**计算机图形学**

**第五次实验报告**

**ARAP**

**PB20000264**

**韩昊羽**

# 一.实验要求

* 实现ASAP,ARAP算法
* 巩固使用UEngine框架
* 学习矩阵的SVD分解

# 二.操作环境

## 2.1 QT图形化编程

IDE：Microsoft Visual Studio 2019 community

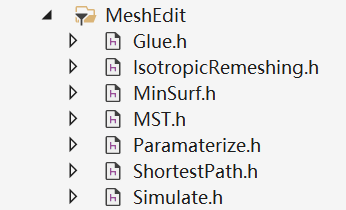
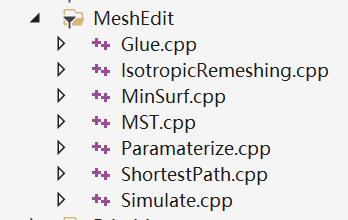
QT: 5.12.12

Cmake：3.23.1

UEngine

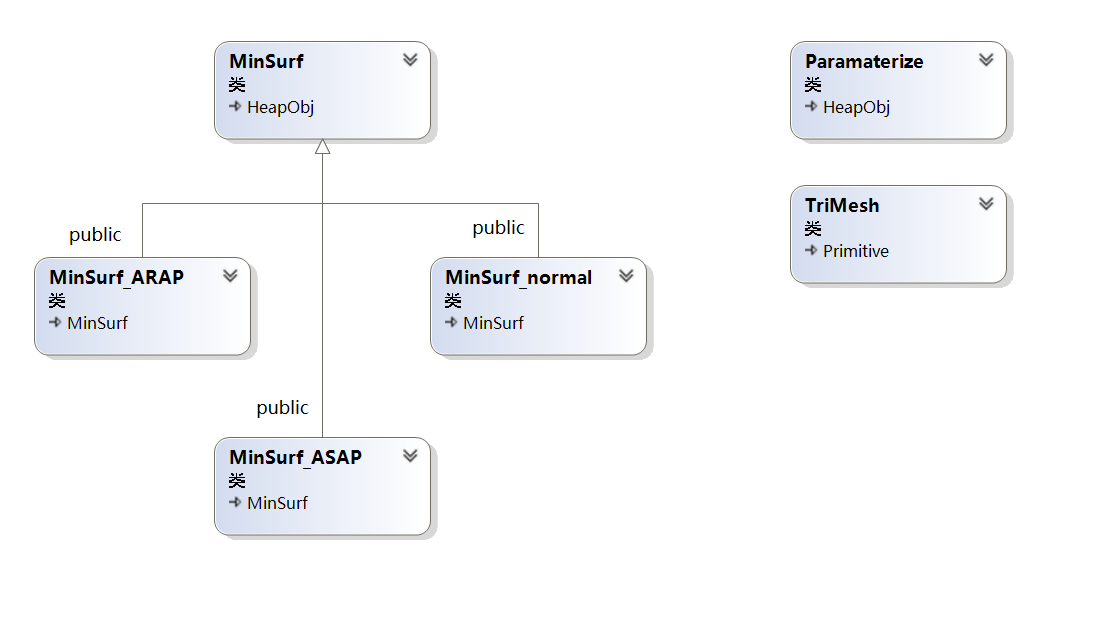
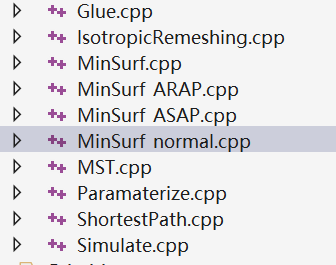
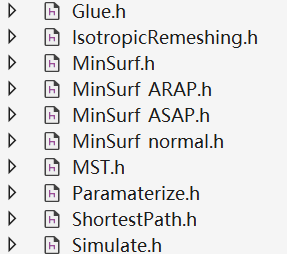
# 三.架构设计

## 3.1 文件结构

和第四次作业的结构基本相同，我本来的设想是用之前的Minsurf类作为基类，根据极小曲面、ASAP、ARAP等不同方法继承自Minsurf类，但在处理Minsurf的继承和实例化的时候出现了问题，因为minsurf还继承自HeapObj类，由此和Heapobj中的一些方法产生了冲突，不能直接实例化，无奈只能将ASAP和ARAP放在了Minsurf中分别作为一种方法来实现。

\*更新：问题已解决，需要重写构造函数和New函数，按照MinSurf的模板写，而且对于相同的函数直接重写函数就行，不用虚函数。现在如下



## 3.2 类图

总体上和第四次作业差距不大，多添加了ASAP,ARAP,one\_way\_ARAP方法，ASAP方法用来实现ASAP算法，ARAP方法用来实现ARAP，其中每一次迭代都调用one\_way\_ARAP方法。acos，distance方法用来简化计算。H文件中的cot和x是初始化后存用于计算的和的值。

# 四.功能实现

总的来说，算法的精髓是希望保留每个面三角形的局部性质，通过求解方程组来解决整体性质。先做一些声明：表示在作为第t个三角形的顶点的点i，表示参数化后第i个点的位置，注意他们俩都是二维向量。表示第(i,j)半边对应的角度，表示（i，j）半边对应的三角形对应矩阵（2\*2）。根据文章，我们直接把能量表示为

即要通过求解方程组来求出能量的最小值。其中对于，ASAP取旋转伸缩，ARAP取旋转。

**4.1 ASAP**

第一步：将已知的Mesh全等的映到二维平面（注意三角形之间的位置不影响结果，只要保证同一个三角形内坐标的差值不变就行），可以先固定一个点，求出和它相连的两条边的长度，和夹着的夹角，这样可以在二维坐标中建立一个全等的三角形。同时，我们可以把每个角的cot值计算出来备用。

第二步：选定锚点：对于ASAP算法，如果没有固定的两个点，很可能会解出全部为0（平凡解），因为这时面积为0，能量最小，所以我们需要选定两到三个锚点。锚点的选取采用boundary的第一个点和距离他最远的那个边界点。

第三步：矩阵赋值：能量中共有2\*nV+2\*nT个变量（nV是顶点数量，nT是面数量），其中两个锚点已经经过固定。设矩阵的形式为。我们对每一个半边循环，对于一个半边，能量涉及到了六个变量：，其中1，2表示两个坐标。则一个半边最多改变矩阵中的六个位置，对这六个位置进行更新就行。设分别表示。一个半边涉及到能量为

对求导：

对求导：

对求导：

对求导：

对求导：

对求导：

按照系数调整矩阵即可。

第四步：求解矩阵，更新坐标。

**4.2 ARAP**

ARAP要先获取一个初始化的展开，在这里采用了ASAP，一开始我使用了第四次作业的极小曲面方法，但会导致三角形发生反转（个别样例），于是使用ASAP作为初始化结果，也减少了迭代次数。与ASAP不同的是，矩阵要满足

第一步：初始化。直接ASAP就行，顺便也把x和cot初始化了

第二步：选定锚点。注意这时因为变换是保大小的，所以不能随意确定两个点的值，会导致错误的压缩或反转，理论上确定的两个点的距离应该和整个模型有关，但这里采用只使用一个锚点（因为保大小所以不会出现平凡解），保证没有锚点干扰结果是最优的。

第三步：求解矩阵，SVD分解。

对矩阵S进行SVD分解：

得到SVD分解

取 , 得到并保存（这里所有的行列式都是1）

第四步：矩阵赋值。和ASAP基本相同，注意因为是固定，所以所有和有关的都要放到右边，即矩阵方程Ax=b的b中。

第五步 更新坐标。

第六步 检测迭代条件，决定是否开始下一次迭代。

# 五.难点难题

1.可能是namespace冲突的问题，在Minsurf文件里没法使用vector，只能用QVector，QVector访问需要用[\*][\*]的形式，而不是at()函数，给我带来了一些困扰。

\*更新：需要指定std空间，而不是直接using namespace std

2.处理对称矩阵：要保证能使用LDLT分解，必须保证矩阵是对称的，就要对两个锚点进行特殊处理，让矩阵变成对称的。

3.对每一个半边进行操作会导致要对矩阵同一个位置做多次操作，但如果是系数矩阵的insert()函数只能调用一次，所以要用一个vector提前存矩阵。

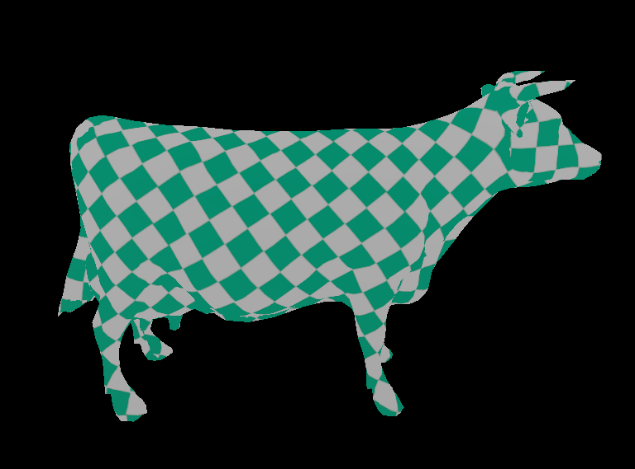
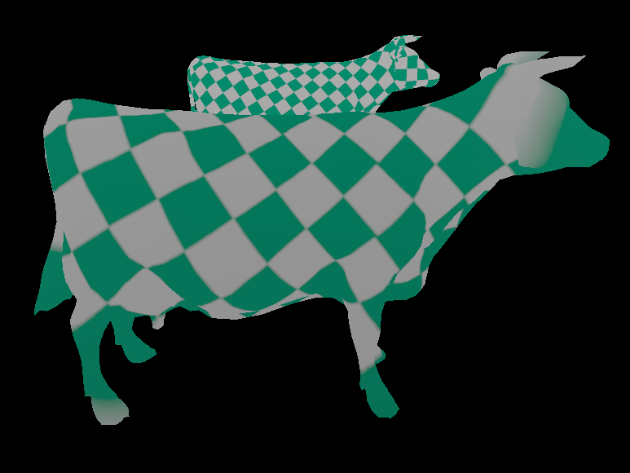
4.后来才注意到cot和x的值应该从一开始就保持不变，因为我们希望它和原网格尽可能的像，而不是在每次迭代都更新一遍，可能会导致平凡解。锚点也是这样。

# 六.实验结果

**Beetle：**

****

**Cow：**



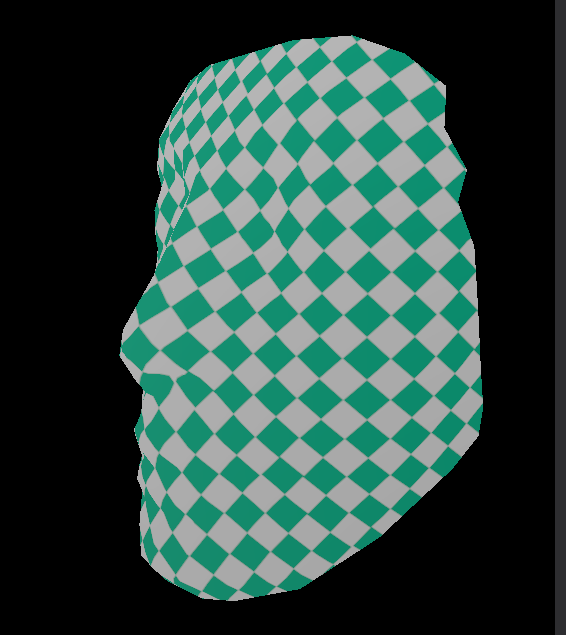
**Gargoyle：**



**isis**：



**Face：**



**合影：**

# 

可以看到相比于普通的极小曲面他们的表现已经很好了，ASAP相比ARAP在处理尖锐的边界（锥点）的时候会发生扭曲，导致在尖端被拉伸的很严重，ARAP看起来就均匀了很多。

# 七.问题与展望

## 7.1 遇到的问题

* ARAP和ASAP在应对点的数量很多的样例时时间过长
* ASAP的调用太占用内存
* 计算过于繁琐，复杂度比较高
* 类的设置，对于Minsurf类的继承

## 7.2 future work

* 缩短时间，提高效率
* 更好的UI界面
* 更好的类结构
* 多边界的样例