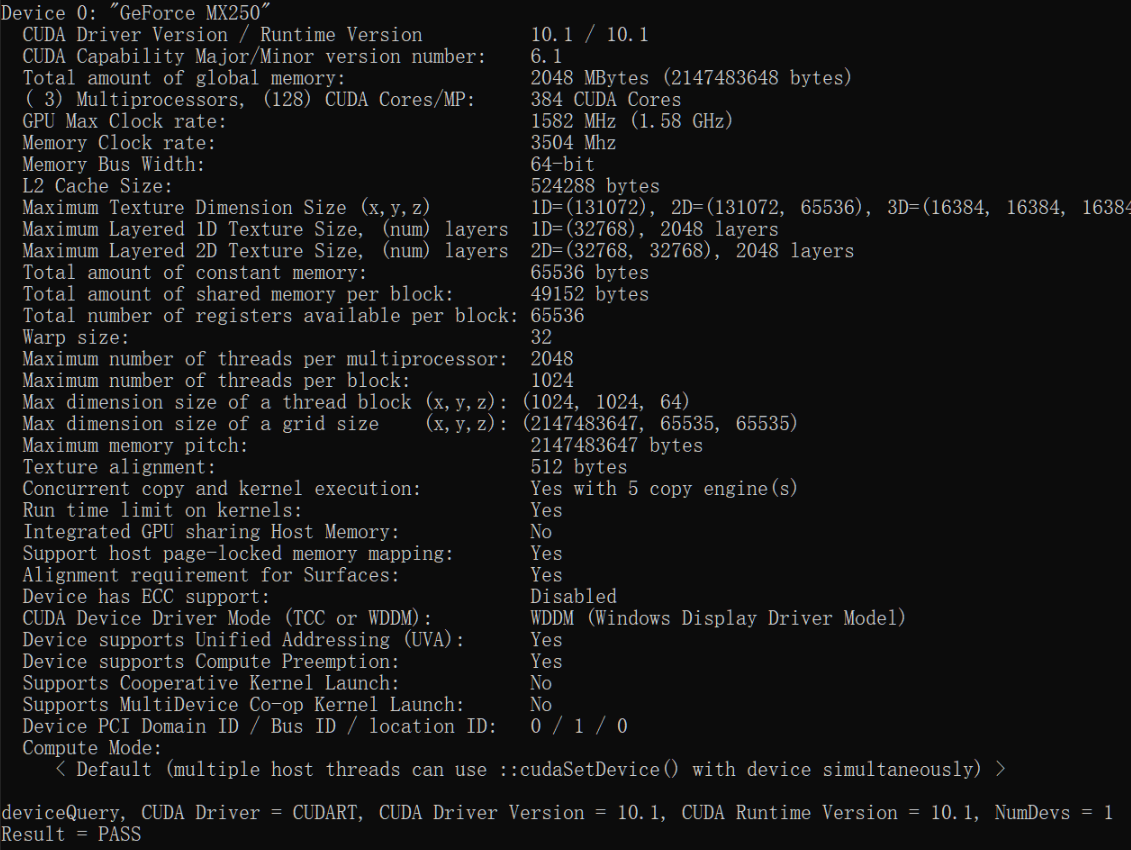
**GPU加速碰撞检测**

**实验环境**



Win10，Visual Studio2019

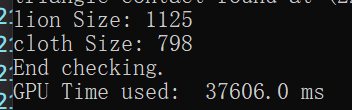
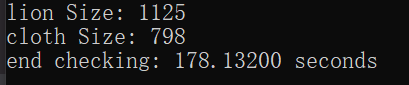
**实验项目**

1. GPU加速检测衣服和身体的碰撞
2. CPU+层次包围盒加速
3. GPU+层次包围盒加速
4. 自碰撞检测

**实验结果**

**GPU加速检测衣服和身体的碰撞（键盘4激活检测）**

在设置时，GPU代码结果如下图。

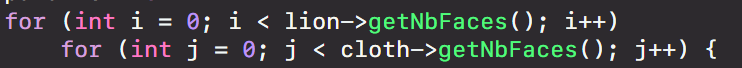


**CPU GPU**

在包含CPU内存和GPU内存传输数据时间的前提下，GPU得到了38秒的结果，加速了。正确性参考lion size和cloth size的数值（即两个STL set的规模）。

其中使用了莫顿码将二维的编码为一个数字，将结果拷贝到CPU主存上后再转为二维数对并输出结果。

总体思路是将二层for循环分散到多个线程去做，但为了保证正确性，需将维或维的



线程数设为1，以保证所有的三角形都能被遍历到，我选择将维设为1个线程。

对于输出结果的存储，我为每个线程限定了1000个数组单元存储。

三组对比试验如下表所示：

|  |  |  |
| --- | --- | --- |
| CPU | 178秒 | 80MB内存 |
| GPU 64× 64个线程 | 47秒 | 260MB内存 |
| GPU 128× 128个线程 | 38秒 | 360MB内存 |
| GPU 128× 256个线程 | 44秒 | 480MB内存 |

更多的线程数要开更大的数组存储和传输结果。

从这个实验可以看出数据传输效率和线程数的选择深刻影响GPU程序的整体效率。

**层次包围盒加速**

用于碰撞检测的包围盒有几种 [1]：包围球(Sphere)，沿坐标轴的包围盒(AABB axis-aligned bounding boxes)，方向包围盒(OBB oriented bounding box)。我采用最容易的AABB包围盒。

具体来说，对象的AABB包围盒被定义为包含该对象且各边平行于坐标轴的最小六面体。两个AABB包围盒相交它们在3个坐标轴上的投影区间均重叠。

构造包围盒有自顶向下和自底向上两种方式。我的想法是先用大的立方体包住所有三角形，然后参考 [1]，确定一个分裂平面，将三角形集合拆分成两部分，递归地调用构建函数，直到只剩一个三角形为止。

在检测碰撞时，对于输入的两个BVH树，需判断当前树节点，如果不相交则直接返回；若相交再分别递归检验两节点的子节点共四组。由于BVH的特性，可以省去很多不必要的基本图形的相交检测，从而大幅降低运行时间。

**层次包围盒+GPU加速**

### 传统的构建BVH树的做法不易并行，我参考了《[Physically Based Rendering: From Theory to Implementation (pbr-book.org)](http://www.pbr-book.org/)》中的做法，里边提出的**Linear Bounding Volume Hierarchies(HLBVH)**易于并行。

# References

|  |  |
| --- | --- |
| [1] | 潘振宽, 崔树娟, 张继萍 and 李建波, "基于层次包围盒的碰撞检测方法," *青岛大学学报: 自然科学版,* vol. 18, p. 71–76, 2005. |