# 实验报告

成员：李明哲 2016011053

袁明瑞 2016011045

（实验说明：本程序是在linux系统下编写编译的，没有Windows可执行文件，所以只能给出源码，但是经测试好像源码在windows系统下opencv库有一些问题，不能正常编译，所以只能在linux下运行，编译器使用clion即可。要想在本地运行成功，需要修改画图程序main文件第98行的fopen的参数为本地的一个位置，并把图形程序mainwindow.cpp中所有的fopen（"/home/li/cghm/binary.dat","wb"）中的"/home/li/cghm/binary.dat"改为上面修改的路径，同时修改该文件第54行的program->start函数的参数为图形程序编译后生成的可执行文件的路径。最后图形程序和画图程序要分别编译。以后如有时间，我会写一个自动配置这些东西的脚本，目前只能用这样的方法将就运行一下。生成的图片在画图程序目录下，名字为“test.jpg”）

## 用到的计算机图形学原理

1. 投影变换原理：在世界坐标系内构建物体，根据摄像机位置应用坐标变换，得到观察者坐标系；再利用投影矩阵得到观察到的图象。

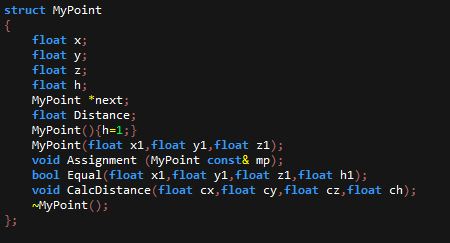
2. 物体遮挡：根据z-buffer算法计算不同物体的次序，形成观察者看到的图象。

<应用算法> 坐标变换、投影变换、z-buffer.

## 主要代码实现原理

1. **类架构：**

**MyPoint:**

****

**<变量>** h-齐次坐标， x,y,z-三维空间位置;

Distance-记录该点到观察点的距离;

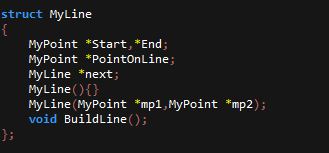
next-所有点组成的链表

**<函数>** Assignment-复制另一个点

Equal-判断两个点是否相等

CalcDistance-计算距离

**MyLine：**

****

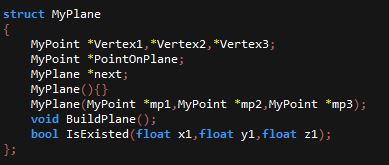
**<变量>** Start End – 起始、终止点；

PointOnLine – 该直线上所有点

next – 所有直线组成的链表

**<函数>** BulidLine – 构建一条直线

**MyPlane：所有的平面都设定为三角形平面，简化类别**

****

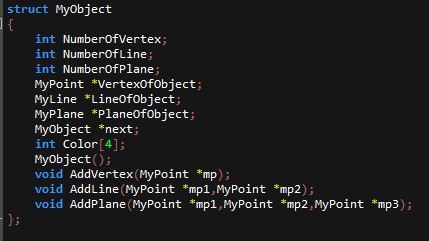
**<变量>** Vertex-i 表示三个顶点；

PointOnPlane 平面上所有点；

next 一个物体中的所有平面

**<函数>** IsExisted 判断某点是否出现在该平面上；

**MyObject:**

****

**<变量>** 记录点、线、面的数量；

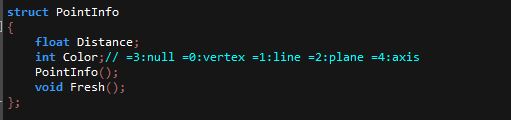
以链表形式存储点、线、面；

存储物体颜色；

**<函数>**

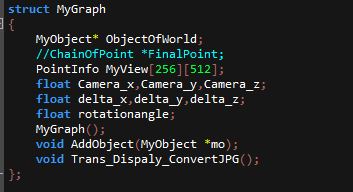
添加 点、线、面。

**PointInfo:**

****

记录与投影平面相关的一些信息，包括某个点到投影平面的距离、该点的颜色。

**MyGraph：**

****

**<变量>** ObjectOfWorld – 世界坐标系中的物体

MyView – 投影平面，设定大小是256\*512；

Camera – 记录摄像机的坐标；

Delta – 表示摄像机在平动方向的移动距离；

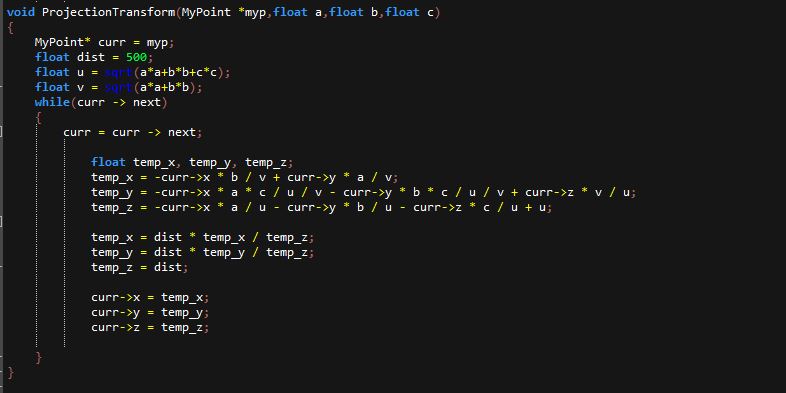
Rotationangle – 表示摄像机旋转的角度

**<函数>** AddObject – 添加物体；

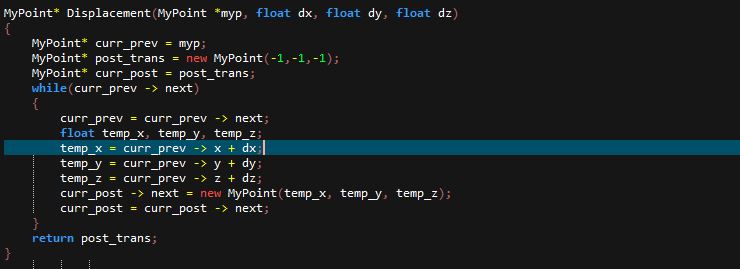
Trans – 将其转化为图片输出。

1. **投影变换：**

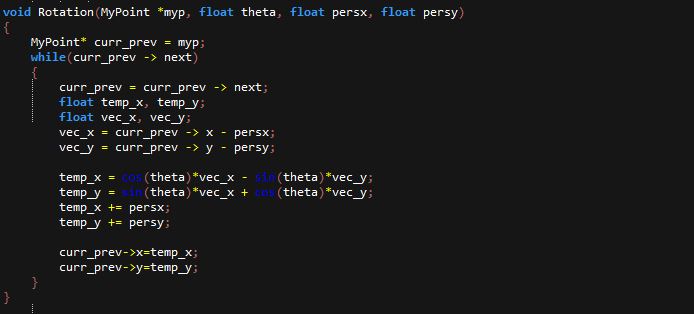
**Projection：世界坐标系转换至观察者坐标系**

****

**Displacement：摄像机平移**

****

**Rotation：摄像机旋转**

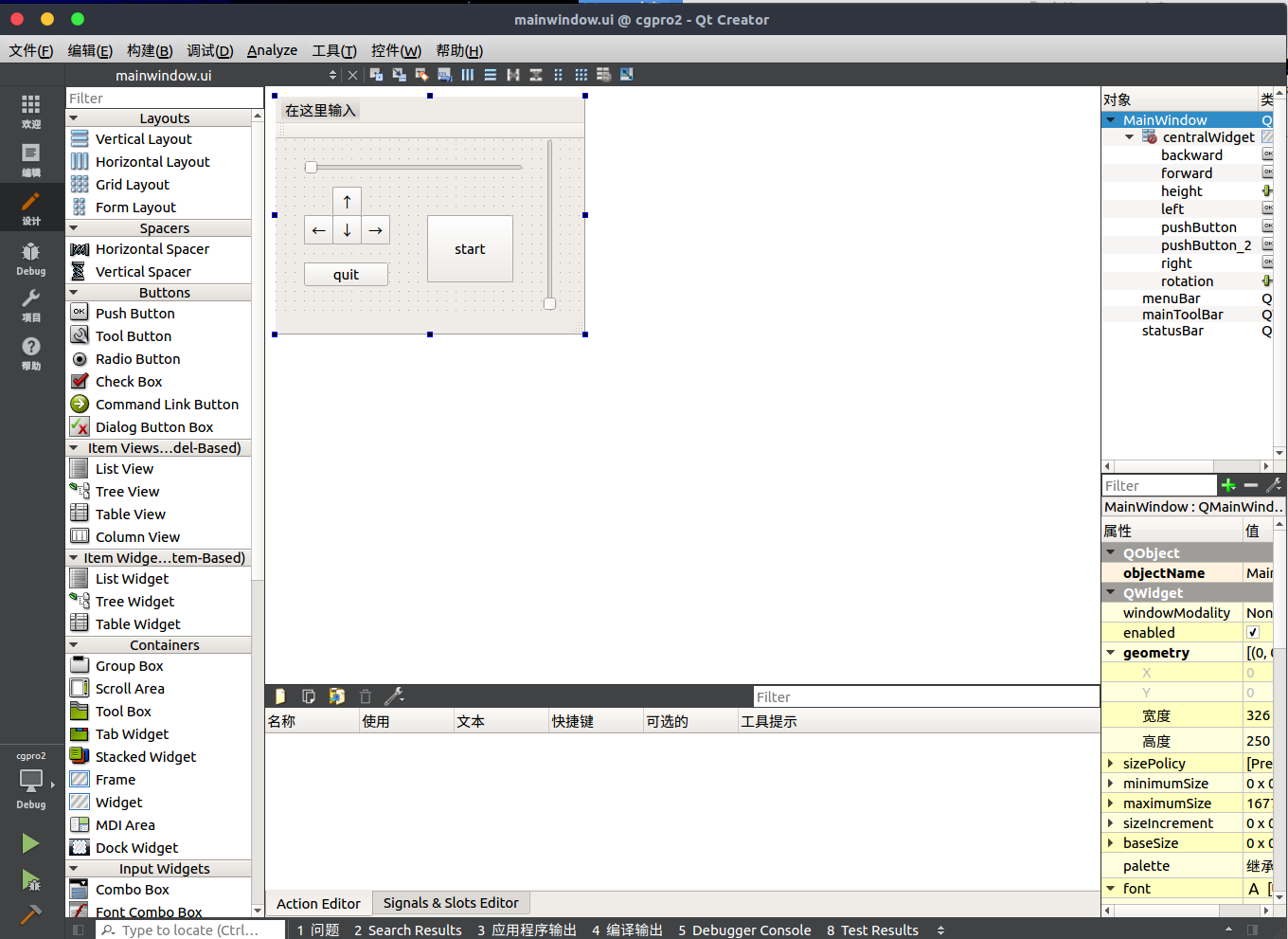


1. **进程机制：**

为了使得每次有新的输入都会重新计算并更新图像，在主函数里采取了检查机制，每20ms检查一下输入文件里的内容是否改变，改变就执行绘图主函数，否则就等待。

## 界面的编写

界面用qt5编写，开发界面如下

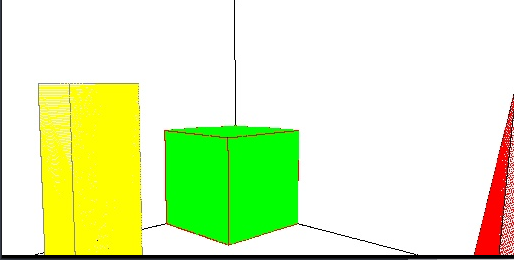


主要思路是start按钮用来执行第一遍画图程序，上下左右按钮调整观察点位置

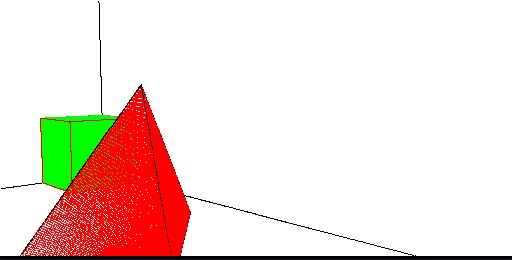
横向滑动条用来改变视角方向，垂直滑动条用来调整视角高低

每次有动作就会更新视角角度，位置的xyz坐标等参数，写入二进制文本中，供画图程序重新画图，按钮滑动条的控制使用了qt的信号槽技术

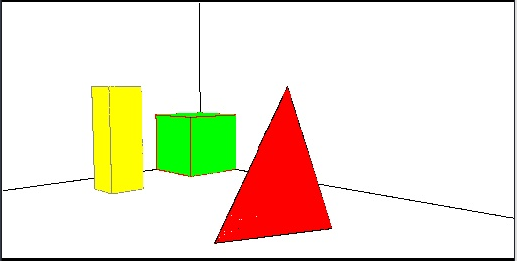
## 展示效果



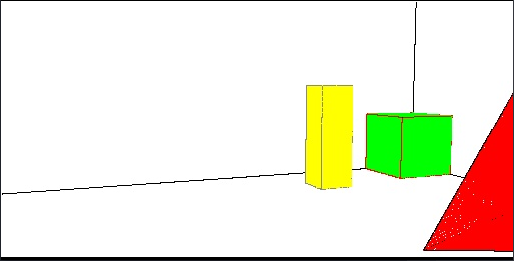
（初始图像）



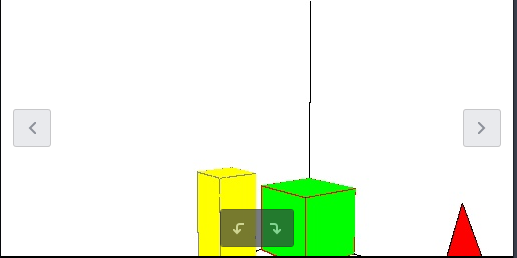
（按↓平移）



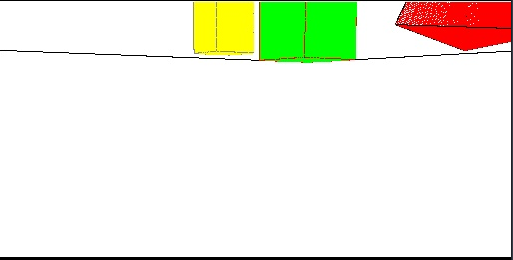
（再按←平移）



（视角向左移动）



（视角向上移动）



（视角向下移动）

## 可改进的地方

一开始写这个程序的时候没有考虑到内存的问题，“MyPoint”这个类设计的不好，投影点和原来的点是不同的对象，每次都要新建一个链表存这些对象，一幅256\*512，三个几何体的图像占用内存25Mb左右，执行几十次内存就爆了。尝试过析构，但是因为中间不知道哪里写的有问题，强行析构会报运行错误。

之后想到了更好的数据结构，让MyPoint类包含投影点坐标数据成员，这样不用新建链表，理论上内存不会变多，无论执行多少次都会维持在50Mb以下。但是由于时间仓促，而且改数据结构整个程序也要全部配合修改，没有来得及改动。打算暑假修改一下，这样可以让程序更流畅，而且显示的图形可以更多。

改用新的数据结构后，理论上也可以应用更快的填充算法，图像可以做的分辨率更高而不占用更多资源。

在倒数第二节课上的光线效果也提供了一个新思路，可以添加光源并计算物体表面的亮度，这个改动对于这个程序来说并不困难，也可以考虑加进来。

还有就是图形界面与操作的优化，实际上用界面上提供的几个按钮与滑动条控制视角效果很奇怪，我在考虑换成直接在图像上拖拽的操作形式。

以及应用上的改进，这个程序实际上有很强的应用性，可以模拟3D效果，目前想到的应用有3D魔方小游戏，4维几何体在三维空间投影的演示。