息。对于物质点法数据结果，一个粒子对应一个顶点和一个单元，其中每个粒子携带物质的各种物理量，如位置，速度，失效信息，能量，温度等等信息。对于有限元法结果，一个单元对应8个顶点，为六面体单元。在所有数据均为粒子信息的情况下，我们并不能确定哪一部分粒子代表固体或液体，需要表面重构，哪一部分粒子代表碎片云，需要使用介质体的体渲染算法，所以我们需要一个自动归类算法，将不同物质的粒子分开；当对模型进行缩放时，要保证渲染效果相同，还需要对模型进行预处理；而当各类物质有交叠时，还需要实现正确的深度、阴影等效果，这需要一个正确的渲染框架。

4.2模型预处理及自动的粒子归类算法

对于使用SPH、MPM等无网格法仿真的冲击爆炸类问题，其所有的物质都是使用粒子来表示的，不同粒子并没有本质上的差别，也就是说粒子不会有其属于整块物质还是碎片云的信息，这也正是此类方法用来处理冲击爆炸类问题的优势。

对于MPM3DPP的计算结果数据，可以选择输出粒子的”fail”属性和”damg”属性，”fail”属性标示该粒子是否失效，是一个0或1的布尔量，”damg”属性是一个[0,1]区间内的实数量，标示该粒子对应物质的损坏程度，0为完全没有损坏，1为全部损坏。这里我选取fail=0以及damg<0.5的粒子为总体非失效粒子，其余为总体失效粒子。如果粒子非失效，则可判断其一定属于相连的物质块，需要进行表面重构；而对于失效粒子，其可能为碎片云中被彻底粉碎的部分，也可能为被击碎飞溅而出的物质小颗粒，而后者也是需要被表面重构的部分。

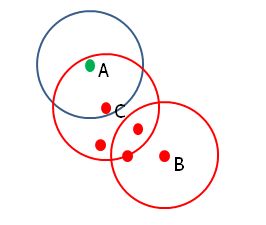
如图【4-1】-a，板中间红色的粒子为失效的粒子，但是其依然附着在板上，并未飞溅而出，此部分显然也应该去重构表面；图【4-1】-b是将所有失效粒子当做介质体渲染的结果，可以明显的看到其浓度过高，和周围的碎片云极不协调。

基于此，本文提出了一种基于对模型求粒子半径的预处理算法来实现自动归类。

首先，将读入的粒子数据组织为一颗平衡的k-d树，然后对每个粒子使用KNN算法求其3个最近邻粒子，将这三个粒子到的距离平均，记为的半径；

其次，将所有的非失效粒子的半径进行平均，得到，即为第二章中提到的长度量纲参考长度。

为了找到失效粒子中需要被重构的粒子，对每一个失效粒子使用KNN算法在的半径内搜索2个最近邻粒子，如果搜索到2个结果，则认定该粒子和周围粒子“连接”，将其归类为需要重构表面的粒子；若没有搜索到粒子或仅搜索到1个结果，则认为该粒子和周围粒子“脱离”，归类为碎片云粒子。如图【】



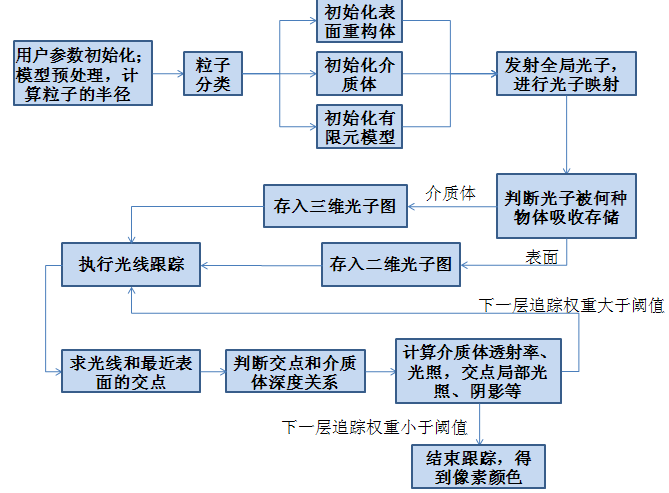
图中所搜半径（红色和蓝色的圆）均为，其中红的的粒子均能在该范围内搜索到2个及以上的邻近粒子，这些粒子连接比较紧凑，可以认为是“一整块”，且其平均距离小于，可以认为该小块的密度和非失效粒子所代表的整块物体接近，所以将其归为一类，认为是需要重构表面皿的部分。对于为什么要搜索2个最近邻，可以看图中的A粒子，虽然A和C的距离在以内，但是和其它粒子的距离均较远，这样导致A粒子和红色粒子组成的紧凑的“块”比较孤立，只有线段AC这一个连接，这样的连接不足以认定它和该“块”真的是一个整体，所以归为碎片云一类。

图【】为使用该算法分离的一个例子

左图为原始的全部粒子，中图为分离出的需要重构表面部分的粒子，右图为碎片云部分粒子，下图为三种情况下渲染出来的结果，可见分离结果还是精度很高的，已经没有了图【4-1】-b中密度过高的“浓烟核”效果。需要指出的是，从渲染结果看，并非所有分离出来的需要表面重构的颗粒都被渲染了，这就是第二章中讨论过的较大光线步进长度导致的微小结构捕捉不到的问题，有待今后的工作完善此内容。

4.3 基于光跟踪算法的渲染框架设计

整个渲染过程的流程图如下：



该系统主要分为三层功能实现：

第一层为全局初始化，包括用户参数初始化，有渲染区域的定义、视点和投影方向的定义、光源的定义、模型材质库的读取以及模型的初始化，有模型的加载及预处理、粒子的分类、分类后各渲染模块的初始化。

第二层为全局光照计算，也即光子映射阶段。该过程向整个场景区域投射光子，进行光子追踪，将不同类型的光子存储到不同的光子图中。

第三层为光跟踪渲染阶段，计算光照，生成最终的图形。该阶段同时动态地进行表面重构工作，表面重构以后数据被保存，渲染变换视角后的后续帧图形时不必重新计算。当改换视角后，只需进行第三层操作，只有当模型发生变化时才需要重新进行一、二层的工作。

* 1. 几个重要功能的实现
     1. 光线和不同类型物体交点的计算次序

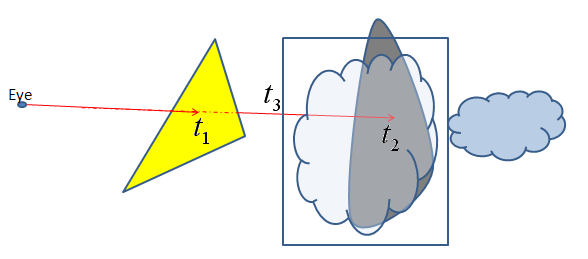
由于光线和不同类型的物体交点计算的方法不同，所以无法通过一个统一的数据结构找到距离视点最近的交点；而如果对每种物体都计算一个和当前追踪光线的交点然后再比较深度，这样会造成大量的不必要的计算，影响效率。

从交点的计算花销大小和交点类型的关系，计算应该遵从如下顺序：

先计算光线和场景物体以及三角形面片构成的模型的交点，得到深度值

再计算表面重构体的交点，当计算开始时，记每一步光线步进的光线前端深度值为，当时停止步进，因为这时表面重构体已经被三角面片或简单几何体遮挡，无需计算重构表面及其交点，若在之前遇到需要重构表面的粒子，则进行重构表面操作，并计算光线和重构表面的交点，记光线前端深度值为，更新

最后判断介质体和的深度关系，记介质体的入口深度为，若则证明介质体被表面全部遮挡，无需进行光线步进操作计算介质体的光照；若，则只需要计算到这一段的介质体光照。过程如图【】所示。



如果黄色三角面在最前方，则后面的所有交点都无需计算；如果没有黄色三角形，则先计算得到，再计算到这一段介质体的光照。

* + 1. 阴影

在计算一点的局部光照时，以光源点为视点进行4.4.1所描述的操作寻找交点，如果在光源和之间存在交点，则认为光线被遮挡，判定在阴影区域内；此时根据交点处的透明度（或者是介质体对光线的衰减）来计算的光照。

* + 1. 光线和透明物体表面相交是界面两侧折射率的判断

在场景中存在透明物体时，光照的计算一般较为复杂，其中的一个问题就是判断光线和透明物体表面相交是界面两侧折射率。因为光线在发生折射时，只与物体的表面发生作用，而物体的表面（可以是三角面片，也可以是重构的表面）只存储该物质对应的材质的折射率，而没有界面两侧的信息。于是本文提出了一种基于折射率栈结构的折射率记录算法，可以处理非常复杂的界面信息，包括多层不同折射率的透明物体嵌套，比如水中的气泡这样的效果。

我们设计一个栈，使得光线在射入一个表面时，将该表面的材质的折射率压入栈中，而在光线射出一个表面时，将栈顶的折射率弹出，其中判断光线是射入还是射出表面可以根据表面的法向量和光线的方向之点积的正负来判断。按照如此过程执行，则栈顶元素永远存储光线前端点所在空间区域对应的物体的折射率，于是在光线穿越界面时，栈顶元素发生变化，旧的栈顶元素和新的栈顶元素分别记录了界面两侧物质的折射率。

在开始渲染工作之前，需要先初始化该折射率栈，实质上是判断视点所在位置是在几层透明体的包围之内。首先将空气折射率压入栈中，然后任意选取空气中的一点为起始点，向视点进行光线跟踪操作，沿路遇到表面时进行压栈或弹出操作，直到光线到达视点，此时的折射率栈即为每条从视点开始的跟踪光线的初始折射率栈。

4.5系统各模块的设计