中期报告

算法设计

碰撞检测算法可分为两个阶段broad phase和narrow phase。broad phase进行保守检测，只计算物体的包围盒体积，筛选出可能碰撞的物体对。在可能碰撞的物体对的集合中进行narrow phase，精确计算碰撞点和法线。

https://developer.nvidia.com/gpugems/gpugems3/part-v-physics-simulation/chapter-32-broad-phase-collision-detection-cuda提供了broad phase的sort and sweep算法、spatial subvision算法和并行的spatial subvision算法。sort and sweep是类似AABB碰撞检测的基于包围盒的算法。spatial subvision将空间划分为多个grid，每个grid至少容纳一个物体，质心位于某个grid的物体会被归进该grid的物体列表。当两个对象同属于一个单元格或相邻单元格时，执行碰撞检测。

对于本次大作业，将空间划分为一个个边长与最大球体直径相等的grid，两球只会在同属一个grid或相邻grid时发生碰撞。

简单的实现是为空间中每个grid进行编号，然后对每个球体使用hash算法将其分到每个grid的桶中，然后对hash值进行排序。排序后，必然能得到一串数值，由一段段连续的相同的值构成。拥有相同的值的一段加上其前一段和后一段，就是需要碰撞检测的球体。精确的碰撞检测只需比较两球的球心距和半径之和的大小即可。

关于复杂度，对于n个球体，hashing过程需要O(n)，排序使用基数排序或桶排序能在O(nd)或O(n+k)完成，遍历排序后的数据需要O(n)，碰撞检测也需要O(n)。综合来看，最终算法时间复杂度在O(n)。

Spital Subvision的CUDA实现

质心所在的cell称为H cell，球体所覆盖但质心不在的cell称为P cell。

初始化，构建两个数组存储球体id和cell id。对每个球体进行hashing，得到的hash值存在cell id数组中作为每个物体的第一个cell id，该cell也是物体的H cell，同时更新球体id数组。接下来计算每个球体的P cell并将其id更新到cell id数组中。理论上来说，每个球体最多可以拥有8个cell id，其中一个是H cell，其余的为P cell。这部分内容可以通过多线程实现。

对数组内容进行排序。由于希望排序后H cell在P cell前面，因此需要采用一种稳定排序，这里选择基数排序。对排序好后的cell id数组进行碰撞检测即可。

根据CUDA教程，将碰撞检测部分放至gpu实现，用类C++形式编写即可。

GPU设计的关键在于充分利用多线程，合理地分配任务给每个线程，才能充分发挥GPU性能。

参考文献

https://docs.nvidia.com/cuda/

https://developer.nvidia.com/gpugems/gpugems3/part-v-physics-simulation/chapter-32-broad-phase-collision-detection-cuda