Loop Subdivison 算法

课程：几何造型基础

姓名：赵崇遥　（PHD first year)

学号：11821039

开发环境

系统信息

系统	信息
操作系统	Linux Mint 19 Cinnamon
处理器	Intel© Core™ i7-4790 CPU @ 3.60GHz × 4
内存	32GB
显卡	GeForce GTX 750
编译器	gcc 7.3
编译工具	Cmake 3.10.2

项目依赖库

矩阵库：Eigen3

并行库：Openmp(verison:4.5)

常用库:Boost(version:1.65.1)

具体查看根目录下的CMakeLists.txt

编译运行

在满足依赖的情况下，进入根目录，开启终端运行：

```
$ mkdir build & cd build & cmake .. & make -k
```

输入模型

将模型（.obj格式）放入根目录下的models/中，并修改根目录下的input.json文件。

input.json

```
{
  "indir":"../../models/",
  "outdir":"../../output/",
  "surf":"bunny",
  "times":5

}
```

将surf中的bunny改为模型名称。可以直接使用bunny作为demo测试。times是细分次数。

运行

开启终端进入build/bin/目录，并运行./z_buffer

```
$ cd build/bin/
$ ./z_buffer
```

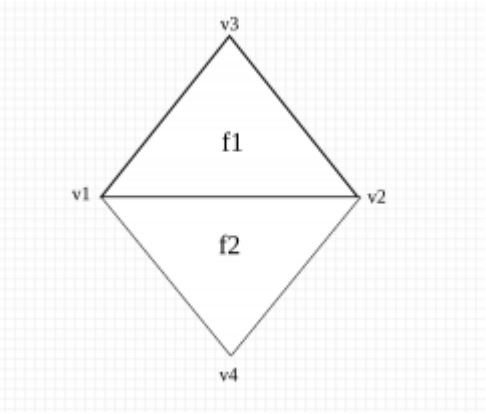
数据结构　edge_core类

很明显该算法是一个可以并行的算法。

本来考虑使用半边结构来做，后来觉得使用半边结构在处理并行的时候比较复杂比较烦，就简单的建立一个边表，类似于翼边结构。

单边结构

```
struct one_edge{
  int f1,f2,v1,v2,v3,v4;
  }
};
```



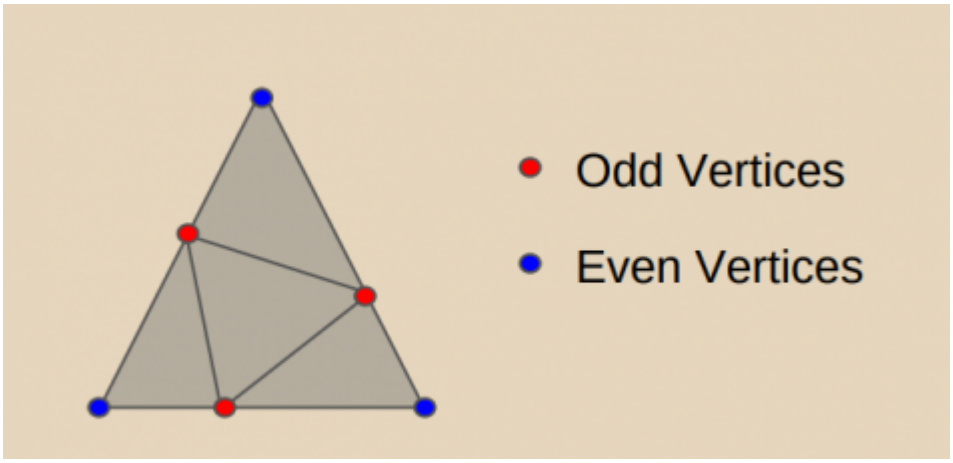
建立两个数组，一个数组存放边，为

```
std::vector<one_edge edges_;
```

一个数组存放每个顶点的valence，即有多少个相邻点与之相连。

```
vector<size_t> valences_;
```

算法简介

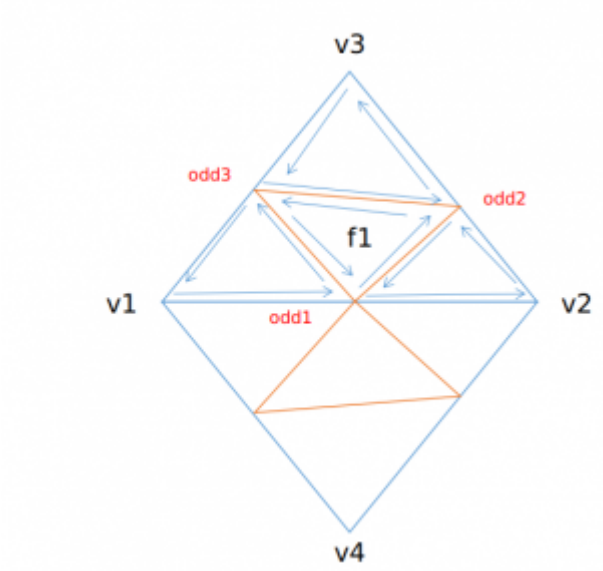


在原三角形网格上的每一个边上插入一个点，称之为odd vertex，而原来的点称之为even vertex。

然后分别去更新odd vertices 和 even vertices的位置。

因此该算法是可以并行的，每一次细分下每个边的更新互不影响。

算法并行实现及优化



New triangle face lookup table

f1*4	f1*4 + 1	f1*4 + 2	f1*4 + 3
v1	v2	v3	odd1
odd1	odd2	odd3	odd2
odd3	odd1	odd2	odd3

对于每一个面片，在加上边后按照一定的顺序设置新的拓扑关系，从而可以并行的在每个边上做处理。这里使用openmp实现线程并行。

建立单边结构的数组

使用一次哈希的方式建立单边结构数组。

更新odd点的位置和even点的位置(new_verts)

这一步直接按照公式，对每条边去算就可以了。

更新拓扑(new_tris)

按照上面的图的顺序在循环每条边，在new_tris中设置 6 个面片（f1 上三个，对面的f2上也是三个）上的六个顶点。

更新单边结构数组

在多次细分时不需要重新建立单边数据结构，每细分一次，单边结构数组的大小为原来的两倍加上细分前面片的数量乘 3。

循环每条边添加一个单边结构并且更新旧的单边结构。

循环每个面添加三个单边结构。

总结与思考

- 在已知拓扑更新方式并且需要知道adjoint table 的时候，使用半边结构不一定是最好的方式。
- 使用智能指针避免在每次循环的时候拷贝数据。

- 细分算法可能并不需要细分很多次，一般细分三到四次可能就差不多了，其次细分前的原网格可能也并不复杂。