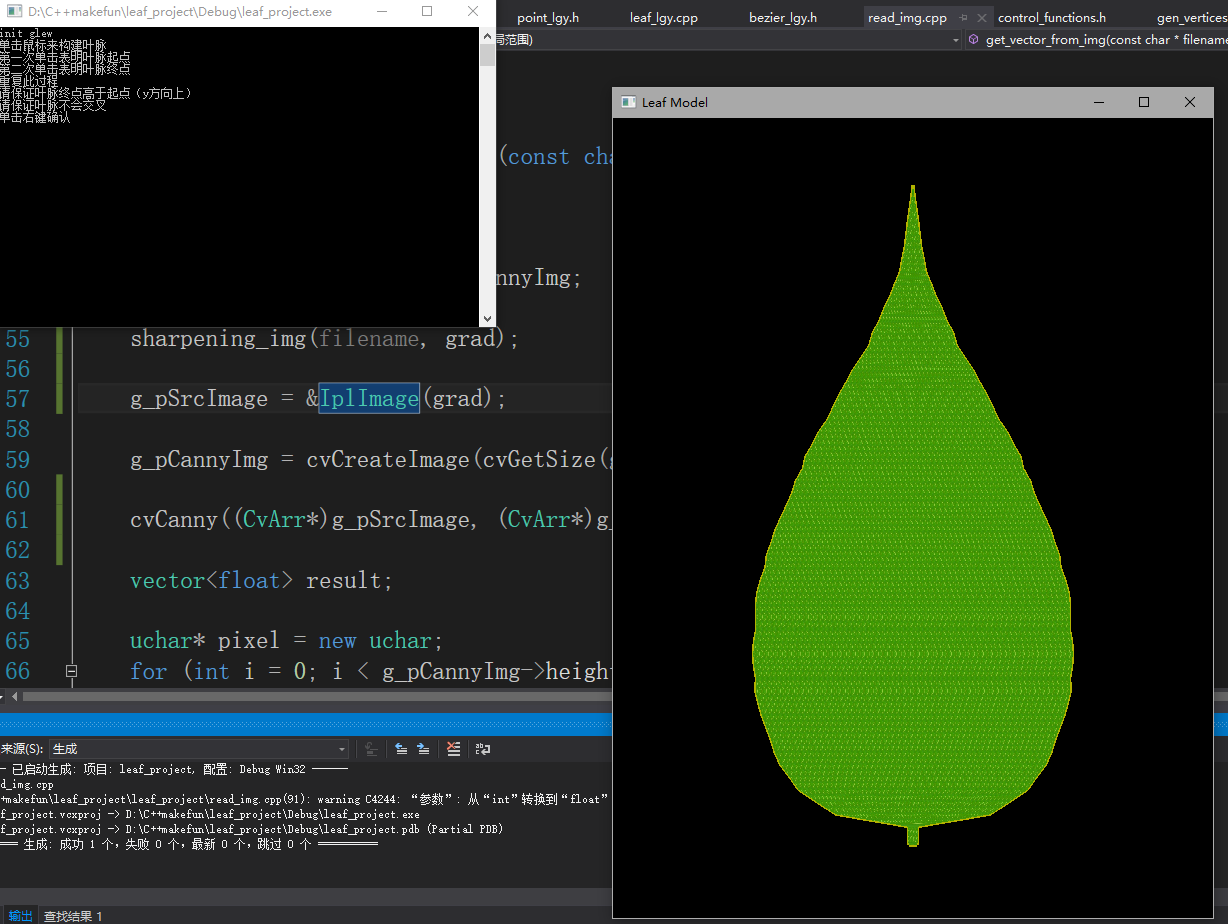
Leaf\_project说明文档

Mail: guanyu.li@outlook.com

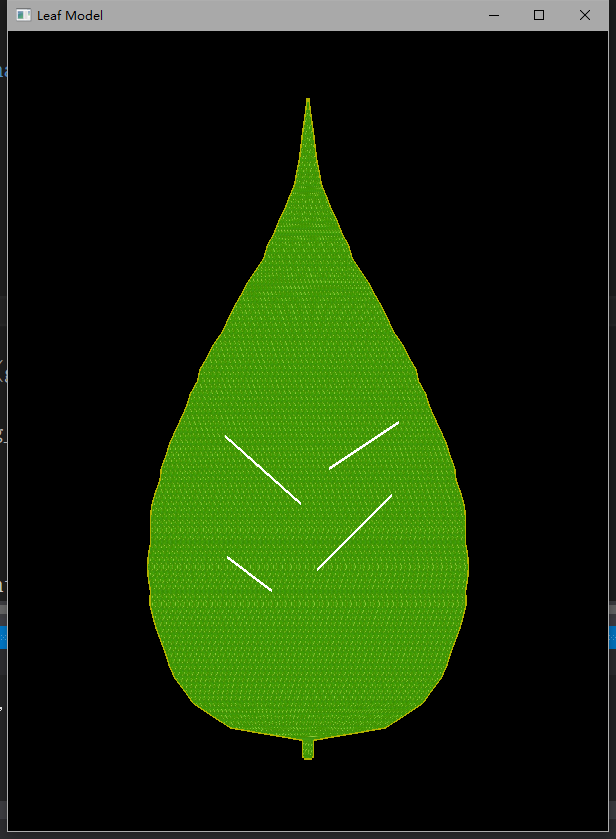
此项目的主要目的是能够通过叶片的图片建立一个叶片模型，并且能够实现交互式变形。基于VS2015，opengl，以及opencv2.4.13。还用到了glew以及glut库。

## 程序说明

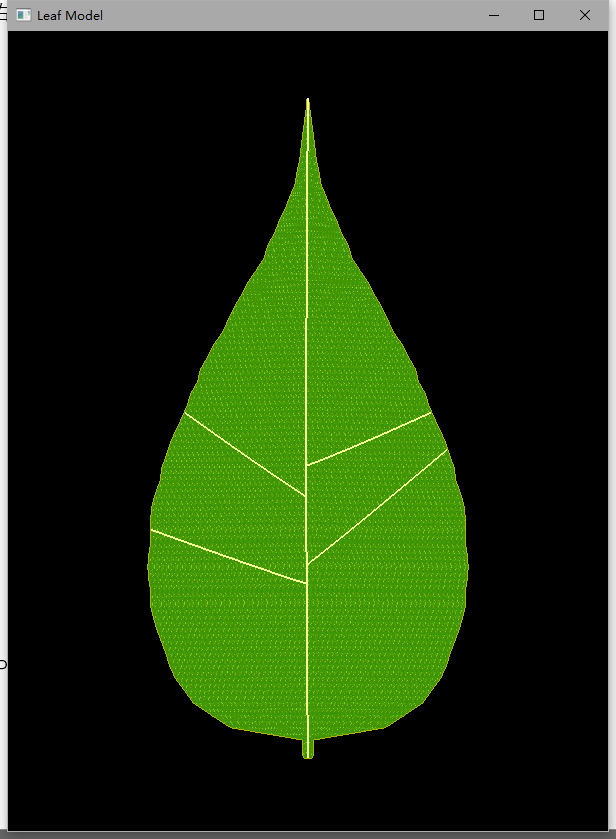
1. 打开leaf\_project.exe, 会自动读取当前目录下名为“leaf\_img.jpg”的文件，并生成一个叶面（提供了3张可以用来做素材的图片，分别叫做leaf\_img.jpg，leaf\_img2.jpg，leaf\_img3.jpg）



2. 在叶面通过点击鼠标左键标明叶脉的起始点和终点（起始点在叶主脉，也就是中心线上），不停创建叶脉，右键单击确认。此时叶脉会创建。

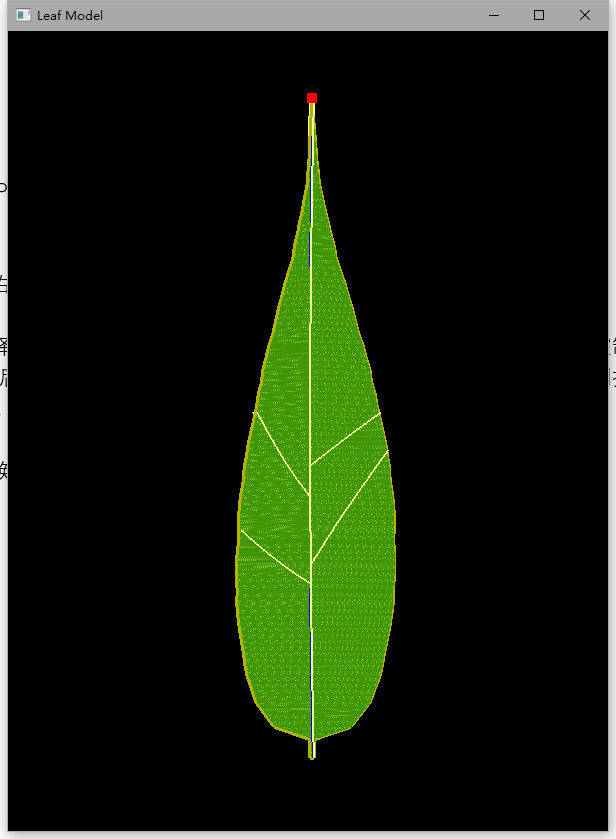


图：单击创建叶脉



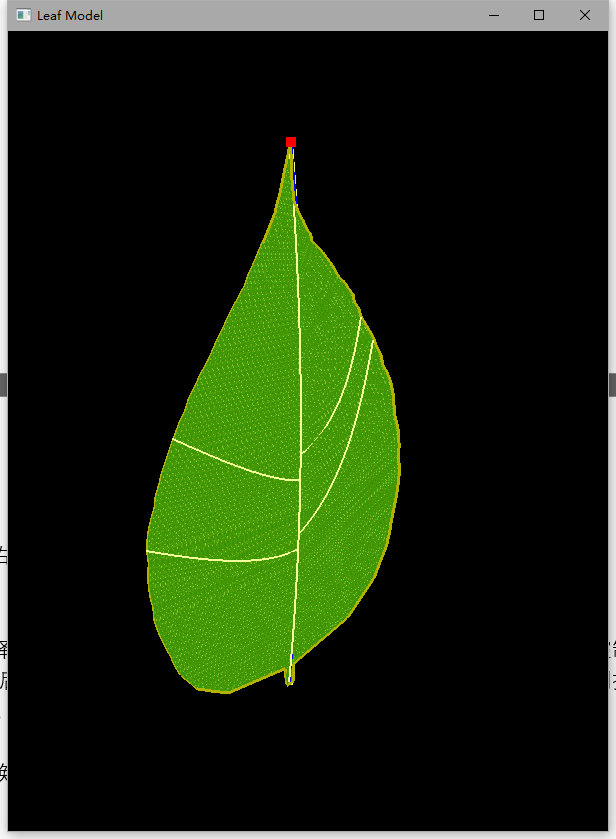
图：单击右键确认

3. 此时按键盘P键可以进行自动预设变形与旋转。



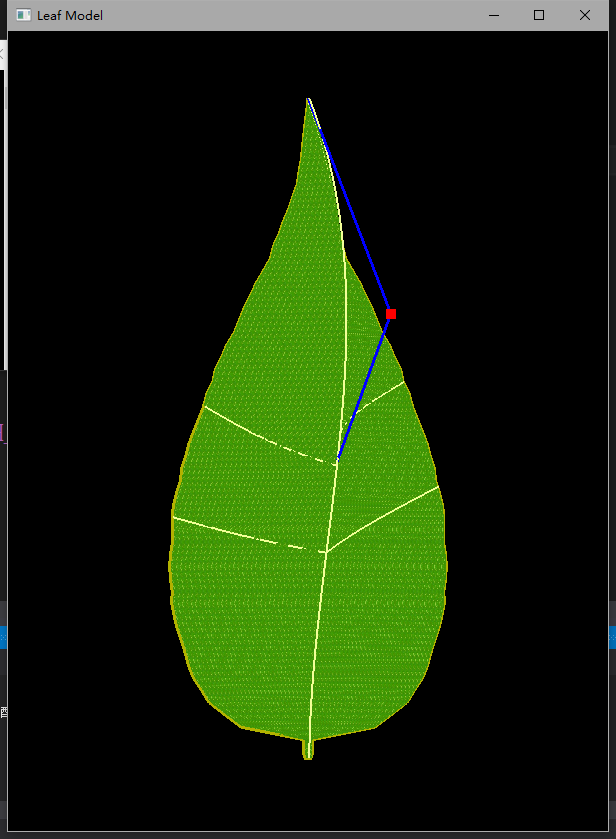
图：变形并自动旋转

4. 键盘上下左右键可以旋转模型

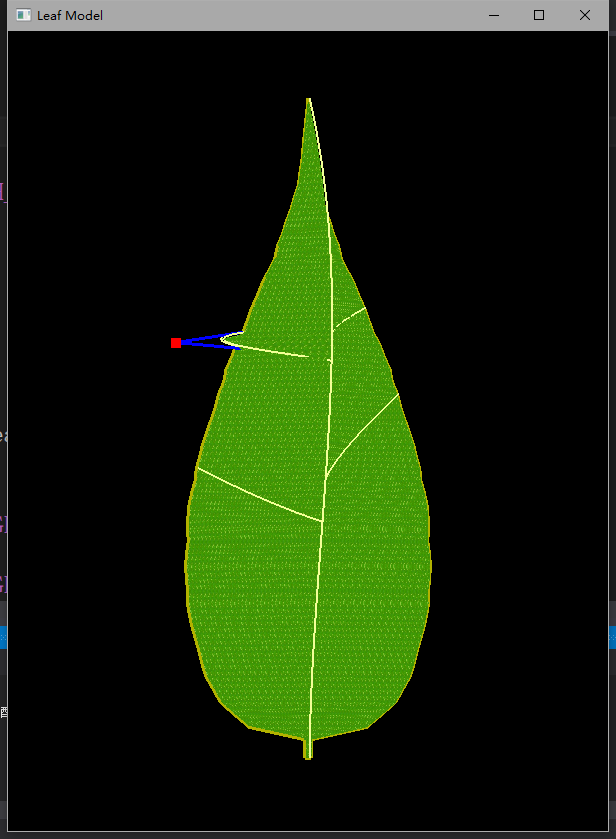


图：旋转

5. 鼠标单击选择叶脉的控制点（叶脉是Bezier曲线），此时会显示此叶脉的4个控制点的连线（蓝色），然后键盘WASD键可分别使控制点上下左右位移，Q和E键可以控制控制点在Z方向的移动。这些移动会使叶面变形。

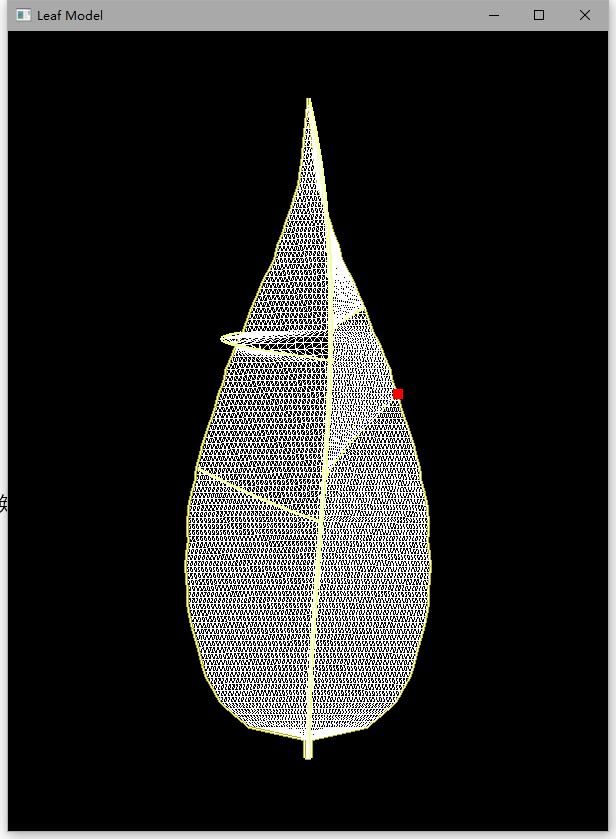


图：主脉控制点z方向位移控制变形



图：侧脉控制变形

6. 键盘T键切换纹理渲染与网格绘图。

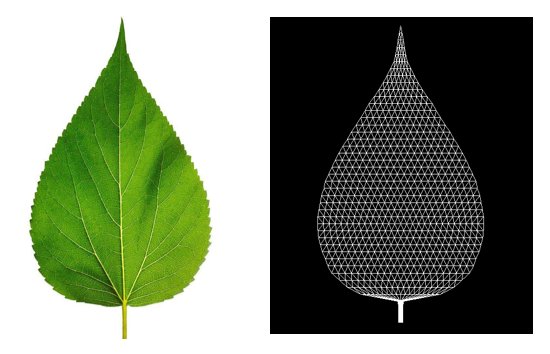


图：网格模式

## 算法思路

1. 图片中提取叶片的算法

主要原理是通过opencv的Sobel()函数实现图片的锐化，再通过cvCanny()函数提取出叶片的轮廓边缘。得到边缘后，逐几行横向扫描，得到叶片每行的宽度，再平均插值建立网格模型。



1. 叶脉生成方法

叶脉和主脉都用4个控制点的Bezier曲线实现。由上面的方法得到的叶面为轴对称图形，取中轴为主脉。侧脉的起始控制点都取主脉上的点，第四个控制点去叶片边缘上的点，第二第三控制点在一四中间平均取。

1. 叶脉控制变形方法。

叶面的变形以每一行上的点为单位进行计算。每一行上有边缘点，主脉点，中间点，以及可能有位于侧脉上的点。

主脉点和侧脉点可以通过主脉和侧脉的Bezier曲线公式根据控制点位置计算出坐标。

叶边缘的形状坐标由主脉的控制点以及侧脉的四号控制点的位置坐标决定，其余边缘点的坐标可以进行同比例变换得到。

每一行的中间点的坐标可以根据特殊点的左边线性插值计算得到。每一行如下图所示。

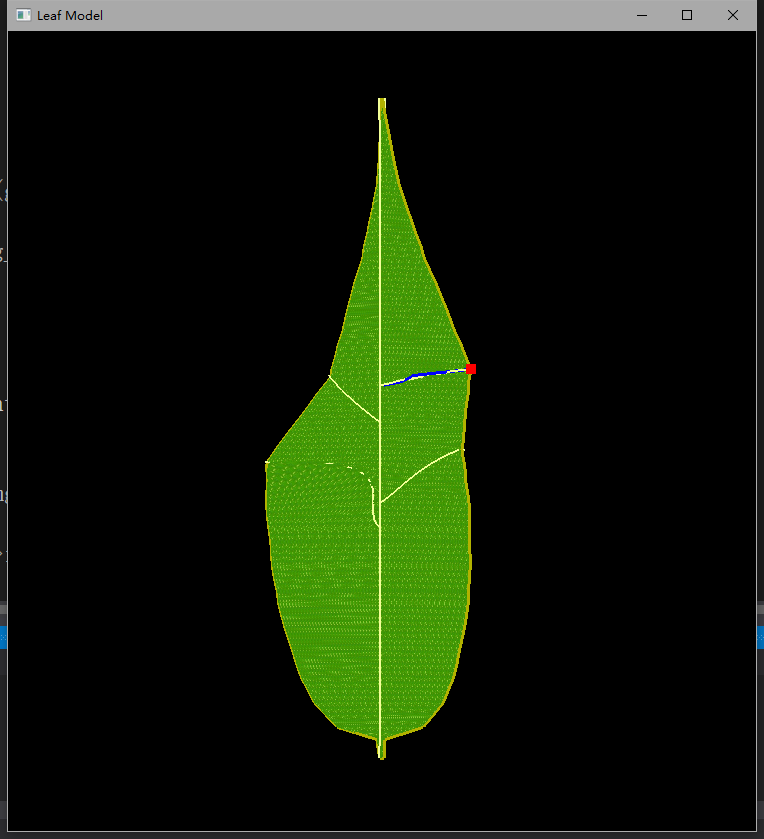
主脉点

侧脉点

边缘点

边缘点

整体变形效果可以如下图所示



## 程序设计

Point类实现了顶点，主要成员是坐标（x, y, z）,以及该点的法向（nx, ny, nz）,原本是想通过法向实现光照的效果，但是时间没来得及实现。

Triangle类实现了三角形，通过v1, v2, v3记录了三角形对应三个点在点表中的索引。

BezierLine类主要实现了Bezier曲线，记录了它的控制点，以及用来拟合该曲线的50个绘制点。该类还实现了根据位置系数计算出Bezier曲线上某个点的坐标，用来计算主脉点和侧脉点的坐标。

Ratio\_Point主要是用来记录特殊点（主脉点，侧脉点，边缘点）的信息。包括该点的类型，涉及的Bezier曲线以及在上面所处的比例位置，或者是边缘点与叶边缘上其他控制点的坐标比例关系。用来在叶脉变形后计算特殊点的位置。

Leaf则定义了叶片。成员v\_list是记录了三角网格顶点的点表，v\_index则记录了每个三角形对应的顶点在表中的索引。由于叶面的实现是每一层每一层根据叶片图片的宽度建立网格，所以用成员layer\_index记录每一层的开始顶点的索引，成员layer\_mid\_index记录每一层中点的索引。成员mainvein是BezierLine类型，记录了主脉信息。成员vein是BezierLine类型的vector，记录了侧脉的信息。成员texGround绑定了纹理。成员point\_with\_ratio记录了所有的特殊点，作为叶面变形的基础。成员leftVeinEgde和rightVeinEgde则记录了左叶边缘和右叶边缘上的点，用来计算叶边缘的变形。成员veinStartIndex则记录了位于主脉上的侧脉的控制点，用了控制主脉侧脉协同变形。

其他的重要函数实现包括：

GLuint load\_texture(const char\* file\_name)，读取了名为“texture.bmp”文件，并建立了纹理用来映射。

int sharpening\_img(const char\* filename, Mat &grad)，通过Sobel()使图片锐化，方便提取叶轮廓。

vector<float> get\_vector\_from\_img(const char \*filename, int step, int &h)，通过逐几行扫描轮廓宽度，得到的信息用来重建叶面网格。

int gen\_vertices\_and\_index(vector<Point> &v\_list, vector<Triangle> &v\_index, vector<int> &layer\_index, const vector<vector<float>> &vvv, float distance)，将宽度信息转变为Leaf的v\_list与v\_index。

其他位于control\_functions.cpp中的函数主要是用过建立glut窗口，以及绑定按键，，鼠标点击事件，绘制事件等一系列功能，在此不一一介绍。

## 项目成果

初步实现了根据一张图片建立叶面模型，并且交互式建立叶面上的叶脉。

提供了交互式变形，主要实现是鼠标选择控制点，键盘使控制点发生位移，进而使叶脉发生变形，进而使叶面发生变形的功能。

同时预设了一种根据观察经验得到的变形模式（键盘P键）。整个模型可以旋转，从不同角度观察。

## 不足与反思

* 轮廓扫描的算法基本只能识别出外凸型轴对称叶片，导致叶面的实现都是轴对称图形。这是为了简化问题出发的做的考虑，存在改进空间。
* 叶面变形算法的中间点坐标都是根据特殊点坐标线性插值，导致特殊点间的叶面连线都是直线，一定程度上模拟真正的叶面变形，但还是存在失真。
* 由于时间所限，未能完全实现光照功能，导致模型看上去不那么真实。这一点可以通过学习后加入合理的光照功能。
* 原最终目标是能够导出.obj格式的模型文件。如今在所需的数据应当足够充分，但是由于时间所限未能完全实现。