Introduction

实验目的：掌握 OpenGl 的使用方法，性质以及功能，理解 OpenGl Pipeline 流程。学会如何读取数据文件并调用 OpenGl 的 API 渲染出符合我们要求的 3D 图形。

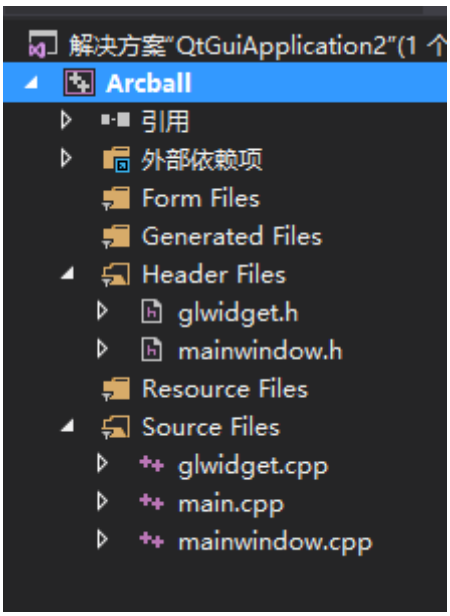
实验目标:

- 利用 qt 读取数据文件，保存到相应的变量中
- 利用 OpenGL 绘制所读取数据的 3D 模型
- 渲染颜色，改变图形大小，加入灯光和点光源
- 实现交互操作：旋转球状模型，鼠标（按中健或者其他健）在二维屏幕上拖动，之后三维模型就会以屏幕中心点为中心进行相应的旋转，鼠标拖动得越长，三维模型旋转的角度就越大。
- 实现绘制法向量的操作

实验环境: Visual Studio 2015 + Qt5.8，操作系统 Win10.

Structure of the code

本程序右三个主文件构成，分别为 main.cpp、mainwindow.cpp、Glwidget.cpp 如下图所示：



main.cpp 程序的入口，创建主窗口

mainwindow.cpp 窗体类，包含菜单栏和 Glwidget，其中函数如下：

public:

explicit MainWindow(QWidget *parent = 0);构造函数，进行窗口组件的初始化

~MainWindow();析构函数

void openFile();用来发送 func 信号，打开文件

public slots:

void openclicked();//定义槽函数 当鼠标点击 loadfile 按钮时执行此函数，并获取图片路径

signals:

void func(QString);触发 Glwidget 中 函数的信号

其中变量如下：

private:

QString fileName; 用来保存读取文件的名称

Glwidget.cpp 负责 3D 图形的绘制，其中包含多个函数，如下所示：

GLWidget() 构造函数

~GLWidget() 析构函数

initializeGL() 设置OpenGL渲染环境，定义显示列表等。

paintGL() 绘制图形和法向量，渲染OpenGL场景。

resizeGL() 设置OpenGL的视口、投影等。

mousePressEvent() 点击鼠标时，更新鼠标位置

mouseMoveEvent() 移动鼠标时，更新鼠标位置，并调用arcBall()

minimumSizeHint() 返回窗体的最小大小

setupVertexAttribs() 设置顶点的属性数据

arcBall(QPoint last_pnt, QPoint curr_pnt); 根据移动的起点与终点，旋转3D图形

double dot(QVector<double> v1, QVector<double> v2); 求两个向量的点乘

double angle(QVector<double> v1, QVector<double> v2); 求两个向量的角度

QVector<double> crossProduct(QVector<double> v1, QVector<double> v2);求两个向量的叉乘

void process_line(QByteArray line); 读取和处理文件中一行的数据

public slots:

void loadFile(QString); 加载数据文件，从中得到顶点数组和方向量数组

void controlNormal(); 控制法向量线段的显示，点击按钮触发

其中变量如下：

QOpenGLVertexArrayObject m_vao; 顶点数组对象

QOpenGLBuffer m_vertexBuf; 缓冲区，用来存储点和向量的信息

QOpenGLBuffer m_indexBuf; 缓冲区，用来存储面的信息

QOpenGLShaderProgram *m_program; opengl shader程序

QMatrix4x4 m_proj; 图像大小矩阵

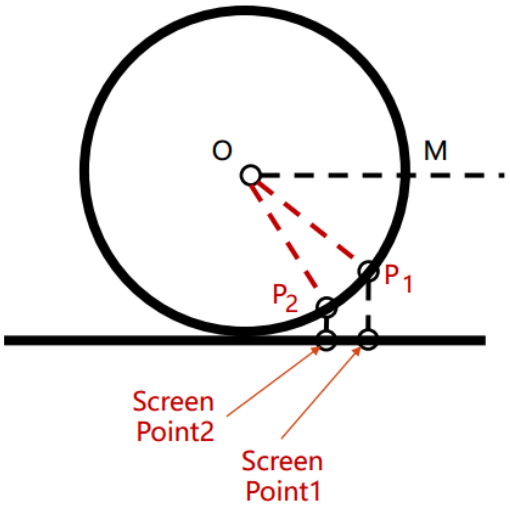
QMatrix4x4 m_camera; 相机矩阵

QVector3D rotateAxis;旋转轴
float rotateAngle;旋转角度

vector<QVector3D> vertex; 存储法向量和点
vector<int> face;存储面信息
QVector3D *vArr_c; 指向法向量和点的指针
int *fSets_c; 指向面向量的指针
bool isShowNormal; 控制是否绘制法向量直线

Algorithm

当鼠标在球面的范围内移动时，我们可以由鼠标在二维屏幕上的二维点坐标P（x,y）通过数学关系求得其在球面上的投影点P'，鼠标从P1点移动到P2点，对应的在球面上就是从P1'移动到P2'。P1'和P2'与球心之间可以形成两个向量，鼠标移动转化成了向量从V1（OP1'向量）转到V2（OP2'向量），V1和V2的向量叉乘得到向量N即是三维物体的旋转轴，V1到V2的转角量就是三维物体的旋转角度。如图所示：



ArcBall 的有关算法

Screen Point(x,y)
Sphere Pont(Px, Py)
Px = 2 * x / width -1
Py = -(2 * y / height - 1)
z = sqrt(1 – x^2 – y^2) (x^2 + y^2 <= 1)
z = 0.0 (x^2 + y^2 > 1)
v1 = P1 – O;
v2 = p2 – O;
Rotate Angle: v1 dotProduct v2
Rotate Axis: v1 crossProduct v2 and then normalize

Show Normal 的有关算法

FaceNormal:

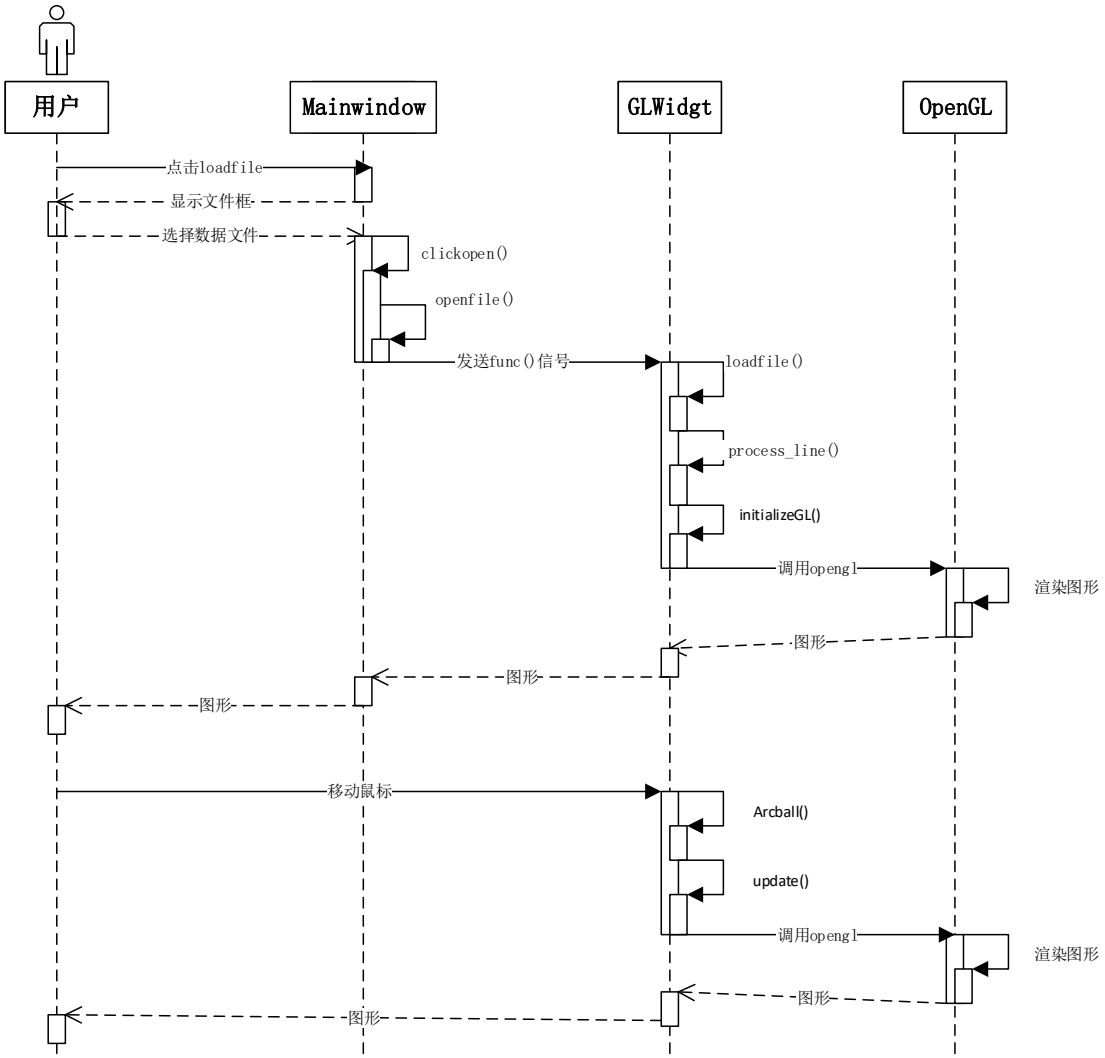
$$N_3 = \overrightarrow{AB} \times \overrightarrow{AC}$$

VertexNormal:

$$N_{vertex} = \frac{1}{n} \sum_{k=1}^{n} N_k$$

Figure

Sequence Diagram

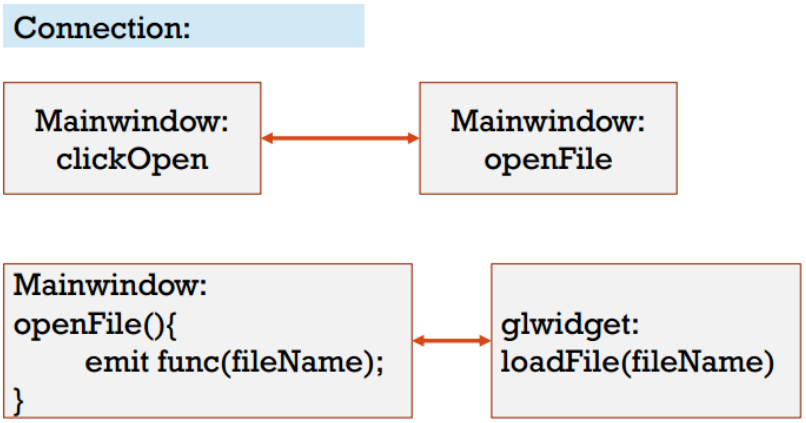


Signal-Slot connection

读取文件的信号与槽：

connect(open, SIGNAL(triggered()), this, SLOT(openclicked()));
connect(this, SIGNAL(func(QString)), glWidget, SLOT(loadFile(QString)));

其调用顺序为：



加载法向量：

connect(showNormal, &QAction::triggered, glWidget, &GLWidget::controlNormal);

Discussion

本实验轨迹球（ArcBall）可以让你只用鼠标来控制模型（旋转），实验过程中主要是对程序流程的熟悉过程，然后针对不同的功能分别进行实现。从最初的读取文件到展示球模型其中遇到了很多问题，但在老师和同学的帮助下逐步的完成了本实验的内容。但实验只实现了基本的旋转和显示法向量的功能，很多还不完善，例如缩放、变色等等。这个实验勾起了我对图形学的热爱，我想在平时我会逐步完善这些功能并想要进一步的学习图形学的知识。