GPU 加速碰撞检测

实验环境

```
Device 0: "GeForce MX250"
  CUDA Driver Version / Runtime Version          10.1 / 10.1
  CUDA Capability Major/Minor version number:    6.1
  Total amount of global memory:                 2048 MBytes (2147483648 bytes)
  ( 3) Multiprocessors, (128) CUDA Cores/MP:     384 CUDA Cores
  GPU Max Clock rate:                            1582 MHz (1.58 GHz)
  Memory Clock rate:                             3504 Mhz
  Memory Bus Width:                              64-bit
  L2 Cache Size:                                 524288 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 5 copy engine(s)
  Run time limit on kernels:                     Yes
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  CUDA Device Driver Mode (TCC or WDDM):         WDDM (Windows Display Driver Model)
  Device supports Unified Addressing (UVA):      Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            No
  Supports MultiDevice Co-op Kernel Launch:      No
  Device PCI Domain ID / Bus ID / location ID:   0 / 1 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 10.1, CUDA Runtime Version = 10.1, NumDevs = 1
Result = PASS
```

Win10，Visual Studio2019

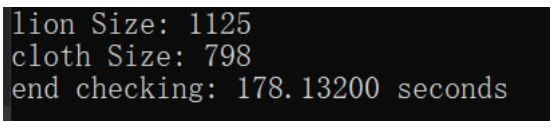
实验项目

i. GPU 加速检测衣服和身体的碰撞
ii. CPU+层次包围盒加速
iii. GPU+层次包围盒加速
iv. 自碰撞检测

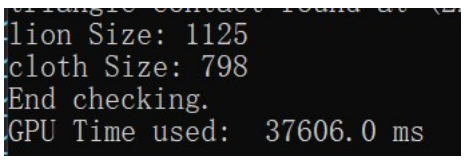
实验结果

GPU 加速检测衣服和身体的碰撞（键盘 4 激活检测）

在设置$\text{grid} = (128,1), \text{block} = (256,1)$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79%。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(i,j)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x维或y维的

```
for (int i = 0; i < lion->getNbFaces(); i++)
    for (int j = 0; j < cloth->getNbFaces(); j++) {
```

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y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)》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.