Lab 4 MiniSurface&&Parm

Pb17111585 张永停

一、实验内容

- 极小化曲面类：MinSurf.h 和 MinSurf.cpp，在其中完成极小曲面生成算法
- 参数化类：Paramaterize.h 和 Paramaterize.cpp，在其中完成网格参数化算法
  - Uniform weight
  - Cotangent weight
  - 显示纹理映射结果

二、算法描述

（一）极小曲面

- $\delta_i = v_i - \frac{1}{d_i}\sum_{v\in N(i)} v = \frac{1}{d_i}\sum_{v\in N(i)}(v_i - v) = 0$
- 由于边界点固定，从而得到线性方程组

  $$\begin{cases} Lx = \delta = 0 \\ x_i = v_i \ over \ Boundary \end{cases}$$
- 通过将不变的边界点移至方程组右端来构建一个只与内点有关的方程组，使用Eigen的LU分解来解方程

（二）参数化

- 首先将边界固定到平面凸多边形（这里实现的是单位正方形）上
  - 通过将边界点按顺序均匀分布在单位正方形上
  - 前1/4的边界点均匀分布在$x = 0(0 \le y \le 1)$
- 定义嵌入

  $$\begin{cases} Wx = b_x \\ Wy = b_y \end{cases}$$

  其中

  $$w_{ij} = \begin{cases} < 0 & (i,j) \in E \\ -\sum_{j\neq i} w_{ij} & (i,i) \\ 0 & \text{otherwise} \end{cases}$$

- Uniform weight
  - $w_j = 1$
- Cotangent weight (geometry aware)
  - $w_j = (\cot\alpha + \cot\beta)$
- 之后便和极小曲面一样，边界点固定，解关于内点的方程

三、代码框架

（一）极小曲面

```
1 private:
```

```
        Eigen::SparseMatrix<double> L_;                          //方程组系数矩阵
        Eigen::MatrixX3d delta_;                                 //方程组右端矩阵
        Eigen::SparseLU<Eigen::SparseMatrix<double>> LU_;        //使用LU解方程
        Eigen::MatrixX3d X;                                      //方程组的解
        std::vector<V*> inside_points_;                          //存放内点
        std::unordered_map<int,int> map_between_inside_;         //将heMesh中的点
                                                                 //的Index与
                                                                 //inside_points
                                                                 //中点的Index做对
                                                                 //应以方便构建
                                                                 //方程组系数
        int inside_count_;                                       //内点的数目

protected:
        void FindInside();                                       //找到内点
        void GetMap();                                           //构建map
        void BuildLMat();                                        //构建系数矩阵
        void BuildDeltaMat();                                    //构建右端系数，
                                                                 //注意这里并不是
                                                                 //所有delta皆为
0，
                                                                 //我们将边界点的值
                                                                 //移到了右端
        void UpdateMin();                                        //用方程组的解替换
```

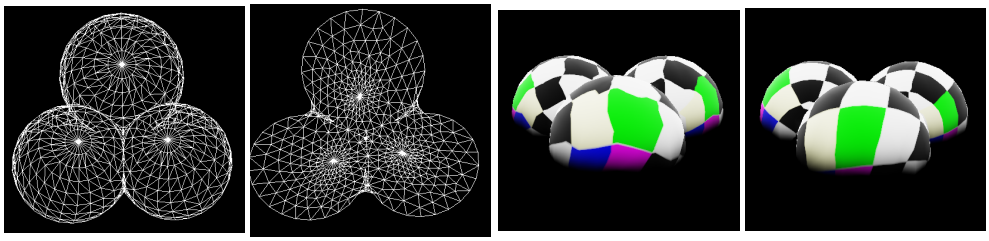
（二）参数化

```
public:
        void SetTex(int m);                                      //设置为纹理填充模式，
                                                                 //同时设置参数化模式
        void SetWMethod(int m);                                  //设置参数化模式
                                                                 //Uniform or CoTan

private:
        void Para();                                             //进行参数化，关键函数
private:
        bool tex_;                                               //是否纹理填充
        int type_;                                               //参数化的模式
        Eigen::SparseMatrix<double> L_;                          //方程组系数矩阵
        Eigen::MatrixX3d delta_;                                 //方程组右端矩阵
        Eigen::SparseLU<Eigen::SparseMatrix<double>> LU_;//使用LU解方程
        Eigen::MatrixX3d X;                                      //方程组的解
        std::vector<V*> inside_points_;                          //存放内点
        std::vector<V*> boundary_points_;                        //存放边界点
        std::unordered_map<int, int> map_between_inside_;//同极小曲面
        int inside_count_;                                       //内点个数
        int boundary_count_;                                     //边界点个数

protected:
        void ParaU();                                            //使用Uniform参数化
        void ParaTan();                                          //使用CoTan参数化
        void FindInside();                                       //查找所有内点
        void SetBoundary();                                      //将边界点固定到正方形
上
        void GetMap();                                           //计算映射
        void UpdatePara();                                       //将方程的解反馈到ui，
                                                                 //根据tex_的值来判断
```

```
30                                                    //是纹理填充还是显示
31                                                    //二维网格
```

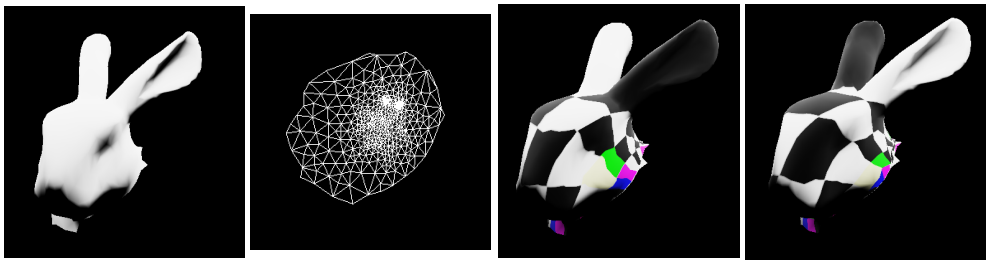
四、实验结果

- Ball

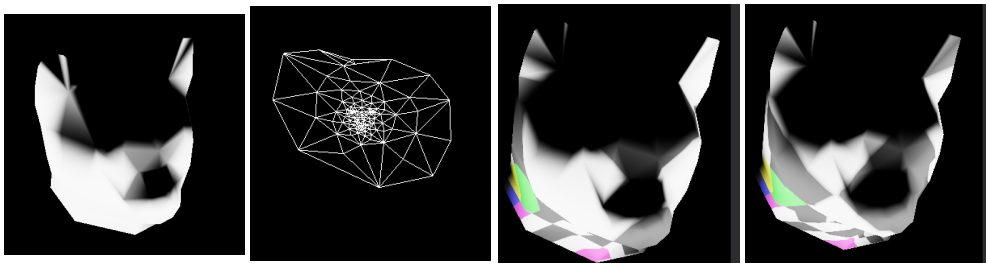


左1:原图 左2:极小化曲面 左3:Uniform参数化 左4:CoTan参数化

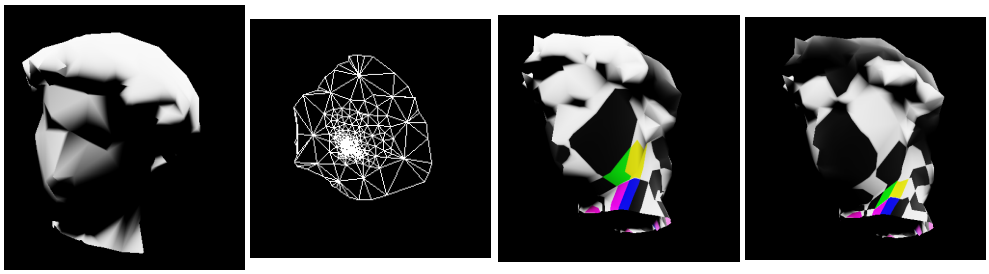
- Bunny Head



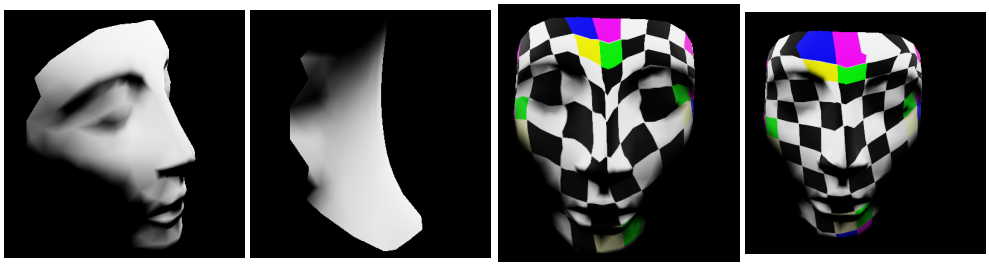
- Cat Head



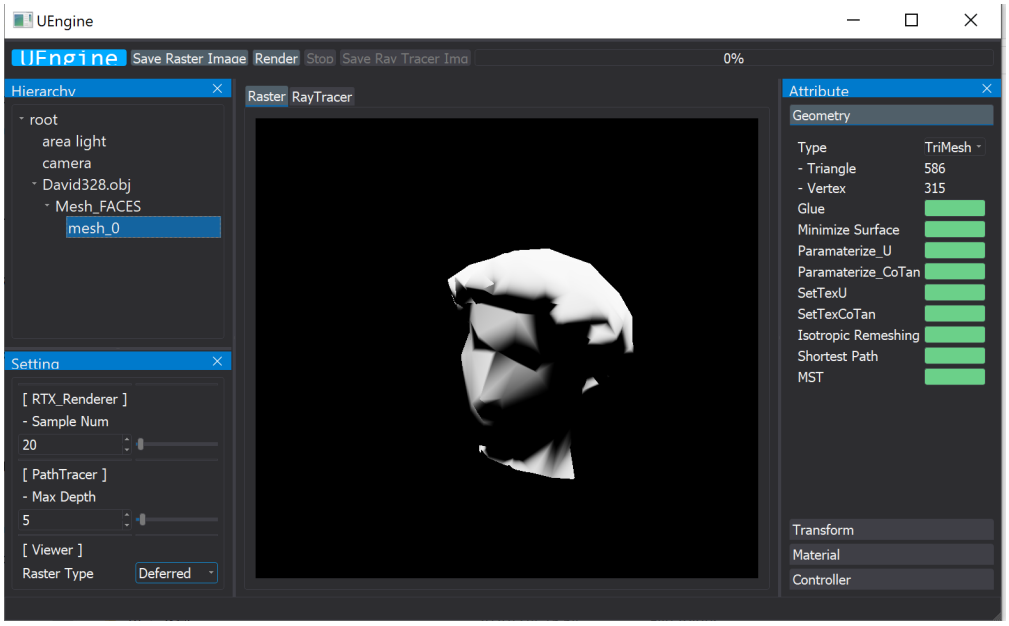
- David



- Nefertiti_face



- 用户界面



右侧工具栏添加了 `Paramaterize_U`，`Paramaterize_CoTan`，`SetTexU`，`SetTexCoTan`，分别实现了，使用Uniform的方法参数化并以二维坐标的形式呈现，使用CoTan的方法参数化并以二维坐标的形式呈现，使用Uniform的方法参数化并将纹理贴在3D图上，使用CoTan的方法参数化并将纹理贴在3D图上

五、实验总结

- 这次实验由于最初对框架不够熟练，走了很多弯路，比如最开始直接使用`heMesh->Vertices()`来获取边界点，结果由于顺序不对，在参数化的时候卡了很久
- 个人感觉比较坑的是参数化固定边界的时候`/`号的使用，由于最开始我是映射到$[0,1] \times [0,1]$的时候是`int/int`,结果就一直不对，后来改成`double`后，由于边界除的比较乱，就导致会有两个点被映射到正方形的同一个顶点上，结果导致使用`cos_theta`的时候报错，并de了好久
- 助教tql！！！！！！