三维运动的碰撞处理

目录

- 碰撞处理
- 离散/连续碰撞检测
- 并行碰撞检测
- 物理仿真中的碰撞处理
- 开源碰撞处理平台

碰撞处理

有限元仿真

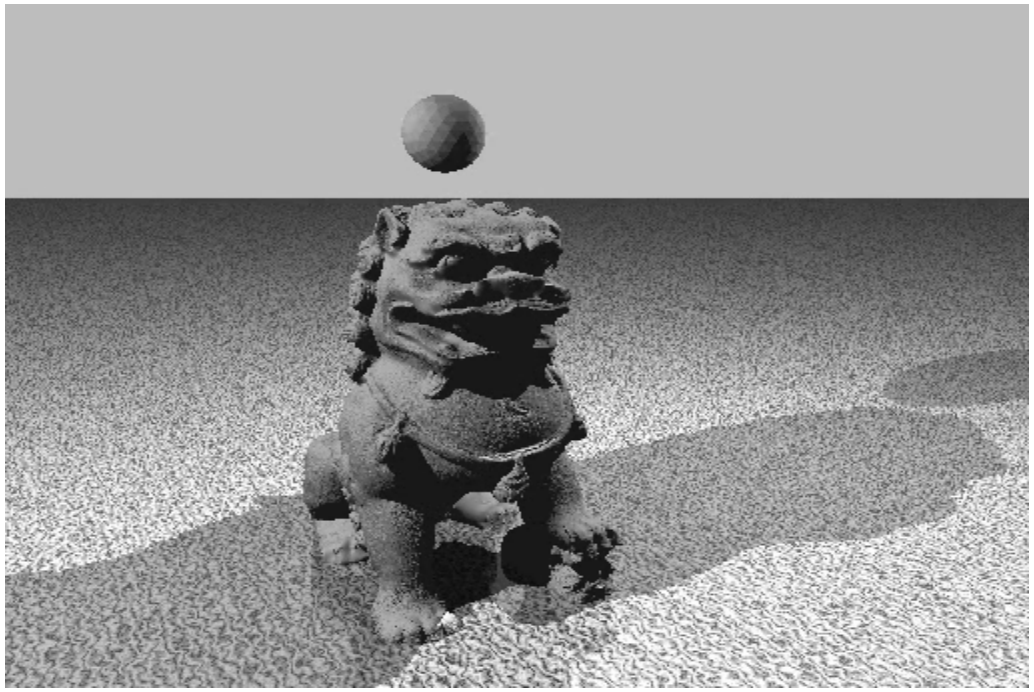
碰撞处理

毛发仿真

碰撞处理

服装仿真

碰撞处理



破碎仿真

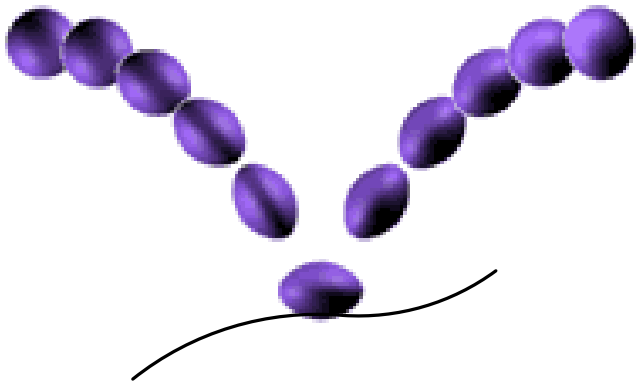
以上所有三维物理仿真场景都需要快速可靠的碰撞处理

碰撞处理

- 包含以下两个阶段：
  - 碰撞检测：找出所有发生的穿透或靠近情况
  - 碰撞响应：解决碰撞，得到一个无穿透状态

碰撞检测

- 碰撞检测对以下应用至关重要：
  - 物理仿真
  - 视频游戏
  - CAD/CAM
  - 机器人、触觉渲染
  - 计算机动画
  - …

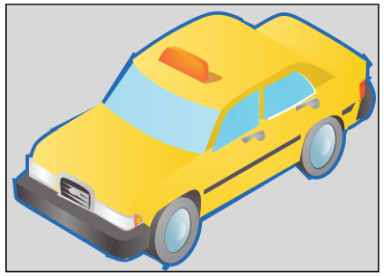


- 通常需要计算出**第一接触时间和位置**

包围盒剔除

- Sphere/AABB/OBB/k-DOP

Sphere



AABB

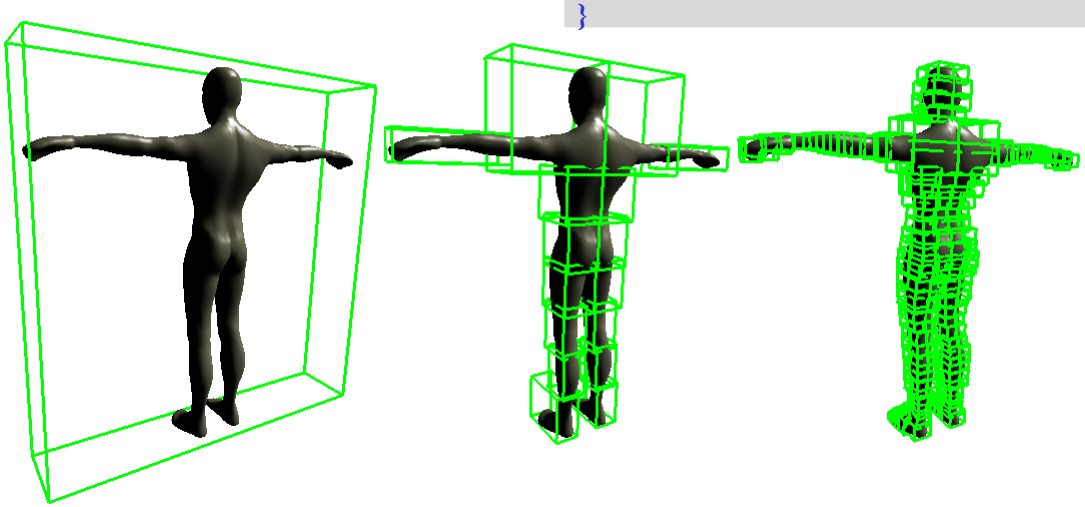
OBB

k-DOP

包围盒层次结构BVH

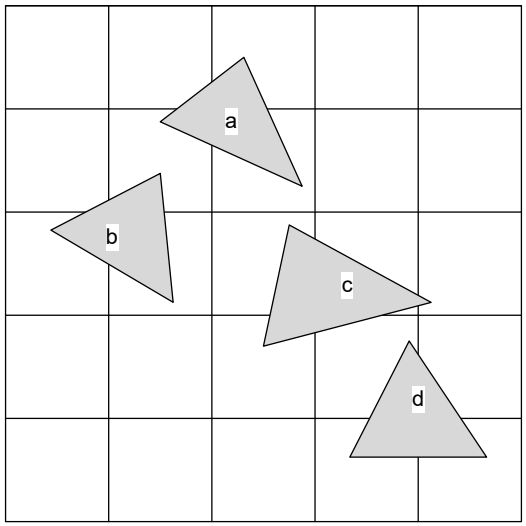
- 基于BVH的碰撞检测

```
void
DeformBVHNode::collide(DeformBVHNode *other,
std::vector<triangle_pair> &tri_list)
{
   //保存叶节点上的三角形对
   if (isLeaf() && other->isLeaf()) {
      tri_list.push_back(triangle_pair(getTriID(), other->getTriID()));
      return;
   }
    // 检测包围盒是否重叠
   if (!_box.overlaps(other->_box)) {return;}
   //递归遍历子树
   if (isLeaf()) {collide(other->getLeftChild(), tri_list);
                  collide(other->getRightChild(),tri_list);}
   else {         getLeftChild()->collide(other, tri_list);
                  getRightChild()->collide(other, tri_list);}
}
```



空间哈希

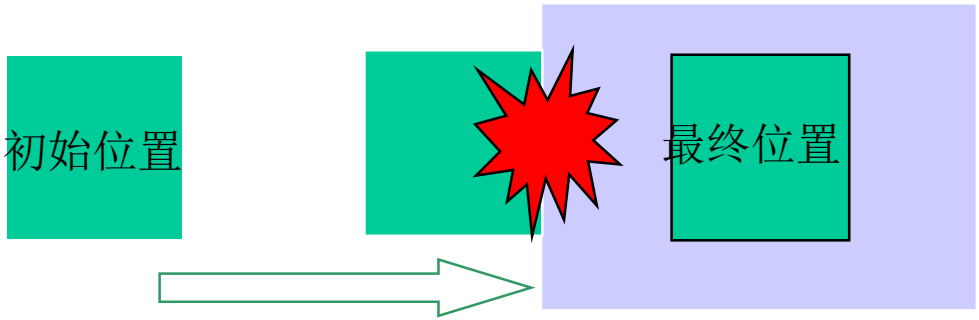
- 基于空间哈希的碰撞检测




```
// 对所有的三角形进行遍历
for (int i=0; i<triNum; i++) {
                AABB bx = tri[i].bound();

                //找到三角形包围盒占用的空间哈希格子
                std::vector<hash_cell *> cells;
                getAABBcells(bx, cells);

                //在这些格子上记录下该三角形
                for (int j=0; j<cells.size(); j++) {
                                cells[j]->record(i);
                }
}

//对空间哈希的所有格子进行遍历
for (int i=0; i<cellNum; i++) {
                hash_cell *c = allCells[i];
                if (c->triNum() == 0)
                                continue;

//对所有落在同一格子上的三角形进行碰撞测试
                test_triangle_contact(c->triangles());
}
```

离散/连续碰撞检测

- 连续碰撞检测(CCD)
  - 沿着连续的变形轨迹进行测试
  - 15次单元测试 (VF和EE连续测试)
- 离散碰撞检测(DCD)
  - 只在离散时间点进行测试
  - 6次边/面相交测试
  - 可能漏掉碰撞或者返回错误的结果！



CCD计算

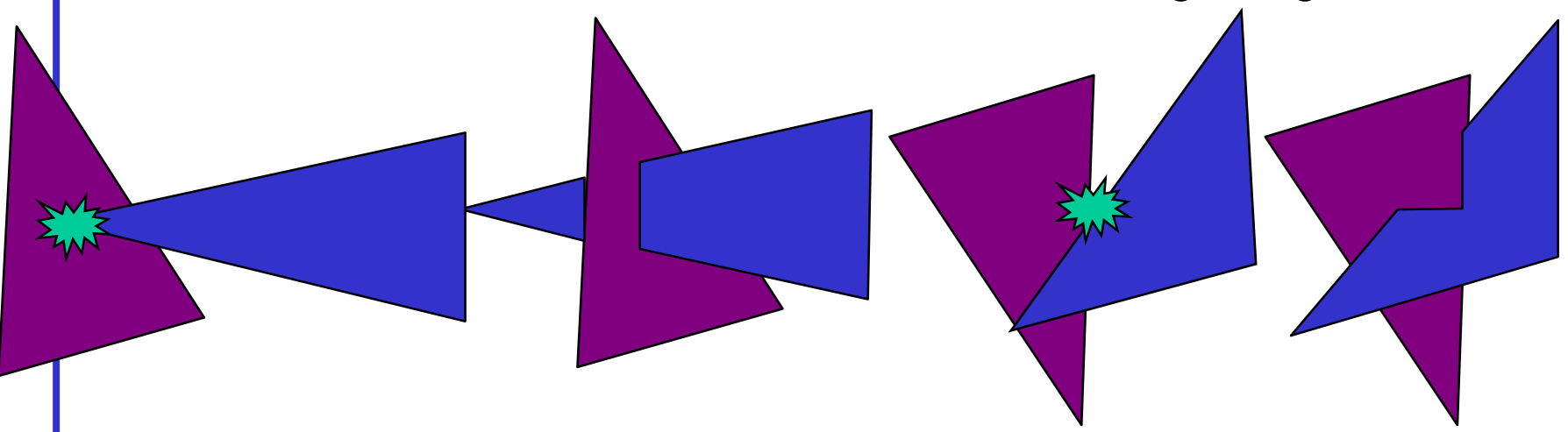
- CCD计算较复杂
  - 一对三角形需要15次单元测试 (6个VF/ 9 个EE测试)
  - 每个单元测试 (Edge/Edge测试或Vertex/Face测试) 都需要求解一个三次方程

CCD计算

- 一对三角形之间的CCD测试可以分解为以下VF/EE测试：

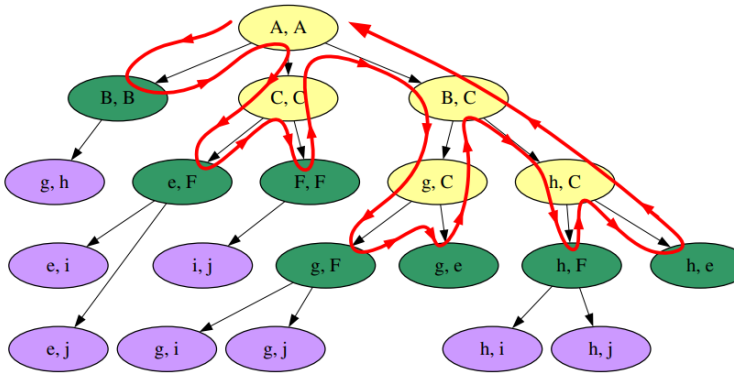
–6个Vertex/Face测试

–9个Edge/Edge测试



并行碰撞检测

- 多核CPU上的可扩展算法



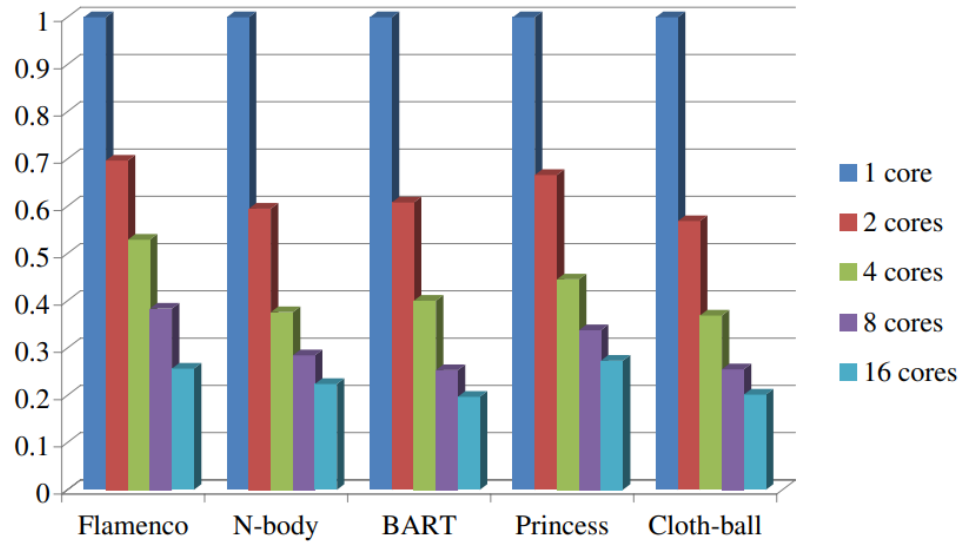
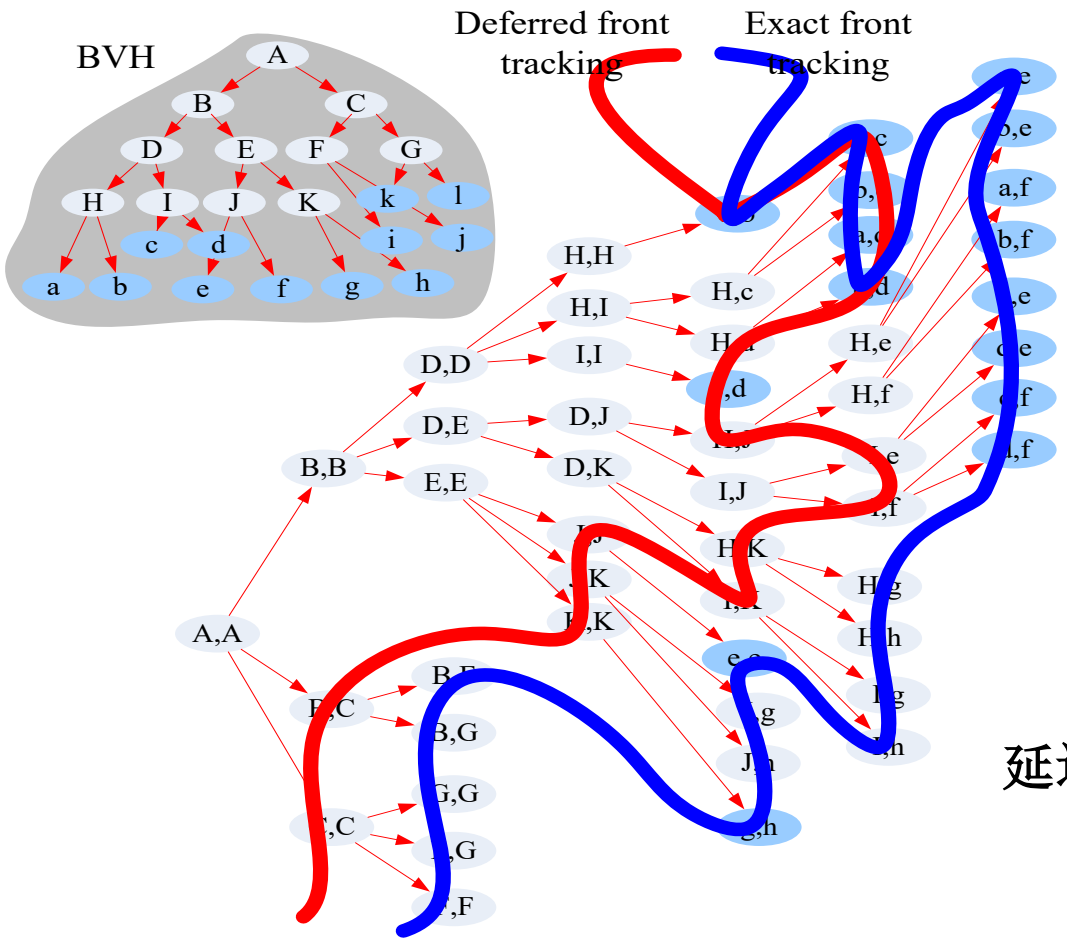


Figure 4: **Speedup of PHC based parallel CCD:** For the benchmarks, the speedups are varying from 3.7*X* to 5.1*X* when all the 16 cores are used.

并行碰撞检测

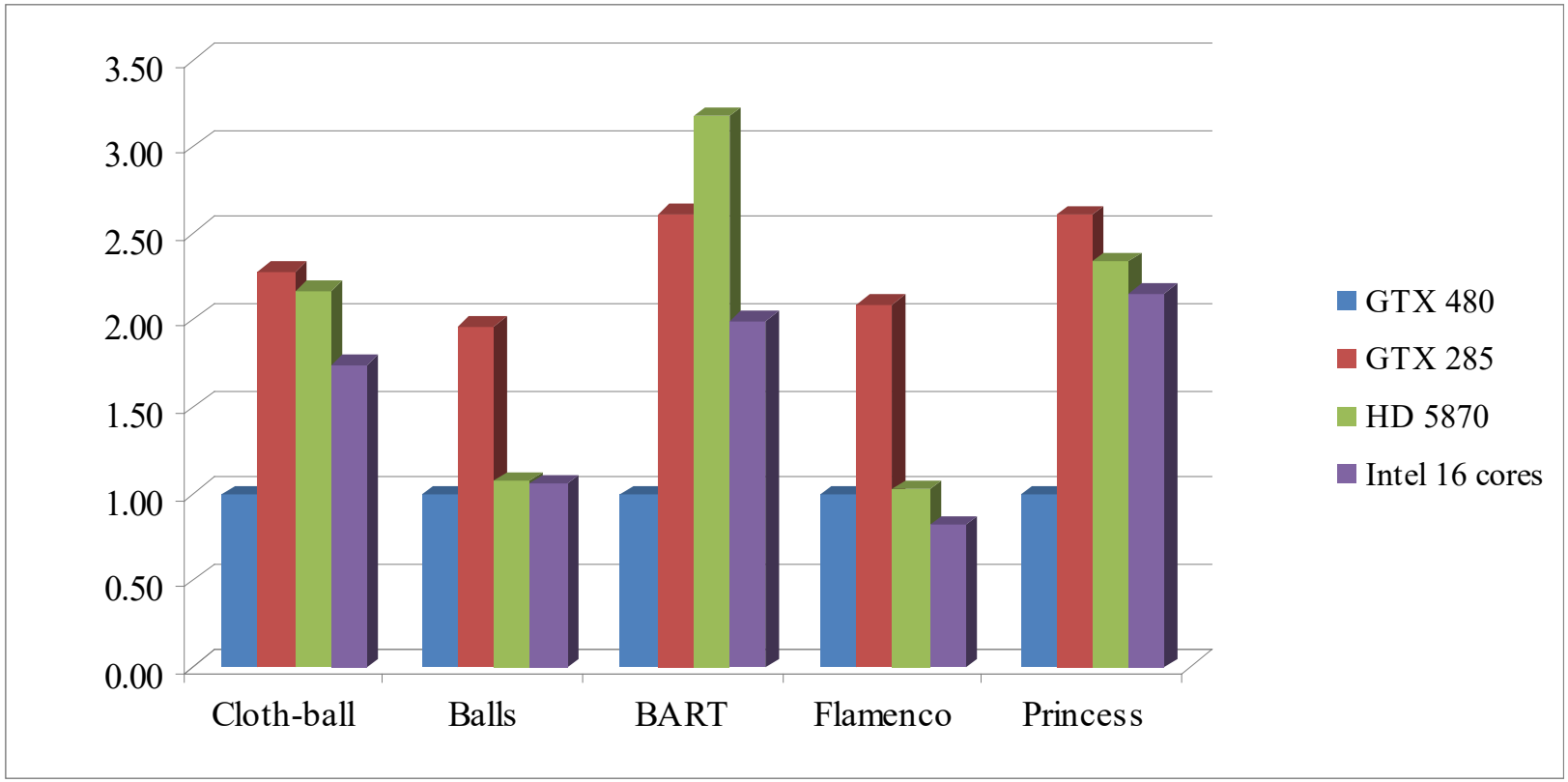
- GPU上的可扩展算法: Collision-Streams
  - 基于GPU的流式CCD算法
  - 通过流注册方法支持边长数据结构
  - 可以与其他剔除算法集成使用
  - 使用了延迟前线跟踪来减少内存开销

延迟前线跟踪



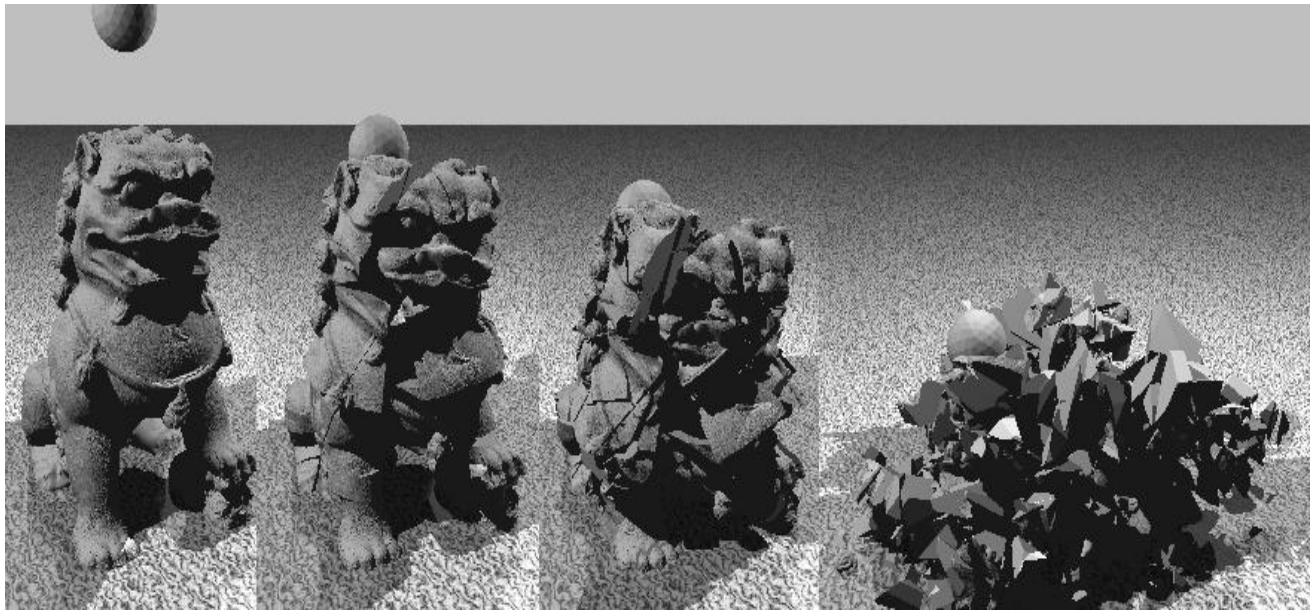
延迟前线跟踪与**精确前线跟踪**的对比

性能对比


3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
Cloth-ball
Balls
BART
Flamenco
Princess
GTX 480
GTX 285
HD 5870
Intel 16 cores

物理仿真中的碰撞处理

- 刚体仿真

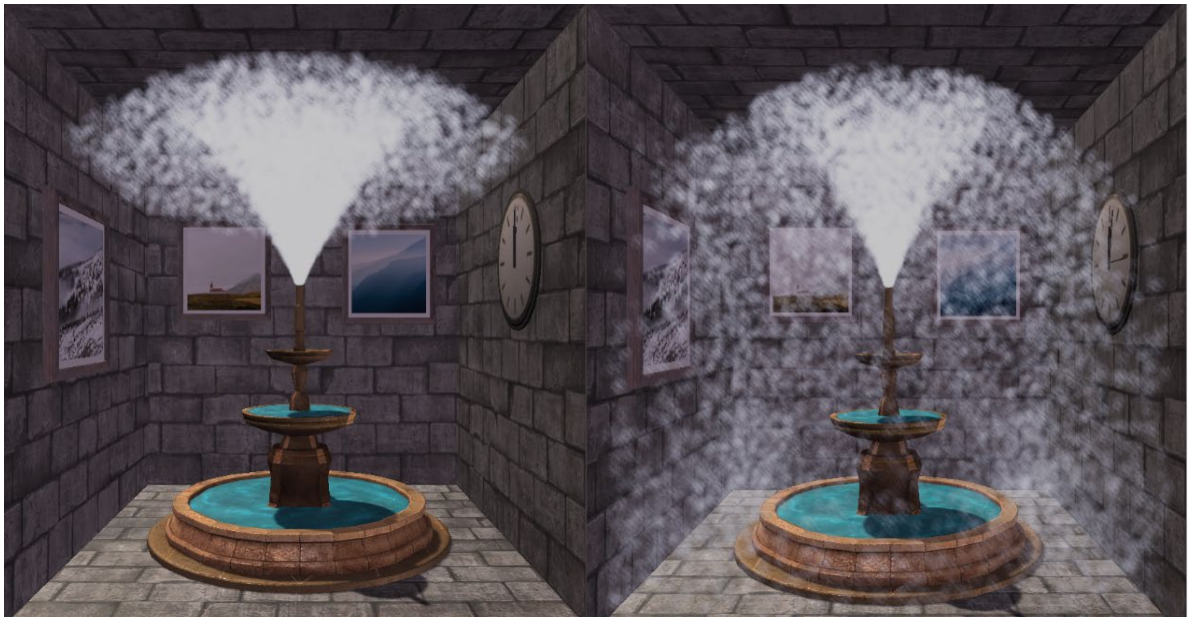


物理仿真中的碰撞处理

- 柔性体仿真

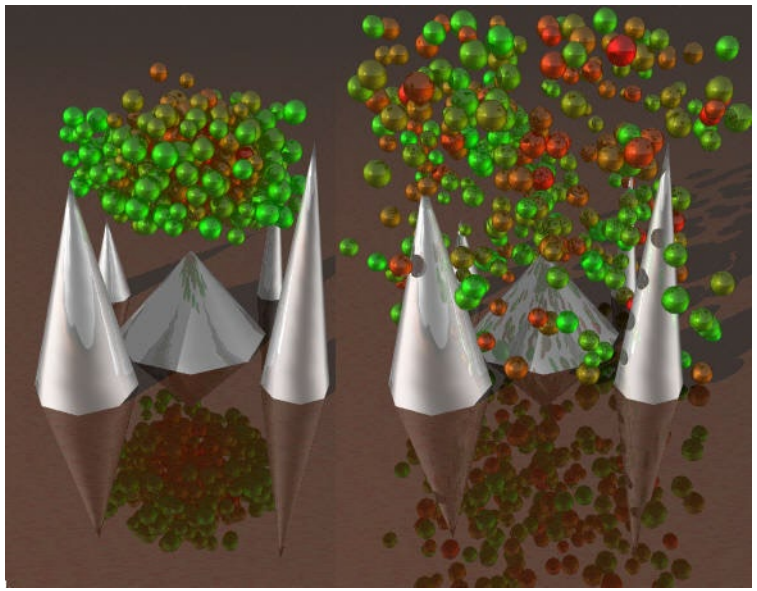
物理仿真中的碰撞处理

- 流体仿真



开源平台

- MCCD
  - http://gamma.cs.unc.edu/MCCD/
  - C++源码
  - 基于OpenMP的多核CPU加速

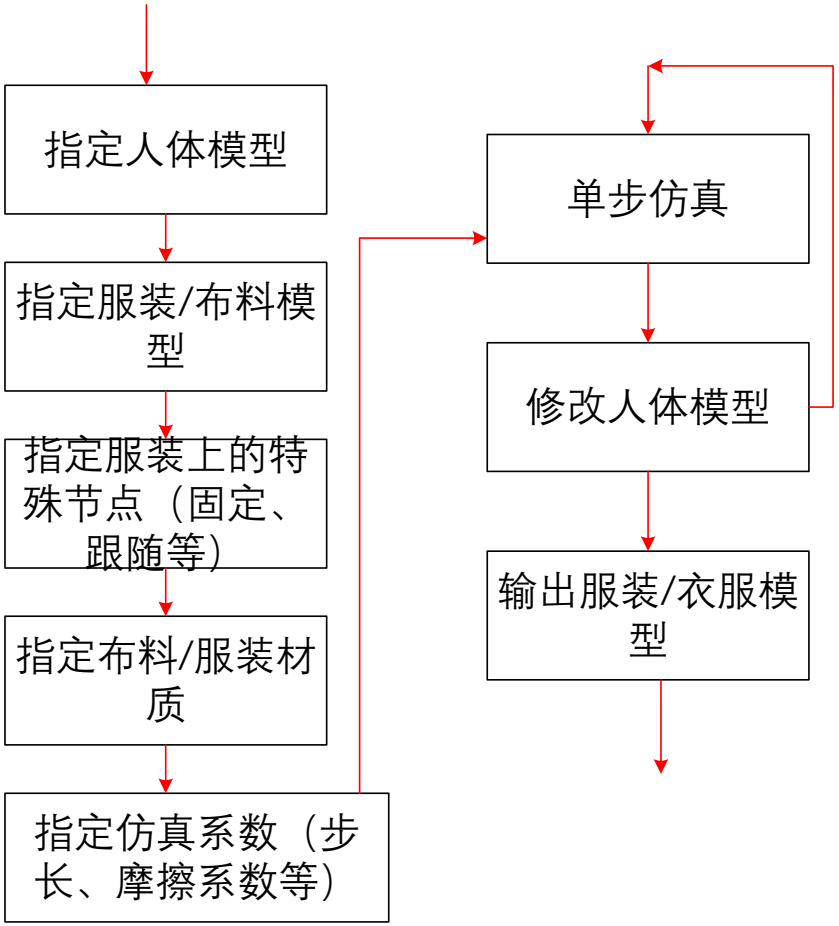


开源平台

- I-Cloth API
  - https://min-tang.github.io/home/gpuCloth/
  - C++ API（DLL与头文件）
  - NVIDIA GeForce GTX 1800 GPUs以上
  - NVIDIA CUDA SDK 9.0以上
  - 支持Windows/Ubuntu操作系统

开源平台

- I-Cloth API
  - 调用流水线



谢谢！