第五次作业 ASAP, ARAP参数化算法 报告

李喆昊 PB17050941

目录

- 算法原理
  - 不固定边界参数化的一般原理
  - As Similar As Possible (ASAP) 算法
  - As Rigid As Possible (ARAP)算法
- 主要问题与代码设计
  - 初始化
  - ASAP算法
  - ARAP算法
- 测试结果
- 总结与反思

一. 算法原理

1. **不固定边界参数化的一般原理**

   不固定边界参数化的本质是：最小化衡量参数化对图形的扭曲程度的能量函数。

   引入辅助线性变换L_t, 可以这样定义能量函数：

   $$E(u,L)=\sum_{t=1}^{T}A_t\|J_t(u)-L_t\|_F^2 \tag{1}$$

   对其整理变形，得：

   $$E(u,L)=\frac{1}{2}\sum_{t=1}^{T}\sum_{i=0}^{2}cot(\theta_t^i)\|(u_t^i-u_t^{i+1})-L_t(x_t^i-x_t^{i+1})\|^2 \tag{2}$$

   这里θ_t^i是原3D网格中第t个三角形中边$u_t^iu_t^{i+1}$所对的角。x_t^i,x_t^{i+1}是将三角形全等映射到2D平面后u_t^i,u_t^{i+1}的对应顶点。

   因此不固定边界参数化的过程就是寻找合适的(u,L), 满足：

   $$(u,L)=argmin_{(u,L)}E(u,L),s.t.L_t\in M \tag{3}$$

   其中M是我们定义的所期望的线性变换族。

2. **As Similar As Possible (ASAP) 算法**

   ASAP算法的目标是最大化的保持参数化后角度的不变性。因此这里我们所期望L_t所在的线性变换族M具有如下这种形式：

   $$M=\left\{\begin{matrix}a & b\\ -b & a\end{matrix}\right\},a,b\in\mathbf{R} \tag{4}$$

   因此(2)式可以表示为：

$$E(u,L)=\frac{1}{2}\sum_{t=1}^{T}\sum_{i=0}^{2}cot(\theta_t^i)\|(u_t^i-u_t^{i+1})-\left\{\begin{matrix} a & b \\ -b & a \end{matrix}\right\}(x_t^i-x_t^{i+1})\|^2 \tag{5}$$

为了求出u,a,b，分别求$\frac{\partial E}{\partial u}$, $\frac{\partial E}{\partial L_t}$,得：

$$\sum_{j\in N(i)}[cot(\theta_{ij})+cot(\theta_{ji})](u_i-u_j)=\sum_{j\in N(i)}[cot(\theta_{ij})L_{t(i,j)}+cot(\theta_{ji})L_{t(j,i)}](x_i-x_j),\forall i=1,\ldots,n \tag{6}$$

由于对L_t求导，在得到的式子中t可以略去，记$\Delta u^i:=(u_t^i-u_t^{i+1})$, $\Delta x^i:=(xt^i-x_t^{i+1})$，得：

$$\begin{aligned} &\frac{\partial E}{\partial L_t}= \\ &\left\{\begin{matrix} \frac{\partial E}{\partial a} & \frac{\partial E}{\partial b} \\ -\frac{\partial E}{\partial b} & \frac{\partial E}{\partial a} \end{matrix}\right\}= \\ &\sum_{i=0}^{2}cot(\theta_t^i)(\Delta u_t^i-L_t\Delta x_t^i)\Delta x_t^{i\,T}= \\ &\sum_{i=0}^{2}cot(\theta^i)\left\{\begin{matrix} \Delta u_x^i\Delta x_x^i & \Delta u_x^i\Delta x_y^i \\ \Delta u_y^i\Delta x_x^i & \Delta u_y^i\Delta x_y^i \end{matrix}\right\}-\left\{\begin{matrix} a & b \\ -b & a \end{matrix}\right\}\left\{\begin{matrix} \Delta x_x^i\Delta x_x^i & \Delta x_x^i\Delta x_y^i \\ \Delta x_y^i\Delta x_x^i & \Delta x_y^i\Delta x_y^i \end{matrix}\right\}= \\ &\sum_{i=0}^{2}cot(\theta^i)\left\{\begin{matrix} \Delta u_x^i\Delta x_x^i-a\Delta x_x^i\Delta x_x^i-b\Delta x_y^i\Delta x_x^i & \Delta u_x^i\Delta x_y^i-a\Delta x_x^i\Delta x_y^i-b\Delta x_y^i\Delta x_y^i \\ \Delta u_y^i\Delta x_x^i+b\Delta x_x^i\Delta x_x^i-a\Delta x_y^i\Delta x_x^i & \Delta u_y^i\Delta x_y^i+b\Delta x_x^i\Delta x_y^i-a\Delta x_y^i\Delta x_y^i \end{matrix}\right\} \end{aligned} \tag{7}$$

整理可得：

$$\begin{aligned} &\frac{\partial E}{\partial a}=\frac{C_2}{C_1} \\ &\frac{\partial E}{\partial b}=\frac{C_3}{C_1} \\ &where:\ C_1=\sum_{i=0}^{2}cot(\theta^i)[(\Delta u_x^i)^2+(\Delta u_y^i)^2] \\ &C_2=\sum_{i=0}^{2}cot(\theta^i)[\Delta u_x^i\Delta x_x^i+\Delta u_y^i\Delta u_y^i] \\ &C_3=\sum_{i=0}^{2}cot(\theta^i)[\Delta u_x^i\Delta x_y^i-\Delta u_y^i\Delta x_x^i] \end{aligned} \tag{8}$$

按照以上式子建立稀疏方程组即可。

3. **As Rigid As Possible (ARAP)算法**

   ASAP算法的目标是最大化的保持参数化后图形的刚性。ARAP算法的核心是采用了"Local/Global"的迭代方法。

   其过程如下：

   1. 初始化，包含两步：
      - 将3D网格上的三角形全等映射到2D平面上，实现细节见二。
      - 进行初始参数化（可以使用HW4实现的固定边界方法）
   2. Local阶段：固定u,求L_t

      对下式进行SVD分解：

      $$S_t(u)=\sum_{i=0}^{2}cot(\theta_t^i)(u_t^i-u_t^{i+1})(x_t^i-x_t^{i+1})^T=U\Sigma V^T \tag{9}$$

      取

      $$L_t=UV^T \tag{10}$$

   3. Global阶段：固定L_t,求u

求解稀疏方程组：

$$\sum_{j\in N(i)}[cot(\theta_{ij})+cot(\theta_{ji})](u_i-u_j)=\sum_{j\in N(i)}[cot(\theta_{ij})L_{t(i,j)}+cot(\theta_{ji})L_{t(j,i)}](x_i-x_j),\forall i=1,\dots,n \quad (11)$$

4. 重复2，3步，直到结果收敛或达到指定步数。

二. 主要问题与代码设计

1. **初始化：**

   问题：如何将3D网格全等映射到2D平面上？

   我采用的方法是：

   对某一个3D三角形，设其3个顶点为u_0, u_1, u_2, $u0u1, u0u2$的夹角为θ, 则映射后的x_0, x_1, x_2坐标为：

   $$x_1:(0,0,0),\ x_2:(\overline{u_0u_1},0,0)\ x_2:(\overline{u_0u_2}cos\theta,\overline{u_0u_2}sin\theta,0) \quad (12)$$

   使用c++的`vector<map<V*, pointf2>>`进行哈希存储。

   值得注意的是，这样映射多个3D三角形映射后会重合，而且同一个点在不同三角形中映射得到的坐标也不同，但因为算法中关注的是映射后三角形内局部的位置关系，所以这两点对算法没有影响。

2. **ASAP算法**

   问题1：稀疏方程组的参数如何设置？

   如算法原理中所示。我设置系数矩阵的维度为$[2\times(nV+nT)]$，其中nT为mesh中三角形的数量。前$2\times nV$行存放u_x, u_y的方程，后$2\times nT$行存放a, b的方程。

   问题2：锚点如何选取？

   选取相隔距离尽可能远的两个边界上的点作为锚点。代码如下：

```
auto HE_boundaries = heMesh->Boundaries()[0];
auto he1 = HE_boundaries[0];
auto he2 = HE_boundaries[HE_boundaries.size() / 2];
auto v1 = he1->Origin();
auto v2 = he2->Origin();
```

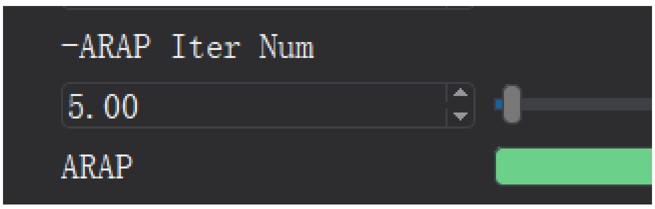
3. **ARAP算法**

   问题1：锚点如何选取？

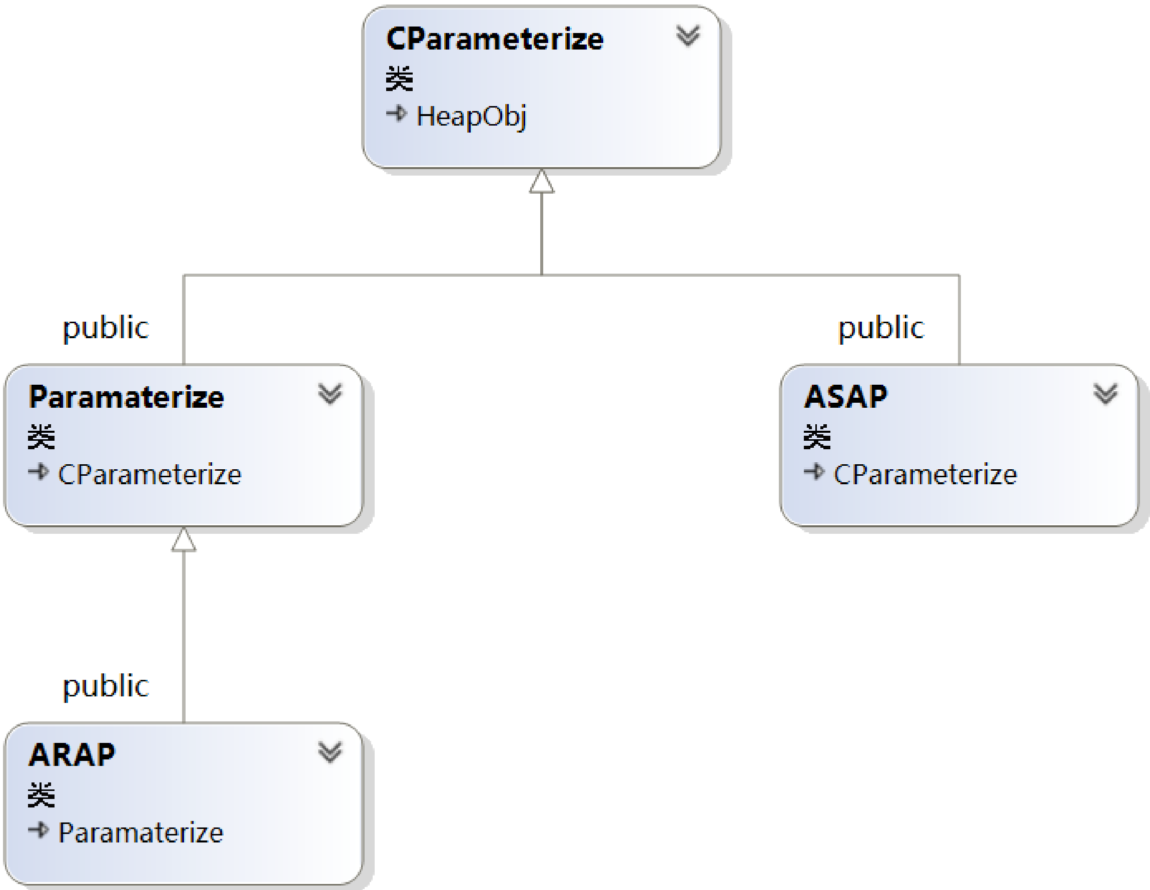
   由于观察发现ARAP算法对于锚点的选取不是很敏感，因此选取3Dmesh中的第一个点作为锚点。

   问题2：迭代如何判断终止？

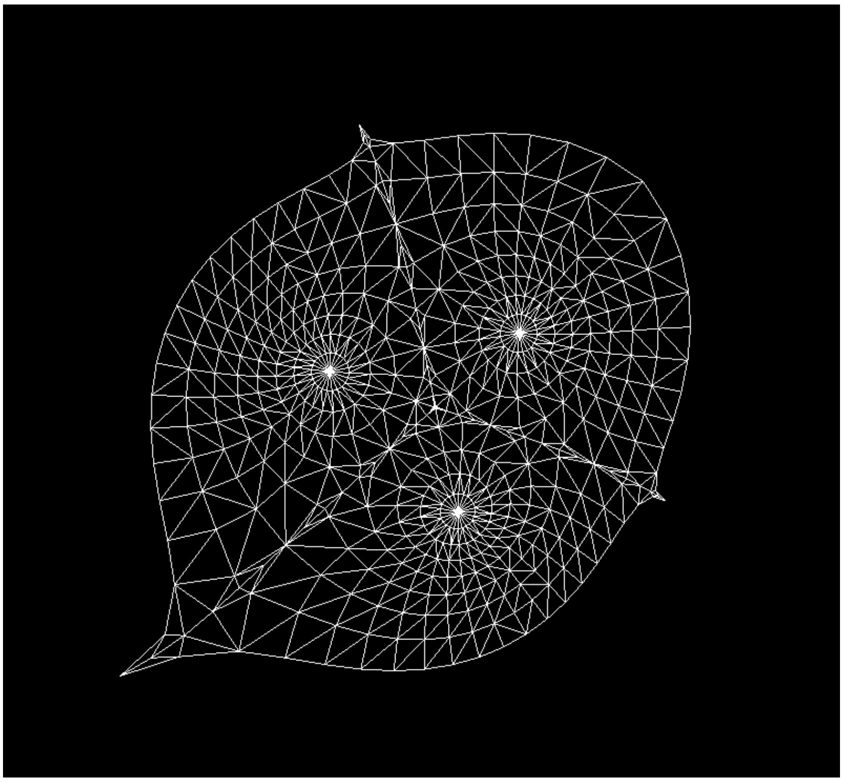
   达到了指定的迭代次数，或者相隔两次迭代，得到的参数化结果点之间的最大距离在设定的误差范围之内。用户可以在UI界面中改变Iteration Num。

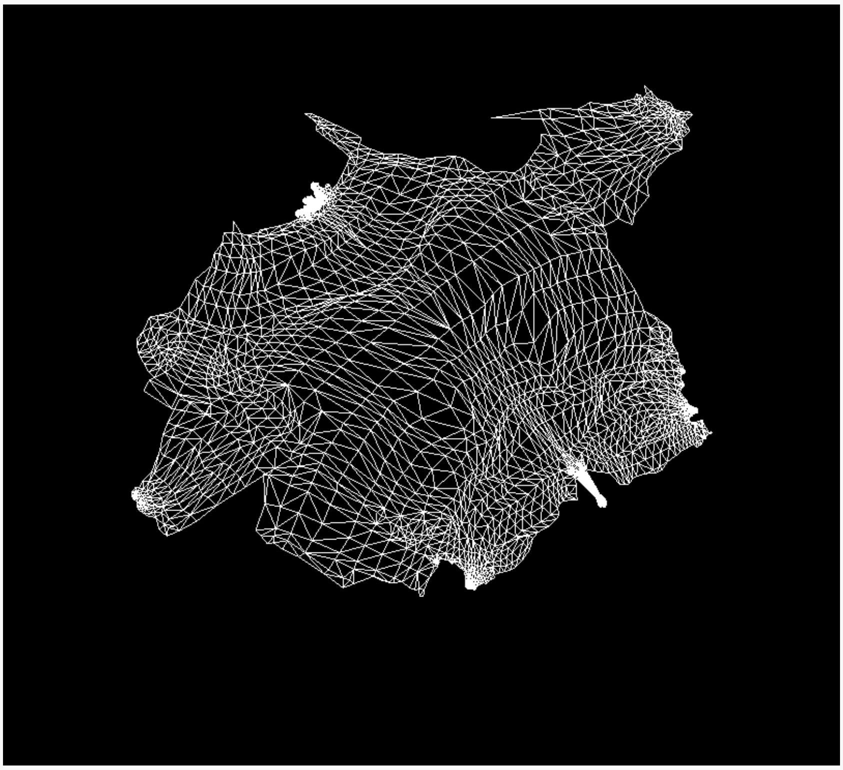
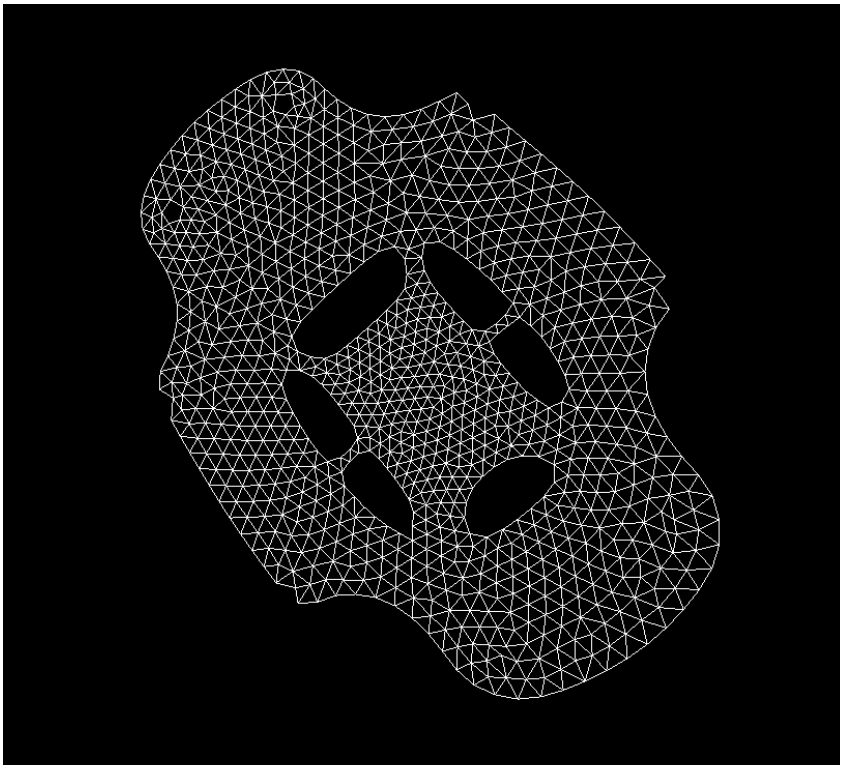


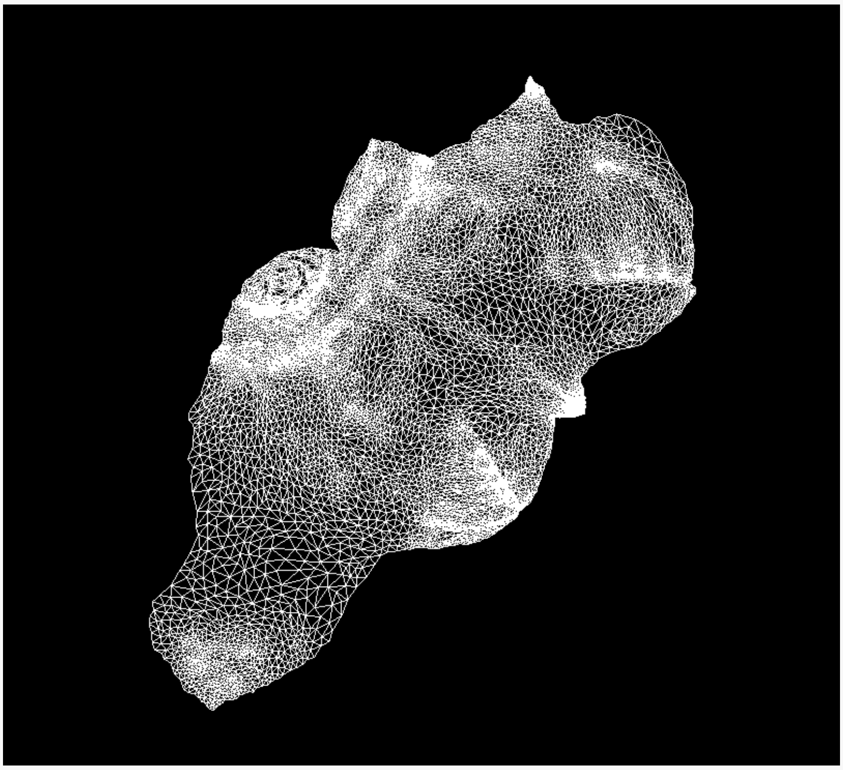
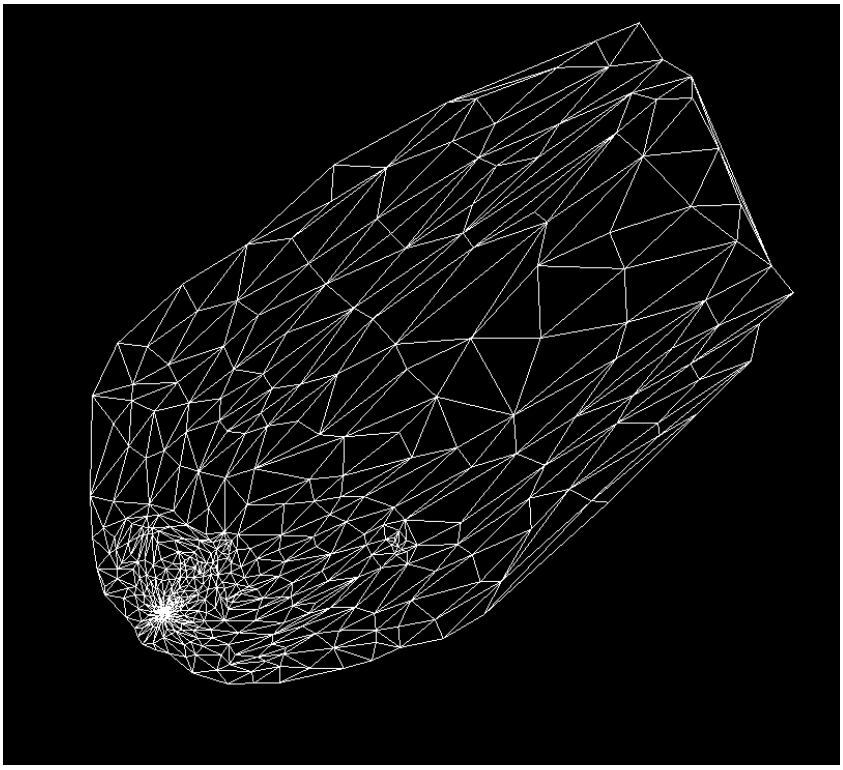
复用了HW4的参数化类，类图如下：(HW4的类为`Paramaterize`， ARAP类使用了HW4的类进行初始化)

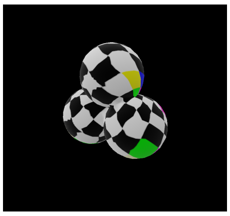
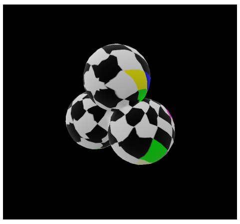
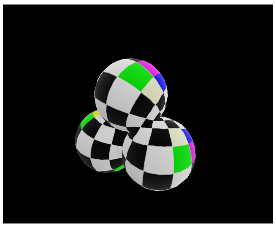
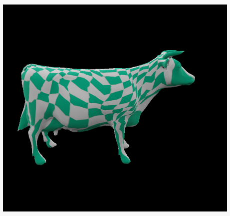
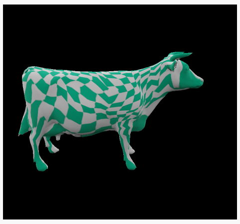
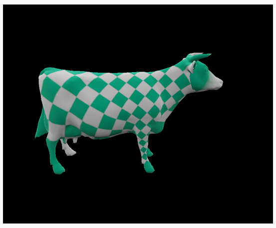
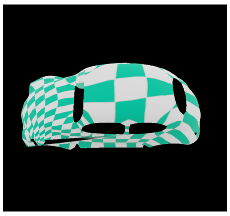
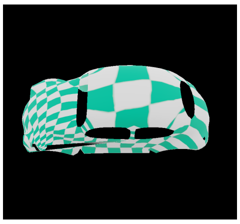
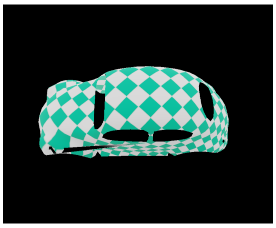
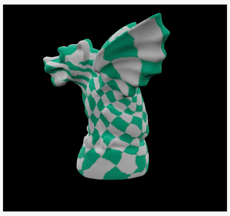
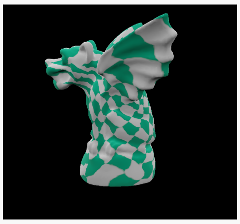
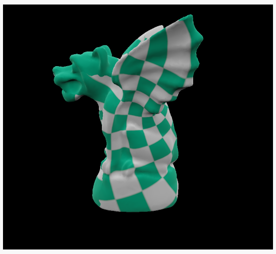
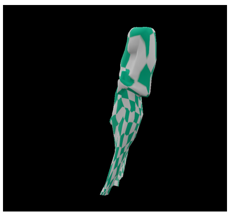
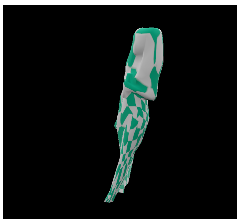
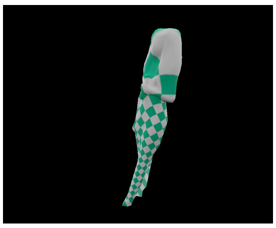


三. 测试结果

obj	ASAP	ARAP
ball		

obj	ASAP	ARAP
cow		
beetle		

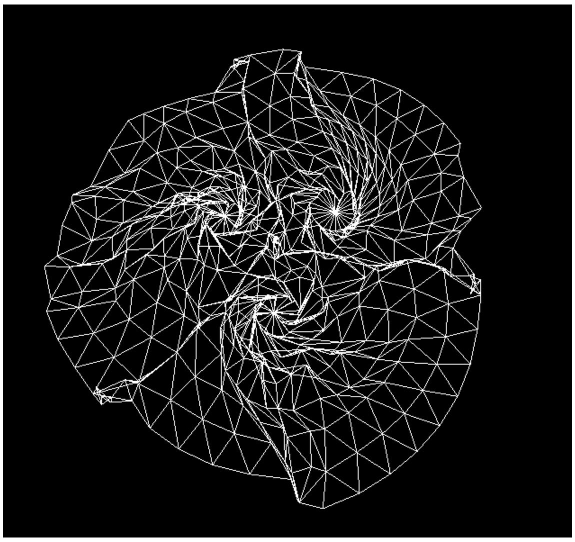
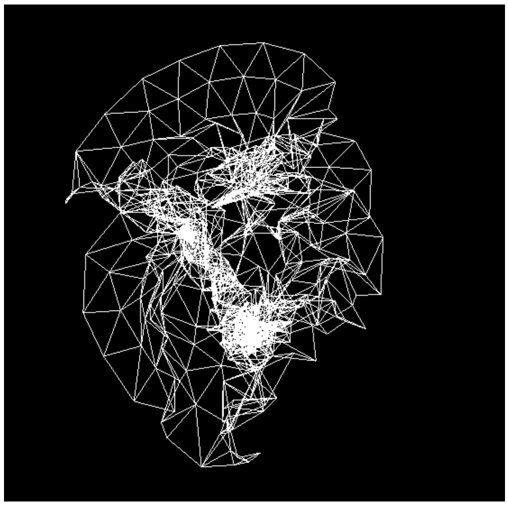
obj	ASAP	ARAP
Gargoyle		
Isis		

obj	度数	cotan	ASAP	ARAP
ball				
cow				
beetle				
Gargoyle				
Isis				

四. 总结反思

本次作业花了很多时间在理解算法原理、对ASAP求导、对ARAP debug上，但是很遗憾的是ARAP图形在迭代中会缩小、扭曲的问题仍然没有解决。

会出现如下图的问题：

迭代1次	迭代10次
	

经过周日全天的大量时间检查代码正确性、进行调试仍然没有解决ARAP的这个问题。

主要的原因有：

- 没有从数学上理解为什么会出现如图中所示的扭曲现象
- 没有弄明白如何将三角形展开使得其不会在迭代中收缩到一点。

期待在之后的时间中将这一问题解决。