**基于GPU进行三维网格热传导模拟**

金秉阳 航空航天学院 22124045 手机号：15356156070

# 摘要

针对水密、二边流形的三维网格模型的热传导问题，本文基于GPU进行并行加速计算，对所有三角形网格的边进行哈希映射以得到邻接三角形数据，并在核函数中进行循环迭代计算，相比于采用CPU计算，得到了约1000倍的加速，在更短时间内实现了热传导问题的求解。

# 关键词：GPU、并行计算、热传导

# Abstract

Aiming at the heat conduction problem of the watertight and 2-manifold 3D mesh model, this paper based on GPU parallel acceleration calculation, hashes the edges of all triangle meshes to obtain adjacent triangles, and loops and iteratively calculates in the kernel function. Compared with the use of CPU calculation, about 1000 times speedup is obtained, and the heat conduction problem is solved in a shorter time.

# Keywords：GPU, parallel computing, heat conduction

# 引言

物理模拟问题是计算中极具有挑战性的问题，一个简单的三维热传导模型可以较方便地使用GPU进行加速，本文主要基于C++语言和CUDA，利用VS开发了一个GPU并行计算三维网格热传导地模拟程序，并给出实验结果。该程序输入为.obj文件格式存储的非结构三角网格模型，需保证模型水密且均为二边流形。该程序会对模型进行可视化渲染，并计算其热传导模拟结果。

对于水密且均为二边流形的三角网格模型，在拓扑上，任意一个三角网格均与3个三角形相连（即共享一条边），如图1.1所示，三角形A与三角形B、C、D相连。：



**图1.1 三角形相连示意图**

考虑热传导仅会在相连三角形之间发生，对于三角形A而言，其邻接三角形定义为与三角形A共享一条边的三角形，在图1.1中即三角形B、C、D。在时间递进的每个步骤中，如果某个单元的邻接单元温度更高，那么热量将从邻接单元传导到该单元。相反地，如果某个单元的温度比邻接单元的温度高，那么它将变冷。

在上述热传导模型中，本文对单元中新温度的计算方法为，将单元与邻接单元的温度像擦汗加起来，然后加上原有温度，计算公式如等式1.1所示：

等式1.1[1]

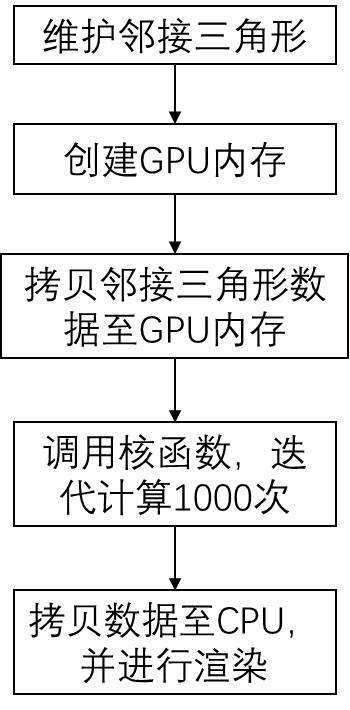
本程序可视化渲染部分参考老师所给出基于glut库的渲染，热传导模拟计算部分主要依赖CUDA完成。

本文所提及所有代码均通过git进行版本管理，远程仓库位于github，地址为：https://github.com/Luciuser/GPU\_HeatConduction。

所上交内容中，src文件夹存储本次作业全部源码、extern文件夹存储用到的第三方库，objects文件夹存储用到的.obj格式网格文件，lib文件夹存储链接静态库，build文件夹为Visual Studio生成的工程文件，其中build/bin即为生成的可执行程序，bin文件夹包含CPU及GPU可执行程序及相应链接动态库。

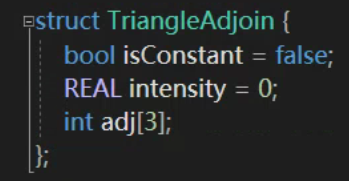
# 程序流程

图2.1给出了本文基于GPU进行热传导模拟的程序流程。



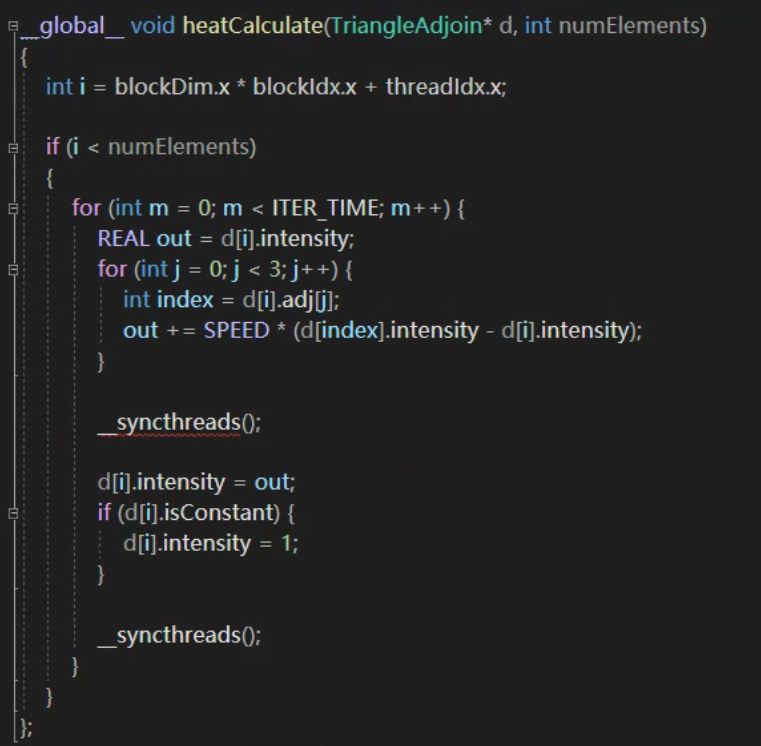
**图2.1 三角形相连示意图**

1. 邻接三角形数据通过图2.2所示结构体进行存储，其中isConstant标志该三角形是否为热源点；intensity表示该三角形的热量；adj数组记录邻接三角形索引。



**图2.2 邻接三角形存储结构体**

1. 所创建GPU内存即存储TriangleAdjoin结构体的数组，拷贝均采用CUDA所提供的内存拷贝函数。
2. 核函数内部进行循环遍历，如图2.3所示。



**图2.3 热传导问题核函数**

默认情况下核函数内部进行1000遍历，每次循环，核函数执行两项工作：根据公式1.1计算每个三角形在下一个时间点的热量；将热量重新写回内存，并将热源位置的热量重置为1。

# 加速说明

为提高并行加速效果，本程序采取了一些加速方案，以下针对这些加速方案进行说明。

## 邻接三角形构建

老师所给定CPU计算程序中，使用BVH树+AABB包围盒方案获取每个三角形的邻接三角形，并进行热传导计算。在此过程中，每个三角形所记录的邻接三角形数量不等，平均值约为20个左右。本程序取消了BVH树及AABB包围盒方案，通过共享边机制获取每个三角形的邻接三角形，并确定每个三角形的邻接三角形有且仅有3个。考虑到热量传导的基础条件之一是有接触，因此从理论上认为应当建立本文所述邻接三角形。

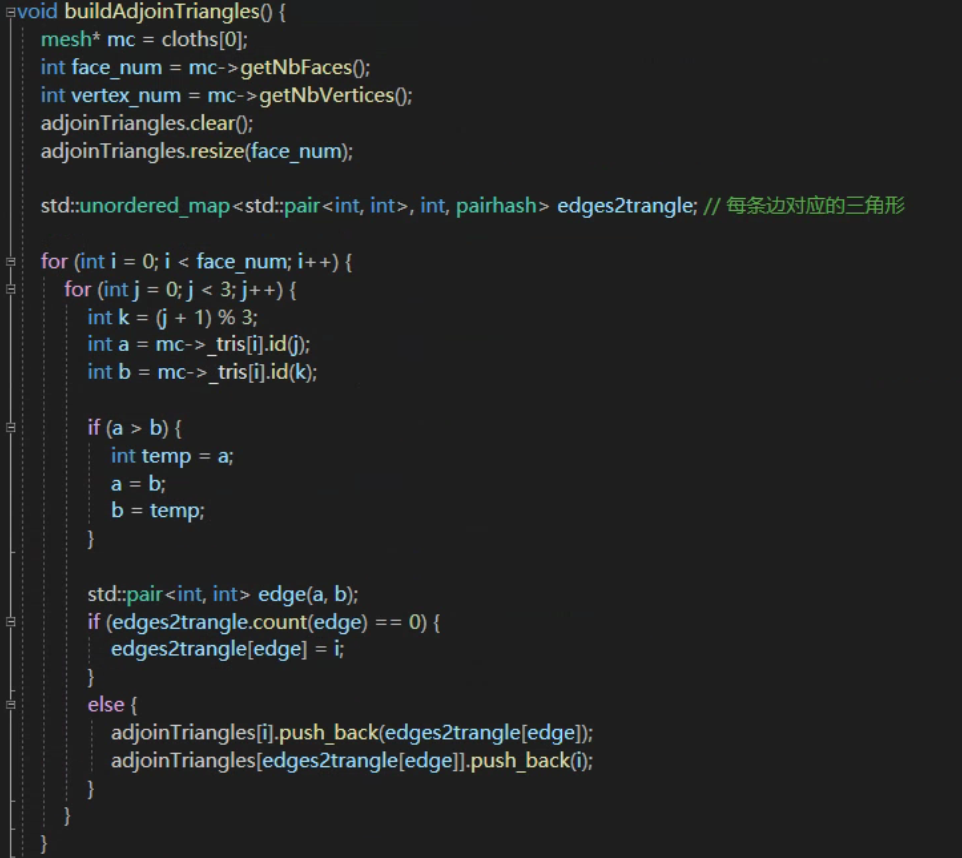
本程序中大量使用每个三角形的邻接三角形数据，且该部分数据在整个程序的生命周期中均不发生变化，因此选择在开始进行热传导计算之前建立每个三角形的邻接三角形数据，后续均直接读取此数据进行计算。

邻接三角形数据的建立通过对边进行哈希得到。考虑如图3.1所示的两个三角形A和B共享点v1、v2。由于该模型水密且属于二边流形，同时包含点v1和v2的三角形有且仅有两个，即A和B，故A和B关于边v1v2邻接。



**图3.1 邻接三角形**

建立邻接三角形的具体实现如图3.2所示，在遍历所有的三角形的三条边中，根据点v1、v2建立边，并对边进行哈希映射。若该映射已存在，则建立两三角形之间的邻接关系；若该映射不存在，则加入该映射。



**图3.2 建立邻接三角形**

## 常量内存

由于本程序需要读取大量网格，并存储相应的邻接三角形数据及热量数据。GPU常量内存空间相对较小，无法存储大量数据，因此不采用constant memory存储数据，所有数据均存储于global memory中。

## 核函数循环计算

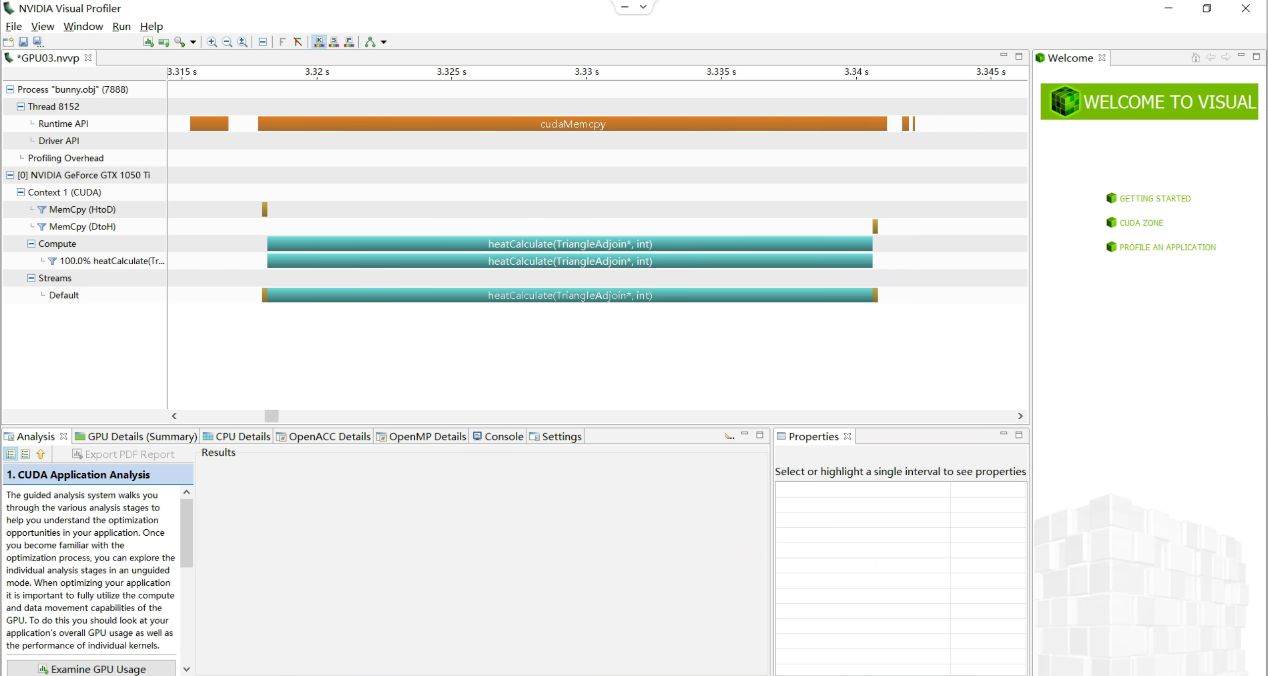
在本程序计算过程中，热传导结果仅作为下次GPU计算的输入数据和可视化渲染，其中可视化渲染由CPU进行，但无需将每次计算的结果进行渲染。因此考虑对核函数进行循环计算。GPU每次根据公式1.1计算结束后，不会将数据送还给CPU，而是继续进行迭代计算，直至达到所设定循环次数后，再将数据送还给CPU进行可视化渲染。通过此方法可以极大幅度减少热量数据自CPU内存拷贝至GPU以及自GPU拷贝至CPU。

具体核函数循环计算的代码可参见图2.3。

## 流

在进行GPU加速中，使用流（stream）可以充分利用GPU的工作调度，即在进行内存拷贝的同时，GPU核进行计算。

但本工程中，由于采用了上述3.3所使用的核函数循环计算的策略，内存拷贝的时间远远小于计算时间，图3.3所示为NVIDIA Visual Profiler软件对可执行程序性能分析的结果，其中黄色部分为内存拷贝时间，蓝绿色部分为计算时间。由图易得，采用流进行加速收益很小，本文不再使用流进行加速。



**图3.3 程序性能分析**

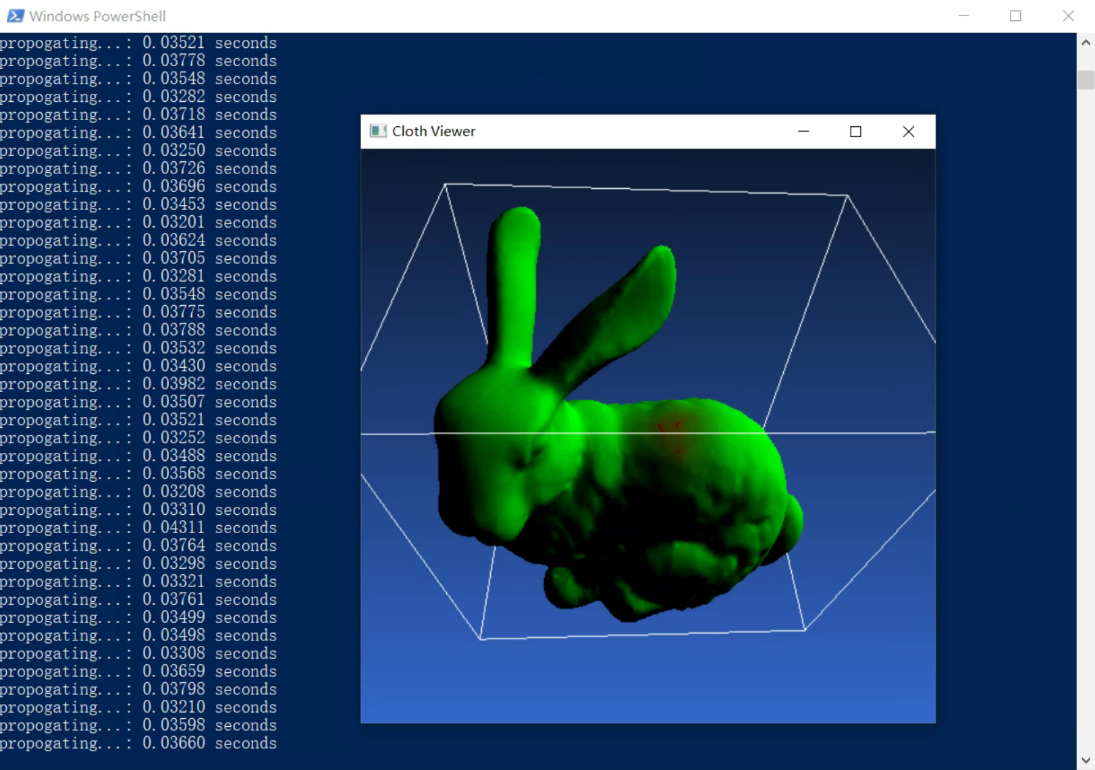
# 实验结果

此部分主要提供渲染结果核实验数据。测试用机CPU为Intel(R) Core(TM) i5-7300HQ @ 2.50GHz，内存为8 GB，显卡为NVIDIA GeForce GTX 1050 Ti，测试环境为Window 10。其中GPU进行循环的次数为1000次。

## Bunny兔

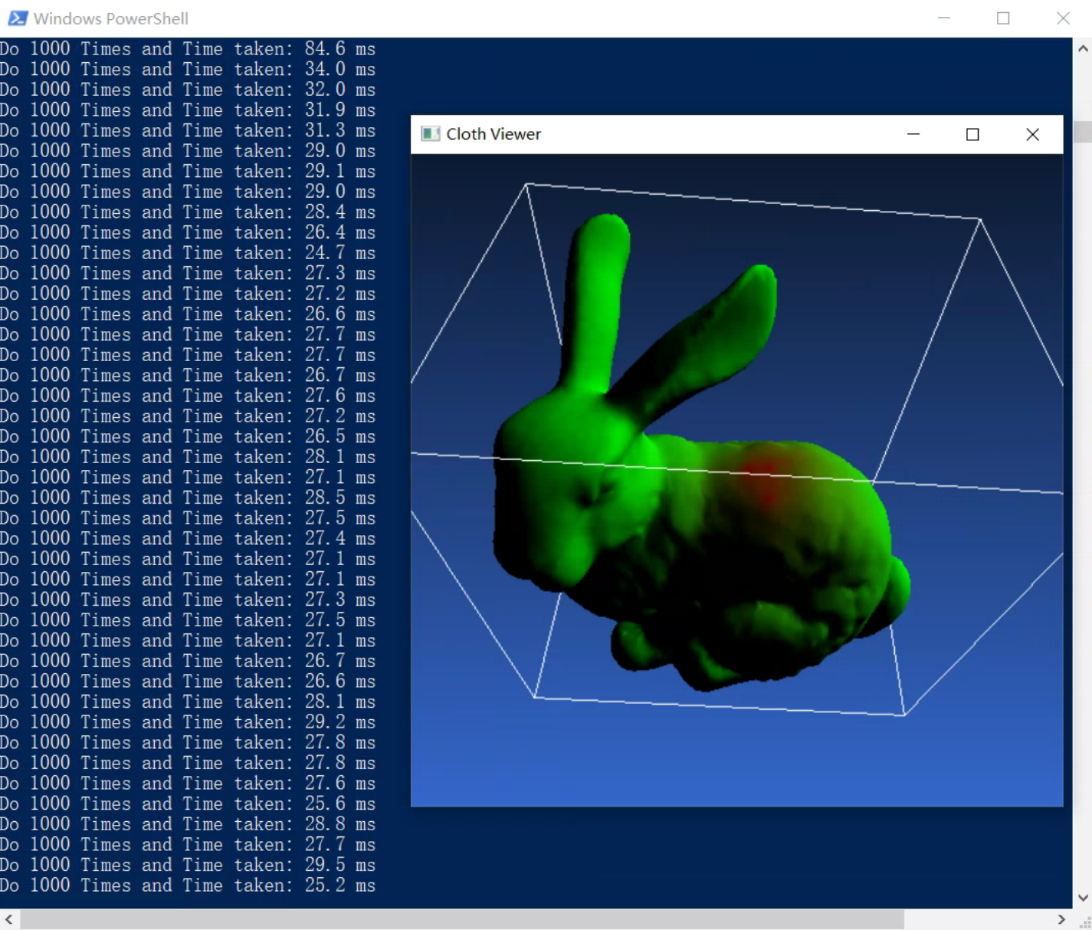
以下实验采用bunny兔的模型，该模型由18221个顶点和36438个三角形组成，并保证模型的水密性和二边流形。

使用CPU进行计算的结果，如图4.1所示，其中红色表示热量值接近1，绿色表示热量值接近0，每次计算平均耗时约35 ms。



**图4.1 CPU 计算bunny每次迭代耗时**

使用GPU进行计算的结果，如图4.2所示，其中红色表示热量值接近1，绿色表示热量值接近0，每1000次计算平均耗时约29 ms。



**图4.2 GPU 计算bunny每1000次迭代耗时**

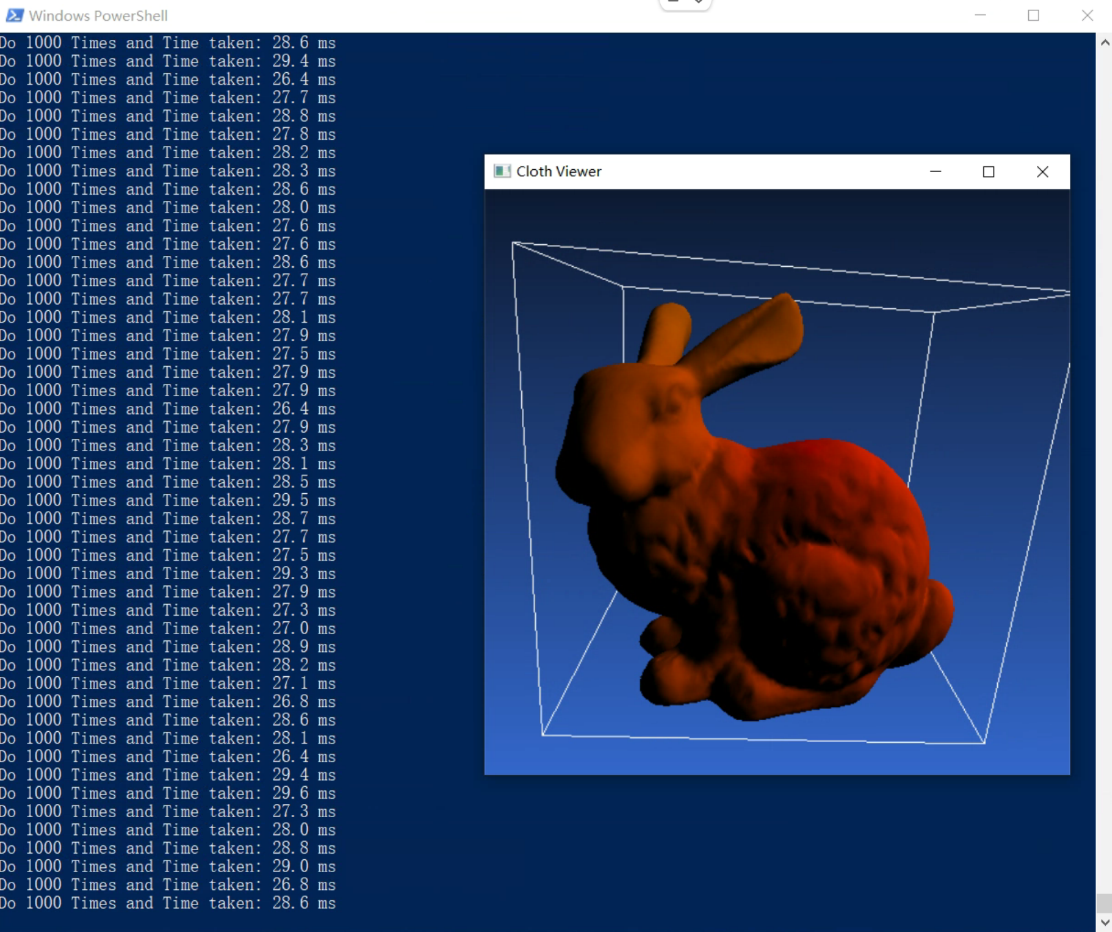
由上述实验数据，相比于CPU计算，采用GPU并行计算可达到1000倍左右的计算结果。

使用CPU迭代计算10分钟，所得到的可视化渲染结果如图4.3所示，其中红色表示热量值接近1，绿色表示热量值接近0。每次计算平均耗时约35 ms。



**图4.3 CPU 计算bunny 10分钟渲染结果**

使用GPU迭代计算10分钟，所得到的可视化渲染结果如图4.4所示，其中红色表示热量值接近1，绿色表示热量值接近0。每1000次计算平均耗时约29 ms。



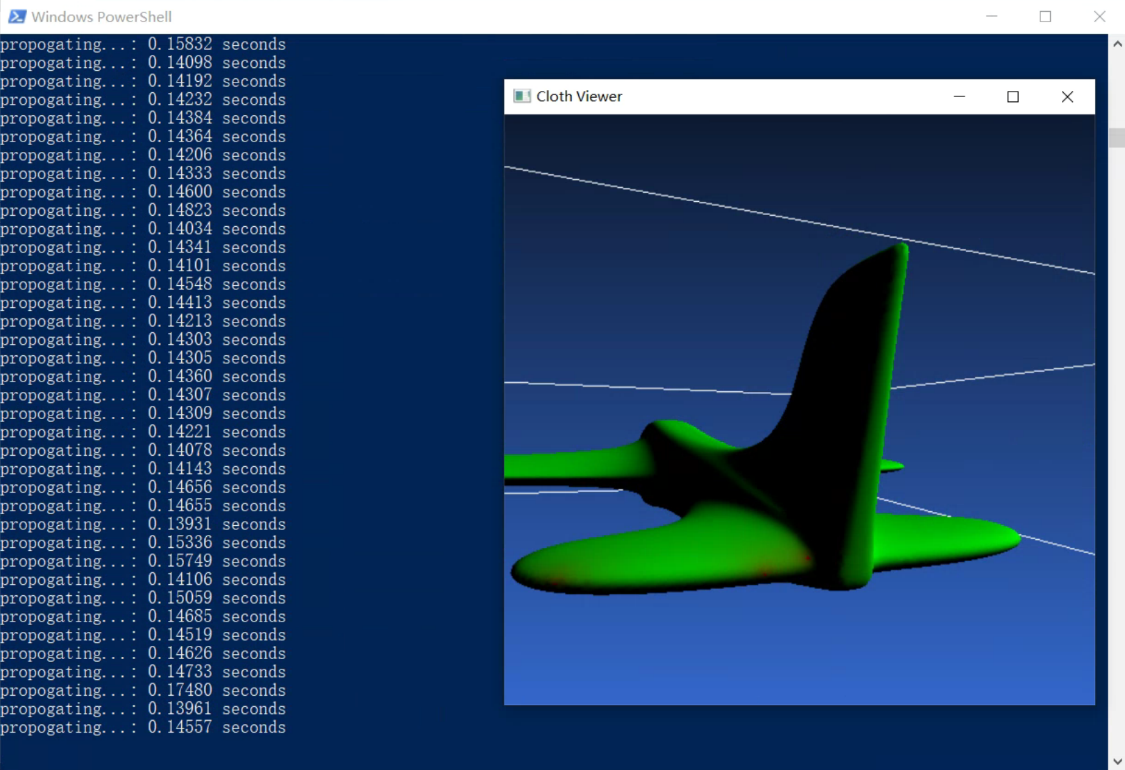
**图4.4 GPU 计算bunny 10分钟渲染结果**

由上述实验数据，GPU计算10分钟后，整个模型除耳朵部分外均已呈红色，耳朵部分也已呈现橙红色，渲染结果已接近热传导最终结果；而CPU计算10分钟后，整个模型除背部接近网格源点位置外均为绿色，表明热传导尚未传递至除背部外其他区域。可见GPU并行计算有着很高的加速效率。

## Aircraft

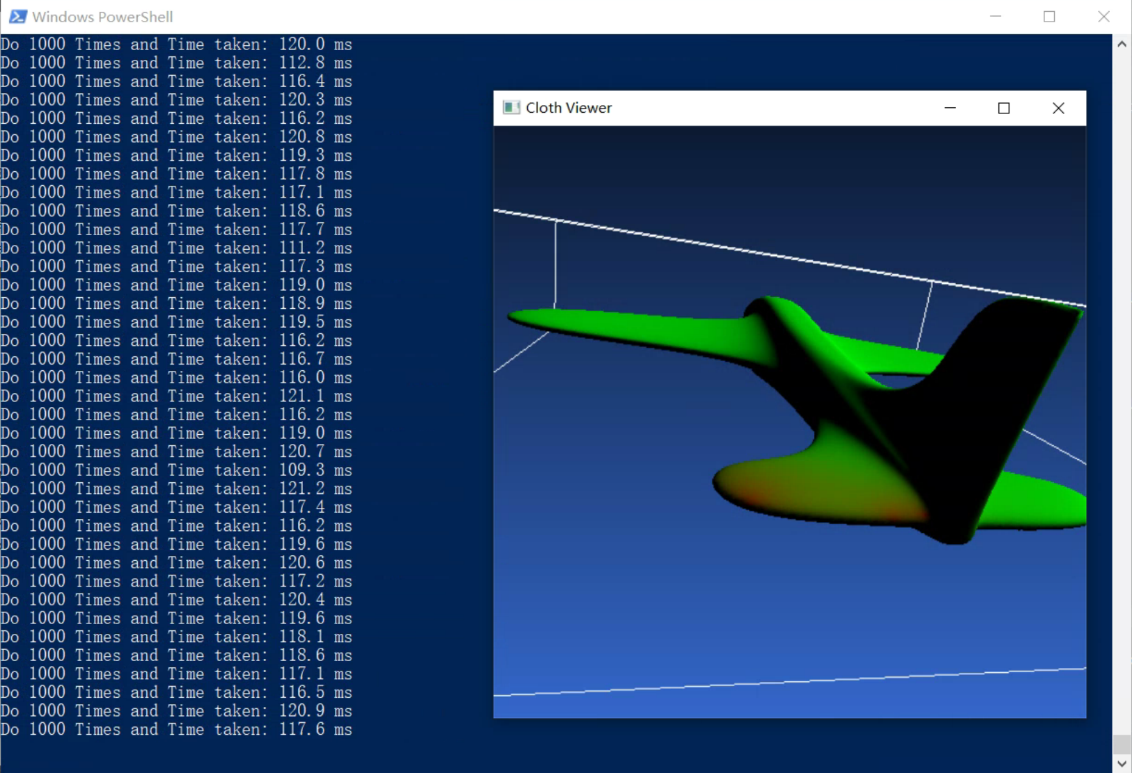
以下实验采用aircraft的模型，该模型由84798个顶点和169592个三角形组成，并保证模型的水密性和二边流形。

使用CPU进行计算的结果，如图4.5所示，其中红色表示热量值接近1，绿色表示热量值接近0，每次计算平均耗时约143 ms。



**图4.5 CPU 计算aircraft每次迭代耗时**

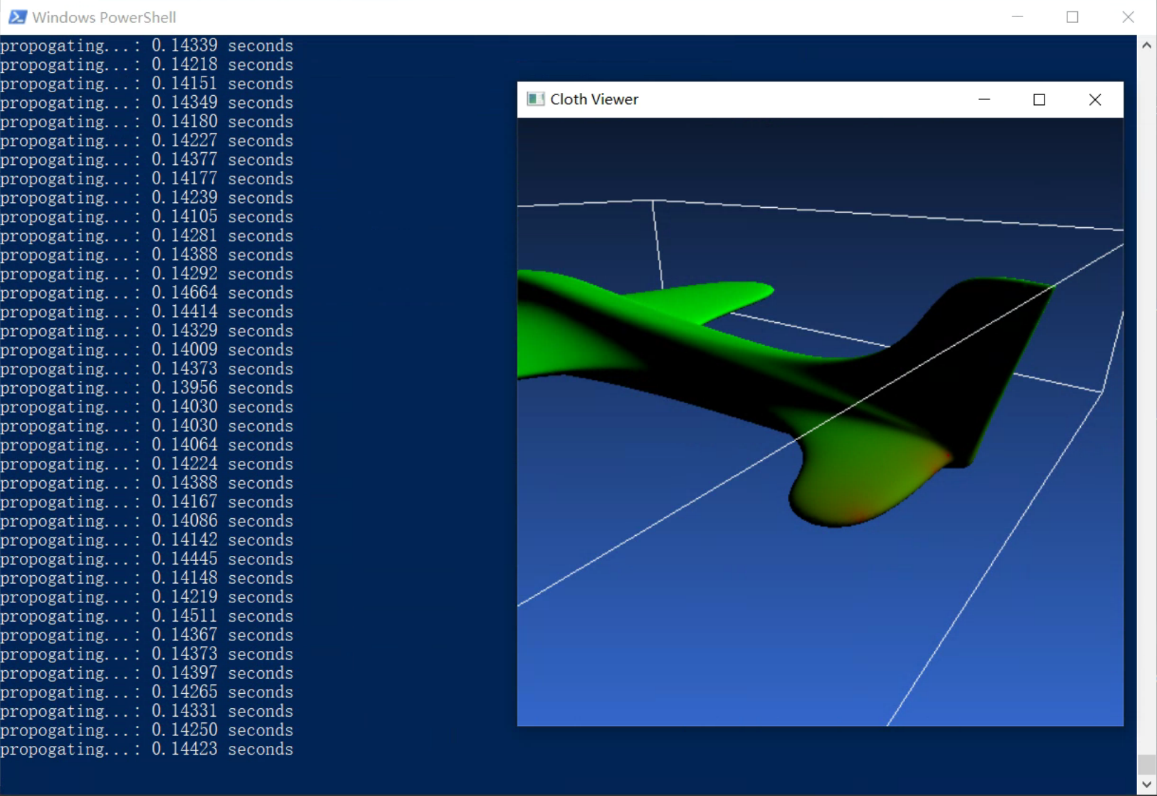
使用GPU进行计算的结果，如图4.6所示，其中红色表示热量值接近1，绿色表示热量值接近0，每1000次计算平均耗时约120 ms。



**图4.6 GPU 计算aircraft每1000次迭代耗时**

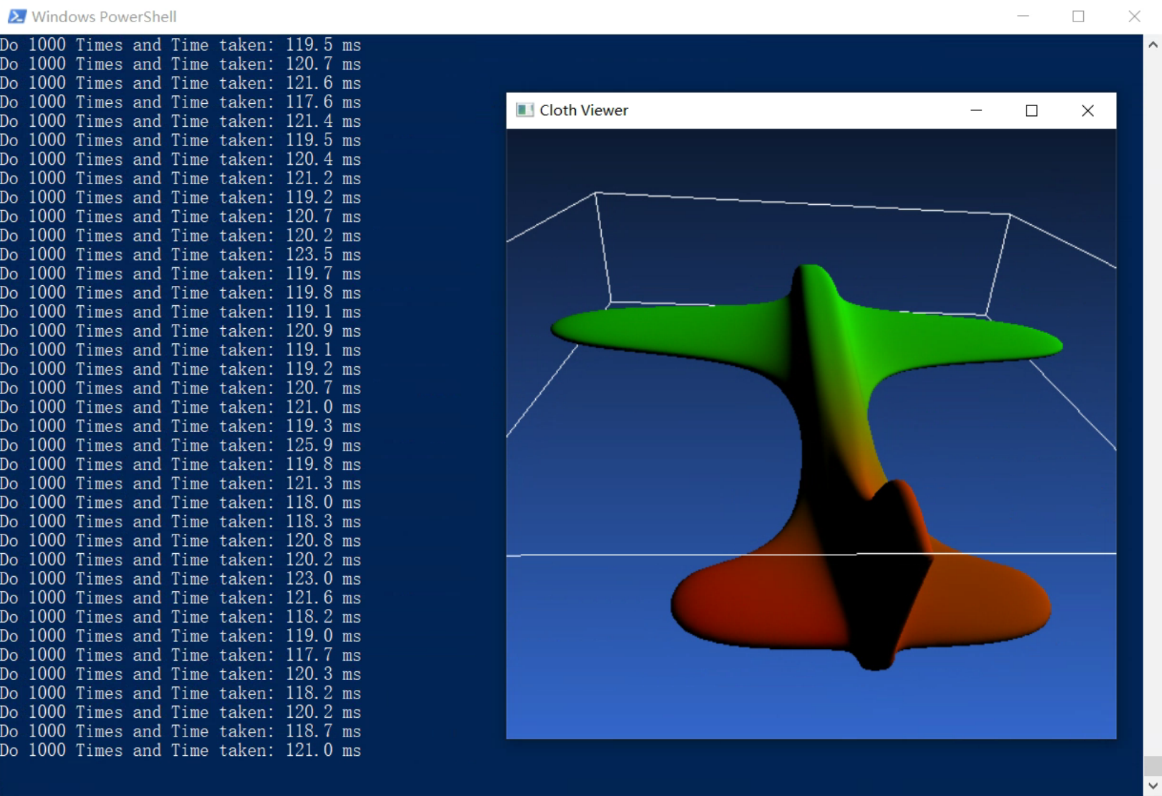
由上述实验数据，相比于CPU计算，采用GPU并行计算可达到1000倍左右的计算结果。

使用CPU迭代计算10分钟，所得到的可视化渲染结果如图4.7所示，其中红色表示热量值接近1，绿色表示热量值接近0。每次计算平均耗时约143 ms。



**图4.7 CPU 计算aircraft 10分钟渲染结果**

使用GPU迭代计算10分钟，所得到的可视化渲染结果如图4.8所示，其中红色表示热量值接近1，绿色表示热量值接近0。每1000次计算平均耗时约120 ms。



由上述实验数据，GPU计算10分钟后，飞机模型前半部分呈现绿色，说明热量尚未传导过去，而后半部分呈红色及橙红色；而CPU计算10分钟后，整个模型仅在热源点附近存在少许红色，整体飞机网格均为绿色。可见GPU并行计算有着很高的加速效率。

# 总结

针对三维网格模型的热传导问题，本文基于GPU进行并行加速，以共享边为基础构建邻接三角形，并采用核函数循环进行计算，相比于CPU计算得到了约1000倍的加速，在更短时间内实现了热传导问题的求解。

本次作业具体完成过程中，依然存在一些不足，存在一些可以继续加速的地方。主要在于可采用 shared memory 进行加速。由于每个线程在计算时除了访问当前的三角形之外，还会访问另外3个邻接三角形；因此每次迭代中，每个三角形共计被4个不同的线程访问1次。在多次访问同一块内存空间时，使用global memory效率较差，可以使用shared memory存储三角形邻接数据和三角形的热量值。但由于shared memory仅会被同一个block内的线程共享，因此在进行block分区时需进行额外操作。具体而言，需将整个三维网格模型按照block的数量划分为多个区，每个区均为邻接三角形的集合，同时每个block的shared memory保存相应区邻接三角形数据。对于区和区之间的交界部分，所对应的两个block需要额外存储一层三角形数据，以保证自己内部的所有三角形均可以进行计算。

后续将考虑采用shared memory方案进行加速，并测试加速效果。

# 参考文献

[1] Jason Sanders, Edward Kandrot 等著，聂雪军等译《GPU高性能变成CUDA实战》，机械工业出版社，2022年1月第1版，87页。