做一个vector，对这个vector 进行数据扩充，这些元数据包含了：点，与这个点相连的前向节点的位置，这代表了一个线段（后续的平面还没有想好怎么实现）。

投影平面 z = -d 注意符号，d为正数

尽量不使用qt内建函数，能自己写就自己写，保证可移植。

Spoint类：二元点。Screen Point

Striangle类：最终屏幕缓冲器Gbuffer直接存这个东西，由三个Spoint构成，有颜色信息。

Gquat类：应该是一个四元组，支持相加，相减，点乘，叉乘。

Gpoint类：继承Gquat类

代表一个空间中的点，应该有4个数据，作为四元组，同时还应该有其他的功能

Gvect类：方向向量，应该可以对方向进行设定，也应该继承Gquat类

Gtriangle类：这个应该是显示的基本单位，在绘制的时候，以这个为单位。代表三角形，在实际中渲染的时候，应该对三角形有一个表示，这个应该有：三个端点的点，法向（要能分内外），颜色（基本颜色），mainwindow存储的数据应该是以它为单位？这个应该有自己的绘制函数，所谓绘制，就是这个三角形先进行投影映射，传入Gbuffer，Gbuffer应该是Mainwindow类的一个数据成员，MainWindow要做的，就是把所有的Gbuffer中的三角形显示出来。

Gvol类：多面体，给出几个点，自动连接这些点，将这些点连成一个最小凸多面体，可以先实现4个点的四面体。

曲面、球等等物体因为要实现网格化，所以先不用考虑，先把多面体搞定再说。

先确定图形绘制，再进一步进行光照绘制。

第一天：

基本实现了三维物体的绘制，可以绘制一个单色四面体，但四面体的几个平面如何显示还是问题，即渲染顺序的问题，此外，还有一个问题是渲染的四面体不知道正确与否，需要检验。

同时，明天希望可以实现光照模型的实现。

逆映射：

先要做到的是：对于一个图形，我如何遍历上面所有点？（在显示的时候，也要遍历，所以，最好的方案是，直接遍历2D模型，先遍历三角形，三个点。最好是把整个屏幕渲染了，每个2D坐标下存1个3D参数，即这点的法向，此外，还应该有一个Zbuffer，存的是这点的Z轴的分量，便于还原其3D位置，pixel类，这种储存方式的缺点是：所有屏幕的点，这样存的数据点太多）我的解决方案是：储存用线性数据结构，而判断显示关系用另一个便于快速查找的储存空间？一个用vector，一个用map?

考虑屏幕显示的问题，将即将显示在屏幕上的2D坐标映射回3D坐标（法向），计算这些点的光照强度。