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

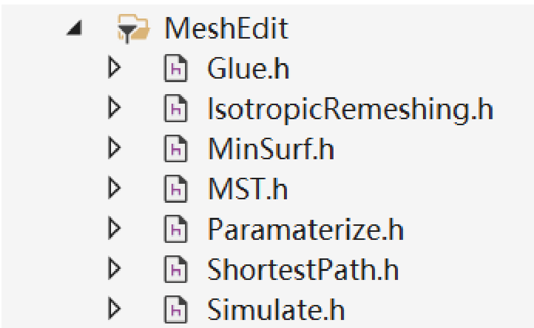
QT： 5.12.12

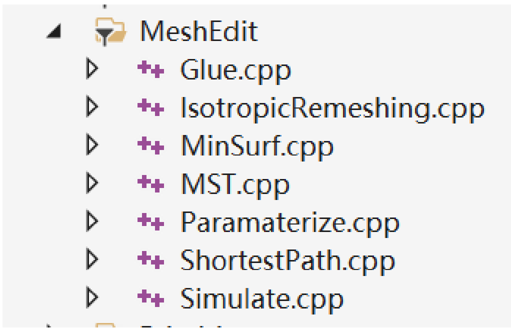
Cmake：3.23.1

UEngine

三. 架构设计

3.1 文件结构

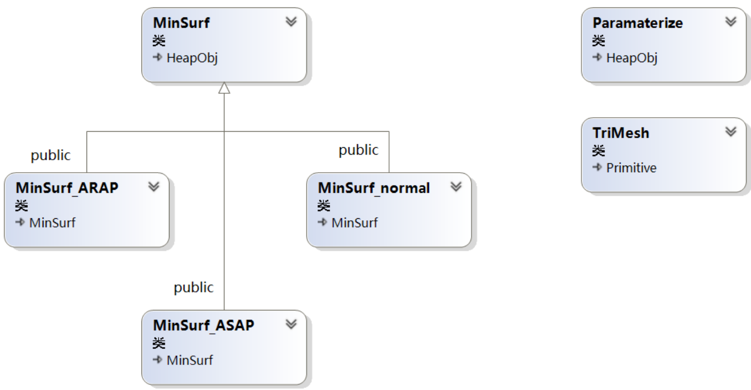




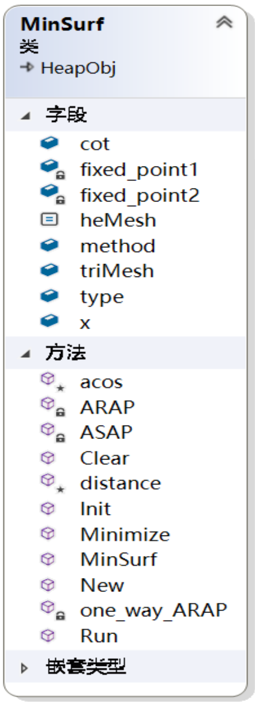
和第四次作业的结构基本相同，我本来的设想是用之前的 Minsurf 类作为基类，根据极小曲面、ASAP、ARAP 等不同方法继承自 Minsurf 类，但在处理 Minsurf 的继承和实例化的时候出现了问题，

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cot_{ij}和x_t^i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：x_t^i表示在作为第 t 个三角形的顶点的点 i，u_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。$\cot(\theta_{ij})$表示第(i,j)半边对应的角度，$L_{t(i,j)}$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

$$\frac{1}{2}\sum_{(i,j)\in he}\cot(\theta_{ij})\left\|\left(u_i-u_j\right)-L_{t(i,j)}(x_t^i-x_t^j)\right\|^2$$

即要通过求解方程组来求出能量的最小值。其中对于$L_{t(i,j)}$，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 $\begin{pmatrix} a_t & b_t \\ -b_t & a_t \end{pmatrix}$。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：$u_{i1}, u_{i2}, u_{j1}, u_{j2}, a_t, b_t$，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 $\Delta_t x_{ij}^1, \Delta_t x_{ij}^2$分别表示$\left(x_t^{i1} - x_t^{j1}\right)$，$\left(x_t^{i2} - x_t^{j2}\right)$。一个半边涉及到能量为

$$\frac{1}{2}\cot(\theta_{ij})(\left((u_{i1} - u_{j1}) - (a_t\Delta_t x_{ij}^1 + b_t\Delta_t x_{ij}^2)\right)^2 + \left((u_{i2} - u_{j2}) - (a_t\Delta_t x_{ij}^2 - b_t\Delta_t x_{ij}^1)\right)^2)$$

对u_{i1}求导：$\cot(\theta_{ij})\ \left((u_{i1} - u_{j1}) - (a_t\Delta_t x_{ij}^1 + b_t\Delta_t x_{ij}^2)\right)$

对u_{i2}求导：$\cot(\theta_{ij})\ \left((u_{i2} - u_{j2}) - (a_t\Delta_t x_{ij}^2 - b_t\Delta_t x_{ij}^1)\right)$

对u_{j1}求导：$-\cot(\theta_{ij})\left(\left(u_{i1}-u_{j1}\right)-\left(a_t\Delta_t x_{ij}^1+b_t\Delta_t x_{ij}^2\right)\right)$

对u_{j2}求导：$-\cot(\theta_{ij})\left(\left(u_{i2}-u_{j2}\right)-\left(a_t\Delta_t x_{ij}^2-b_t\Delta_t x_{ij}^1\right)\right)$

对a_t求导：$\cot(\theta_{ij})\left(\left(-\Delta_t x_{ij}^1\right)\left(\left(u_{i1}-u_{j1}\right)-\left(a_t\Delta_t x_{ij}^1+b_t\Delta_t x_{ij}^2\right)\right)+\left(-\Delta_t x_{ij}^2\right)\left(\left(u_{i2}-u_{j2}\right)-\left(a_t\Delta_t x_{ij}^2-b_t\Delta_t x_{ij}^1\right)\right)\right)$

对b_t求导：$\cot(\theta_{ij})\left(\left(-\Delta_t x_{ij}^2\right)\left(\left(u_{i1}-u_{j1}\right)-\left(a_t\Delta_t x_{ij}^1+b_t\Delta_t x_{ij}^2\right)\right)+\left(-\Delta_t x_{ij}^1\right)\left(\left(u_{i2}-u_{j2}\right)-\left(a_t\Delta_t x_{ij}^2-b_t\Delta_t x_{ij}^1\right)\right)\right)$

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${a_t}^2+{b_t}^2=1$

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：

$$S_t(u)=\sum_{i=0}^{2}\cot(\theta_t^i)(u_t^i-u_t^{i+1})(x_t^i-x_t^{i+1})^T$$

得到 SVD 分解

$$S_t(u) = U\Sigma V^T$$

取$L_t = UV^T$ ，得到a_t, b_t并保存（这里所有的L_t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L_t，所以所有和L_t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

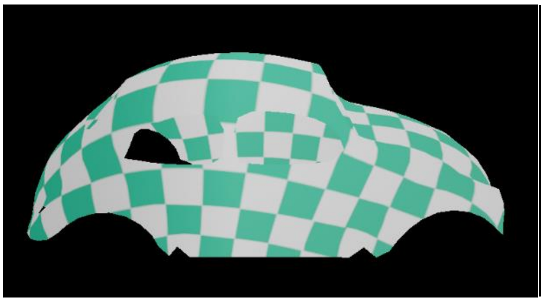
2. 处理对称矩阵：要保证能使用 LDLT 分解，必须保证矩阵是对称的，就要对两个锚点进行特殊处理，让矩阵变成对称的。

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。

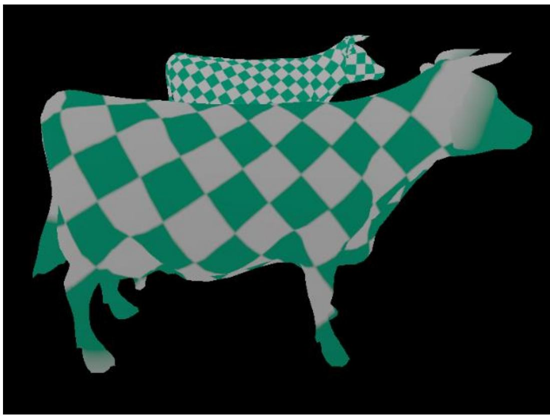
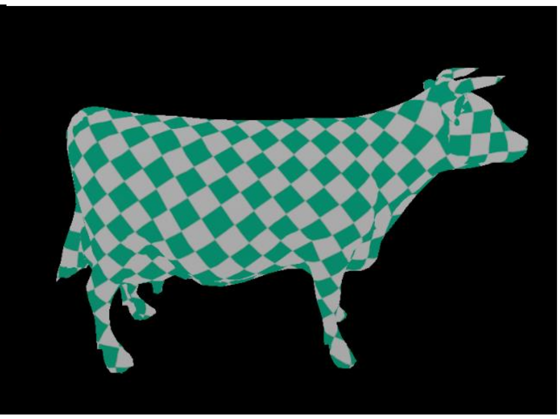
4. 后来才注意到 cot 和 x 的值应该从一开始就保持不变，因为我们希望它和原网格尽可能的像，而不是在每次迭代都更新一遍，可能会导致平凡解。锚点也是这样。

六.实验结果

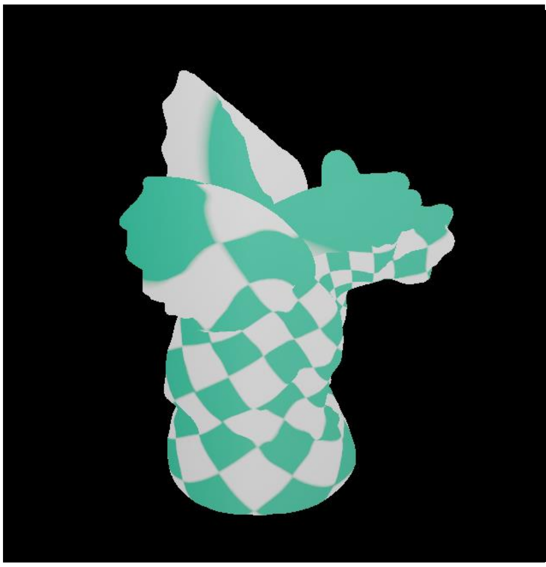
Beetle:




Cow:

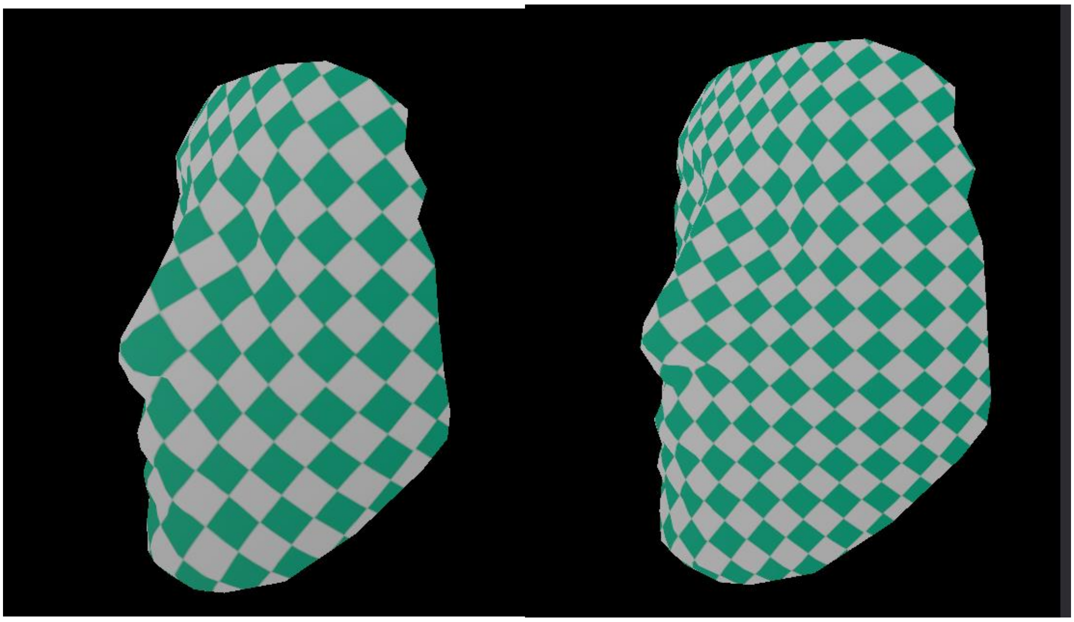



Gargoyle:

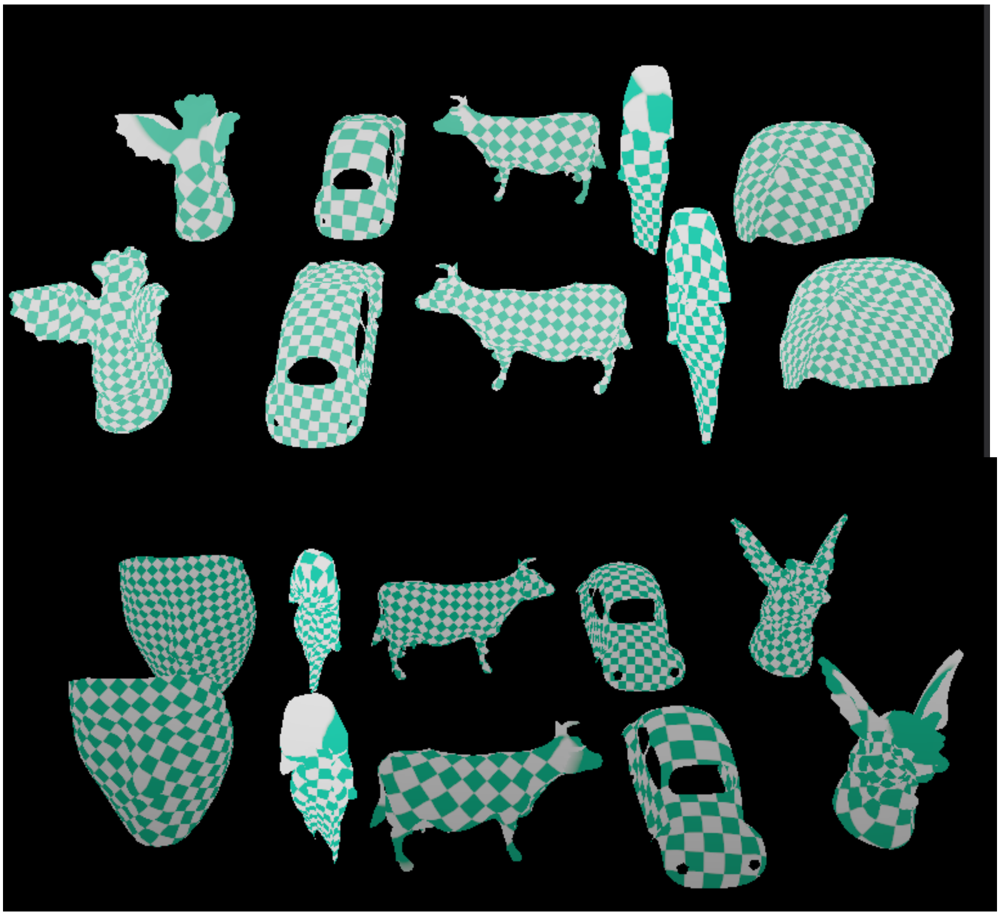



isis:

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