郑州大学毕业设计（论文）

题 目 圆柱测量数据的图形化显示系统开发

院 系 机械工程

专 业 机械工程

年 级 2014级

学生姓名 时猛

指导教师 杨炯

2018年 05月 20日

**摘要**

本文对圆柱数据的测量的技术发展进行了研究和阐述，针对圆柱数据的测量系统，本文设计了一种高精度的测量数据处理方法，并开发了一种形象可视化的上位机软件。利用面向对象技术，多线程技术以及OpenGL图形开发技术等，这些关键技术为整个项目的软件架构和软件设计提供了很大的便利，提高了软件的开发效率以及软件的高效性。

测量数据的显示系统包括数据通信，数据拟合，图形展示等。数据通信是基于半双工通信与PLC连接，获取PLC的测量数据；数据拟合采用最小二乘法圆度拟合。图形展示是基于OpenGL将测量的数据形象的展示为图形，处理后的结果精度高，数据重复性好。上位机软件是基于C++编程语言开发而成的，集成了二维图形展示，状态栏，菜单栏，数据导出以及三维图形展示等功能，打造了一个人性化的软件界面。

关键词 圆度拟合， 图形化显示， 圆拟合（关键词3-5个）

Abstract

In this paper, the technical development of cylindrical data measurement is studied and elaborated. For the measurement system of cylindrical data, this paper designs a high-precision measurement data processing method, and develops a visualized upper computer software. Using object-oriented technology, multi-threading technology, and OpenGL graphics development technology, these key technologies provide great convenience for the software architecture and software design of the entire project, improving the efficiency of software development and the efficiency of software.

   The display system of measurement data includes data communication, data fitting, graphic display and so on. Data communication is based on half-duplex communication and PLC connection to obtain the measurement data of PLC; data fitting is based on least-squares roundness fitting. The graphic display is based on OpenGL's display of the measured data image as a graph, with high precision after processing and good data reproducibility. The host computer software is developed based on the C++ programming language. It integrates functions such as 2D graphics display, status bar, menu bar, data export, and 3D graphics display, creating a personal software interface.

key Words Roundness fitting，Graphical display，Circle fitting

目 录

毕业设计任务：……………………………………………………………..I

摘要…………………………………………………………………………..V

Abstrct………………………………………………………………………..V

1 绪论………………………………………………………………………1

1.1 选题的目的意义………………………………………………………1

1.2 国内外研究综述………………………………………………………1

1.3 本题的主要工作…………………………………………………………2

2 算法分析 ……………………………………………………….3

2.1 圆拟合算法分析………………………………………………………3

2.2 圆柱拟合算法分析…………………………………………………7

2.2.1 圆柱面拟合方法原理

2.3 本章小节…………………………………………………………………9

3 网络通信……………………………………………………………

4 系统架构……………………………………………………………

5 结论……………………………………………………………………….25

谢辞…………………………………………………………………………...26

参考文献……………………………………………………………………...26

1 绪论

1.1 选题的目的意义

在计算机视觉识别中 , 圆、点与直线一直是最基本的图像的基本特征。在我们的生活中包含的大量物体为圆,圆形既是世界上最简单最基本的形状，而且是人类社会产品中普遍存在的形状, 圆的鲁棒精确提取在工业领域和检测领域中有着非常特别重要的意义。

1.2 国内外研究综述

在测量检测的过程中, 通常我们会遇到三维圆柱图形的测量问题, 例如在一些建筑物的顶层、大型机械设备仪器和平时乘坐的地铁遂道的工程测量中, 通常我们是对三维空间物体进行检测，求出它的圆心以及半径,确定我们是否能够施工。并且在机械领域，各个零件的圆孔定位对零部件的安装和装配有这非常重要的作用， 进行零部件圆中心定位的各种方法为:从图像边缘检测所得数据进行圆的拟合 ,或者将边缘数据转换为三维空间数据进行圆的拟合。因此,圆的拟合是圆中心定位的关键问题之一在机器视觉领域，圆是最为基本的图像特征，广泛应用于定位、测量、识别等方面，其拟合方法具有重要的研究意义。经常会遇到需要利用有限样本点进行圆拟合的问题，如在一些大型仪器和地铁隧道的工程测量中，经常需要对空间圆形物体进行测量，得出其圆心及半径，确定其是否满足施工要求。利用有限样本点进行圆拟合的情况，常用的方法有：

1. 平均值法

针对样本点均匀分布在圆上的情况，从理论上来说，其圆心坐标为各点坐标的平均值，圆心到任一点的距离即为圆的半径。但是在实际问题中，样本点不可能正好均匀的分布在圆上，因此误差较大。

1. 加权平均法

平均值法求得的圆心位置会偏离数据点密集的一边，求出的半径也会偏小。加权平均法是将样本点距离其相邻两点的弧长与圆的整个周长的2倍的比值作为权数。样本点距离其相邻两点的弧长可用相邻两点的距离替代。

（三）最小二乘法

通过最小化误差的平方和找到一组数据的最佳函数匹配。但是最小二乘法拟合的平方项对干扰数据非常敏感。

1. 霍夫变换法

将样本点为圆心，在参数平面画圆，许多点对应的圆会交于一点，该交点即可能为圆心。缺点计算量大，运算时占用内存大。

1. 基于共形代数的圆拟合方法

将欧式空间嵌入到共形空间，将圆表示为向量形式，共形空间中点与圆的内积表示点与圆的位置关系，对具有噪声的样本点进行拟合实验。

1. 基于聚类方法的圆拟合算法

该算法将边缘样本点映射到圆心附近，使用聚类算法进行分类滤噪处理并计算圆心，最后以此为基础计算半径。机器视觉进行圆拟合，首先采用工业相机对圆形特征进行拍照，图像在拍摄、传输和存储过程中都会不可避免地产生噪声，这将带来较大的拟合误差。因此去除圆上其他轮廓干扰或对其局部失真进行纠正是一个十分重要的步骤。

1.3 本题的主要工作

基于该项目的目标是完成基于OPEN GL开发圆柱测量数据的显示功能模块；实现利用给定的截面计算圆柱度等数据功能。因此考虑时间因素和学习成本，确定选择利用最小二乘法拟合圆，经过多次迭代生成最终结果。通过C++编程，实现该项目。

2算法分析

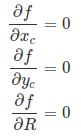
有一系列的数据点深度截图_选择区域_20180511155156，我们知道这些数据点近似的落在一个圆上，根据这些数据点计算出这个圆的参数这是一个非常重要的问题。圆拟合的方法有很多，本文选择的是基于最小二乘法。

2.1圆拟合算法分析

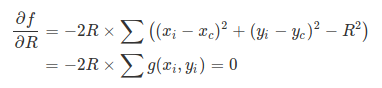
我们知道圆方程为深度截图_选择区域_20180511124008通常的最小二乘法拟合要求距离的平方和最小。也就是深度截图_选择区域_20180511124145最小，但是这个算起来非常的麻烦，也得不到解析解。因此我们退一步。深度截图_选择区域_20180511160736

上面这个式子简单的多。我们定义一个辅助函数：深度截图_选择区域_20180511160836

那么上述公式可以表示为： 深度截图_选择区域_20180511161222按照最小二乘法的通常的步骤，可得f取极值时对应下面的条件。



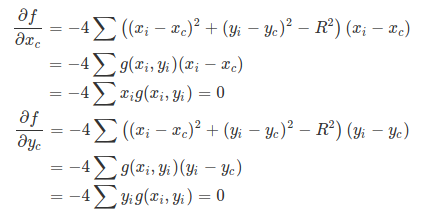
先来化简深度截图_选择区域_20180511161418



我们知道半径R是不可能为0的。所以必然有：

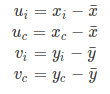
深度截图_选择区域_20180511162353

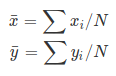
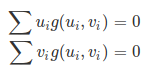
同样

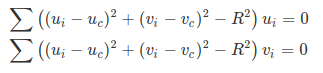


也就是下面两个等式： 深度截图_选择区域_20180511162510

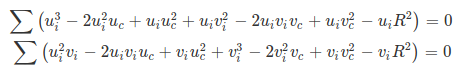
这里设：



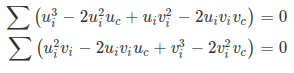
其中： 可以简单推导一下，就会发现：

这个变量替换修改了坐标远点的位置。而我们的等式与坐标原点的具体位置无关。所以必然成立。这两个式子展开可以写成这样：

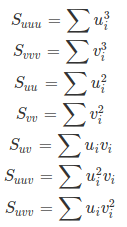
进一步展开：



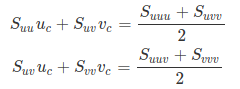
我们知道深度截图_选择区域_20180511163245所以上面两个式子是可以化简的。



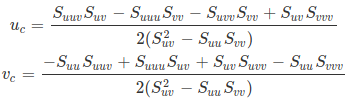
为了简化式子，我们定义几个参数：



那么上面的式子可以写为：

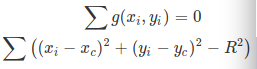


至此，可以解出Uc和Vc了。



那么：

深度截图_选择区域_20180511163627还剩下一个R没有求出来，也很简单。



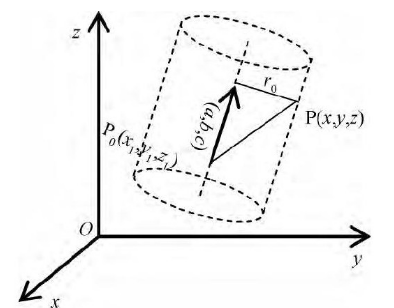
所以：深度截图_选择区域_20180511163750

以上就是最小二乘法的数学原理推导。然而这种方法对误差符合正态分布的数据点很有效。但是当数据中碰到一些干扰点。这些干扰点多数时候是偏向某一个方向的。因此，本文采用的是根据数据点到圆的距离绝对值的和来确定圆的参数，也就是下面的数学公式：

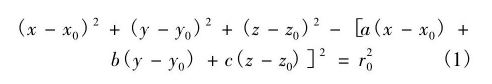
深度截图_选择区域_20180511164157 使得f取得最小值的Xc、Yc和R就是最佳拟合参数。

2.2 圆柱拟合算法分析

已知圆柱面上的点到其轴线上的距离恒等于半径r0，如图所示



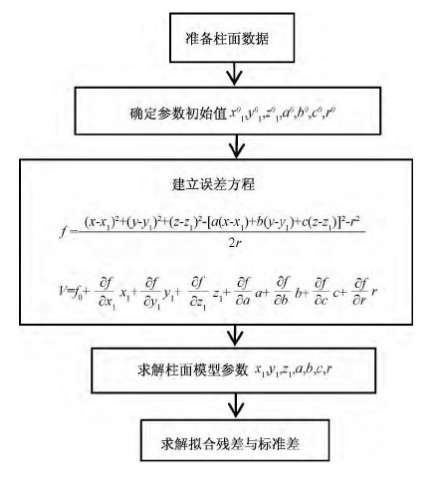
其中点P为圆柱上的任一点，P0为圆柱轴线上任一点，即：



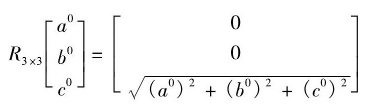
根据以上公式就可以唯一确认一个圆柱。

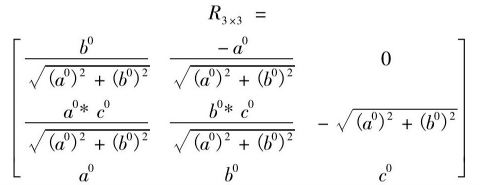
2.2.1 圆柱面拟合方法原理

圆柱面的拟合方法主要包括两个步骤：第一确定初始值；第二建立误差方程求解析值。本文的流程图为：



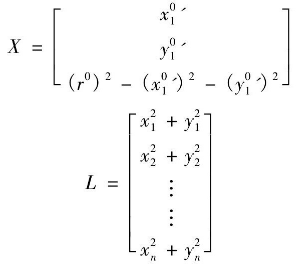
第一：将圆柱面上一点的相邻若干点拟合成一个平面，求得这个平面的法向量即为点的单位法向量。假设旋转矩阵为R3x3,要使：





写成矩阵形式为：

深度截图_选择区域_20180513150817

其中：

深度截图_选择区域_20180513150932

2.3 本章小结

本章主要说明该圆柱测量数据图形化显示系统中圆度拟合的算法分析，圆度拟合算法属于计算机图形学领域的算法分析。无论是在测量，还是在检验的过程中，我们都需要用到圆的拟合，圆拟合算法在工业领域应用的方方面面，我们往往是将一系列的点集生成一个一个的线段，之后依据线段之间的联系组合成为一个圆。针对误差过大的点，我们也应当可以清除数据，这些数据都属于冗余的，因此往往最后都需要做校验以及正确性检测。

3 网络通信

3.1 三菱PLC的分类及FX系列PLC的发展历史

三菱PLC包括一体式的和模块式的，有简单的控制系统，到高端的自动化解决平台。一体式的PLC数据集成式PLC, 包括CPU板、I/O板，显示面板、内存块、电源等，这些组件组合成一个整体。三菱FX系列的PLC属于一体式的，这种PLC多用于中小型系统。其主要特点就是可以实现10点到384点的控制规模，它体积比较小，并且容易安装，性价比高，成本低廉。同时可以实现多种网络、模拟量转换等多种特殊功能。而模块式的PLC包括CPU模块、I/O模块、内存、电源模块、以及各种功能的模块，这些模块可以按照一定规则组合配置。I/o点数可以达到4096点，最快的处理时间可以达到9ns,并且可以使用模拟量、网络等多种特殊功能模块。

70年代日本三菱公司开始研发PLC，之后第一台F系列投入实际应用，至今已经是高性能、高可靠型的最佳代表。FX系列包括FX1S、FX1N、FX2N、FX3U、FX3G等。其中FX1S基本单元有继电器或晶体管输出，不能链接扩展模块和特殊功能模块；FX1N可以连接FX2N系列的特殊功能模块，备有多种功能扩展板供使用；FX2N适用于从普通顺控开始的广泛用途。FX3U属于第三代微型可编程控制器，具有速度、容量、性能、功能的新机器、业内最高水平的高速处理，大大提升了内置功能；FX3G与FX1N系统兼容、高性能、高灵活性。提升用户友好。增强了程序安全性。

在Windows系统下使用的集成编程工具是GX Developer, 在设计、调试方面大大提高了编码的效率，允许以顺序功能图和结构化文本等语言进行编程。网络参数的设定和网络状态监视变得简单容易，并且同时可以在线监控程序、改变程序和数据以及强制I/O的ON/OFF等。可以实现离线编程、模拟测试和仿真等。在整个过程中，PLC通过ON/OFF位软元件或给字软元件赋值来实现真是动作情况。

3.2 PLC通信系统构成

FX系列PLC包括内置的CPU、存储器、输入输出、电源的产品。其中输入输出内置了电源回路和输入输出，用于扩展输入输出的产品，可以在连接其后面的设备供电，输入输出扩展模块内置了输入、输出，用于扩展输入输出。可以连接在基本单元上使用。特殊功能模块是一种可以是实现多种I/O以外其他功能的模块。包括模拟量控制、定位控制、网络系统、高速计数等功能。而像特殊适配器类似与特殊功能模块，也包含了模拟量和定位，通讯等多种功能。其他外部设备也是PLC系统不分割的一部分，它包括通过专用电缆，进行程序编制、对系统作一些设定、监控的手持编程器或编程软件、显示模块的监控设备和存储器以及用于安装特殊适配器用的连接器转换或实现通讯功能时使用。

整个系统构成包含三个方面，一是关于输入输出点数，二是关于特殊扩展设备，第三是关于消耗电流的计算，确认其是否可以扩展的输入输出点数。将剩余的电源容量作为外部负载的电源使用。剩余电源容量是否允许连接，特殊适配器和特殊功能单元。输入输出点数的合计就是计算基本单元、输入输出扩展单元/模块的输入输出点数，以及特殊功能单元/模块的输入输出占用点数，不同型号的输入输出占用点数，可以通过特殊功能单元/模块的台数 X 每台的点数 = 输入输出占用的点数。

如果CPU连接输入输出扩展单元，当上电时，会自动输入输出标号进行标号分配，因此，无需通过参数制定输入输出编号。输入输出编号以8进制进行分配，扩展的输入输出扩展单元/模块，接着前面的输入编号和输出编号，末尾从0开始分配。上电时，CPU会从离其最近的特殊功能模块开始，依次对特俗功能单元分配单元号，而在输入输出扩展单元/模块没有这个单元号。基本单元的输入为内部供电的漏型输入和源型输入的通用产品。通过选择可以将基本单元的所有输入设置为漏型输入，或者源型输入，但是不能混合使用，各基本单元和输入输出扩展单元，可以分别选择漏型输入、或是源型输入。

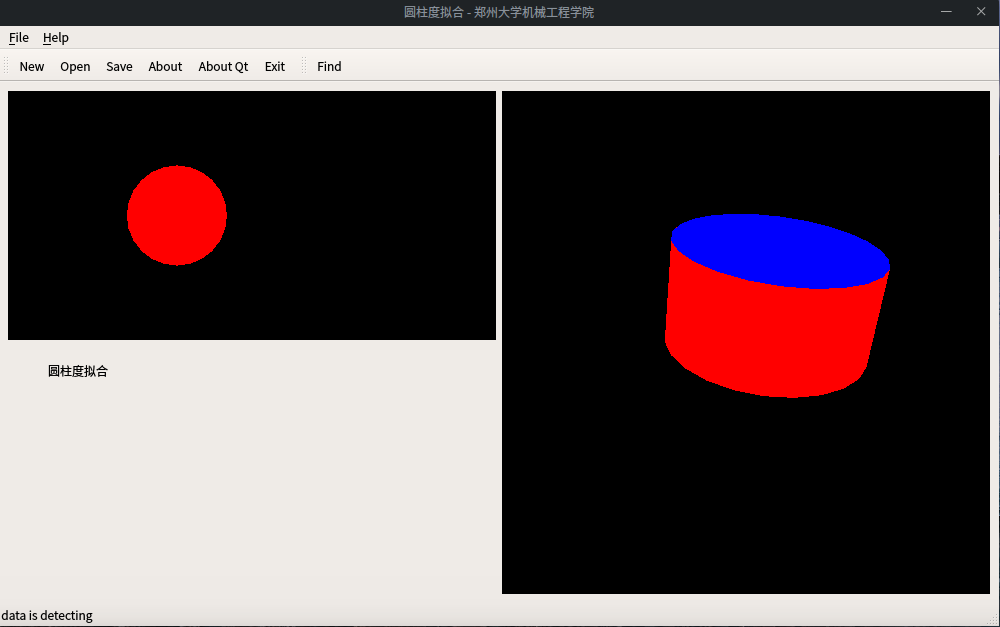
PLC执行程序是通过循环执行的顺序从而实现顺序控制。分为输入处理阶段，程序处理阶段、输出处理阶段。输入处理是在可编程控制器执行程序前，将可编程控制器中的所有状态读入。程序执行过程中即使输入发生了变化，输入映像区也不会发生变化，在执行下一个循环的输入处理时读取该变化。编程控制器根据内存中的内容，从输入的映像区和其他的映像区读出各软元件的状态，从最初依次开始运算，并将每次得出的结果写入到映像区。因此，各软元件的映像区随着程序的执行逐步改变其内容。所有指令执行结束后，输出映像区中的状态会传送至输出锁存内存。所谓的工程就是至程序、软元件注释、参数、软元件内存的一种集合体。

FX系列可编程控制器中，使用SFC图（顺序功能图)作为实现顺控，以一种便于理解的方式表现基于机械动作的各工序的作用和整个控制流程。在SFC程序中，状态S被作为一个控制工序，在其中对输入条件和输出控制的顺序进行编辑。工序推进后，前工序不再执行，因此可以通过各个工序的简单顺序来控制机械。状态S和驱动指令的动作，SFC程序中，用状态表示机械运行的各个工序。当状态为ON时，在此连接的梯形图动作，当状态为OFF时，与此连接的内部梯形图不发生动作。当各个状态设置的条件(转移条件)被满足时，下一个状态为ON，此前如果状态为ON则为OFF。在状态的转移过程中，仅仅在一个瞬间两个状态会同时ON。转移前的状态在转移后的下一个运算周期OFF。不能重复使用同一个状态编号。

特殊功能模块是为了实现特殊功能如AD、DA转换，输入输出定位等的模块，其带有自有的CPU和处理电路，和基本单元进行通信。PLC是依靠内部的位软元件和字软元件的使用来进行PLC的输出控制的。因此特殊功能模块是为了实现特殊功能如AD、DA转换，输入输出定位等的模块，其带有自有的CPU和处理电路，和基本单元进行通信。PLC是依靠内部的位软元件和字软元件的使用来进行PLC的输出控制的。因此，为了对外界的温度、电压、电流、流量等连续变化的的物理量进行控制，必须采用模拟量控制单元。特殊功能模块分为模拟量输入模块、模拟量输出模块、模拟量输入输出模块、温度传感器输入模块、温度调节模块等。特殊功能模块通过PLC的FROM/TO指令对特殊功能模块的缓冲存储器的数据进行信息交换。FROM指令用于把特殊功能模块的缓冲存储器的信息读书其当前值或者是状态信息。TO指令用于把数据及信息写入到特殊功能模块的缓冲存储器中。特殊功能模块内置了缓冲存储器，其作用就是用来与PLC基本单元进行交互，为了对外界的温度、电压、电流、流量等连续变化的的物理量进行控制，必须采用模拟量控制单元。特殊功能模块分为模拟量输入模块、模拟量输出模块、模拟量输入输出模块、温度传感器输入模块、温度调节模块等。特殊功能模块通过PLC的FROM/TO指令对特殊功能模块的缓冲存储器的数据进行信息交换。FROM指令用于把特殊功能模块的缓冲存储器的信息读书其当前值或者是状态信息。TO指令用于把数据及信息写入到特殊功能模块的缓冲存储器中。特殊功能模块内置了缓冲存储器，其作用就是用来与PLC基本单元进行交互。

4.系统架构

圆柱测量数据的图像化显示系统基于C++编程语言，在其上封装的框架Qt，利用面向对象编程在系统之上进行GUL编程。基于面向对象的编程思想，利用类的继承和多态。

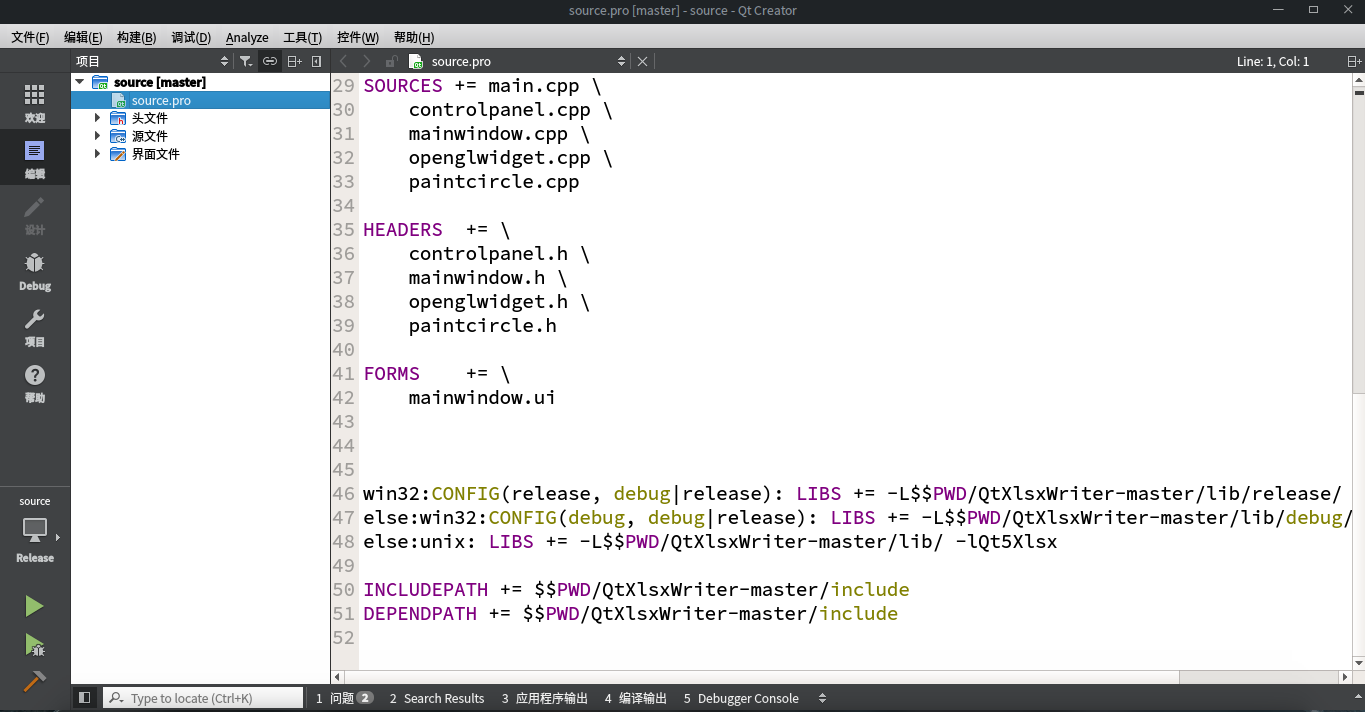


4.1 C++简介

C++是一种中级语言， 由Bjarne Stroustrup于1979年在贝尔实验室设计开发的。C++进一步扩从了C语言，是一种面向对象的语言。C++可以在多个平台上使用。C++是一种静态类型的、编译式的、通用的，支持面向过程编程，面向对象编程以及泛型编程。它中和了高级语言和中级语言的特性。C++完全支持面向对象的程序设计，支持封装、抽象、继承和多态。C++包括核心语言，提供了所有的构件，包括变量，数据类型等等、C++标准库，提供了大量的函数，用于操作文件、字符串等等、标准模板库STL，提供了大量的方法，用于操作数据结构。

4.2 环境设置

本地环境设置，C++语言环境必须包括文本编辑器和C++编译器。文本编译器用于输入程序。文本编译器包括NotePad、EMACS和vim/vi等等，并且在不同的操作系统上，文本编译器的名称和版本都会有所不同。通过编辑器创建的 文件通常称为源文件，源文件包含程序源码。C++程序源代码的后缀为.cpp、.cc或者.cp。C++编译器用于将人类可读的源代码编译为机器可读的机器语言，这样CPU可以按给定的指令执行程序。C++编译器用于把源代码编译为最终的可执行程序。大多数的C++编译器对源文件的后缀名并不在意，一般来说大家都是默认为.cpp。最常用的免费可用的编译器是GNU的C/C++编译器, 非常适用C++语言的编译。在Linux或Unix系统上 使用 g++ -v命令来检查系统是否安装了GCC。

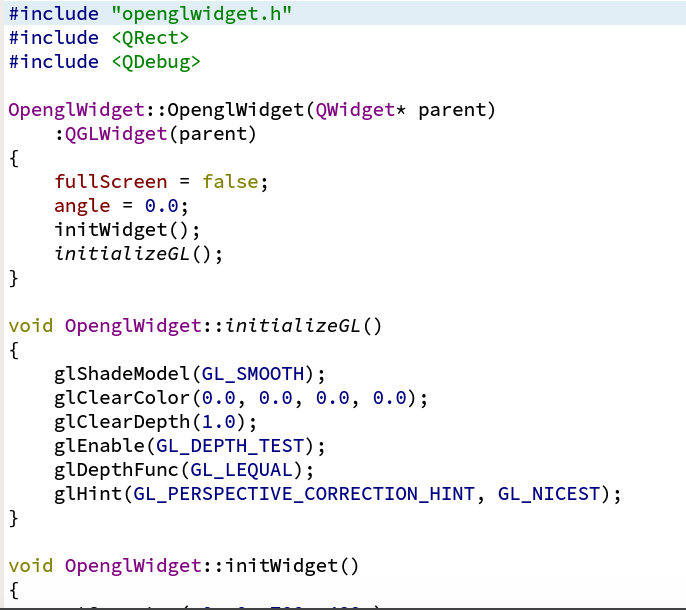


4.3 C++ 基础

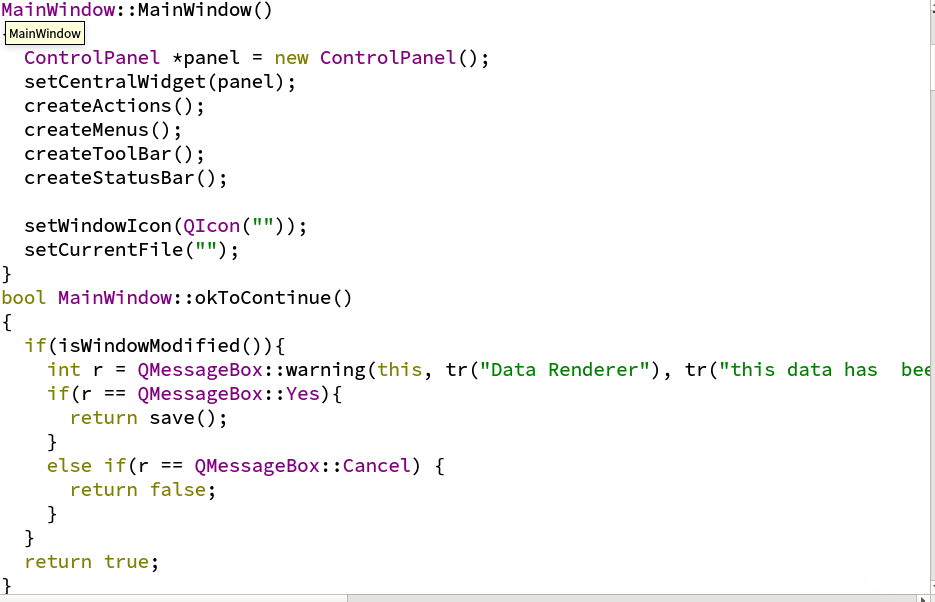
C++程序可以定义为对象的集合。这些对象通过调用彼此的方法进行数据之间的交互。对象是具有状态和行为的一种数据结构，对象是一个类的实例。类是定义了对象行为和状态的模板。方法表示一种行为，一个类可以包括多种方法。C++语言定义了一些头文件，这些头文件包含了程序必须的信息。在C++中，分号是语句结束符。也就是说，每个语句必须以分号结束，块是一组使用大括号括起来的语句，C++不以行末作为结束符的标志，因此，一行可以使用多个语句。main 函数的返回值是返回给主调进程，使主调进程得知被调用程序的运行结果。标准规范中规定 main 函数的返回值为 int，一般约定返回 0 值时代表程序运行无错误，其它值均为错误号，但该约定并非强制。如果程序的运行结果不需要返回给主调进程，或程序开发人员确认该状态并不重要，比如所有出错信息均在程序中有明确提示的情况下，可以不写 main 函数的返回值。在一些检查不是很严格的编译器中，比如 VC, VS 等，void 类型的 main 是允许的。不过在一些检查严格的编译器下，比如 g++, 则要求 main 函数的返回值必须为 int 型。所以在编程时，区分程序运行结果并以 int 型返回，是一个良好的编程习惯。在 C++ 中 int main() 和 int main(void) 是等效的，但在 C 中让括号空着代表编译器对是否接受参数保持沉默。在 C 语言中 main() 省略返回类型也就相当说明返回类型为 int 型，不过这种用法在 C++ 中逐渐被淘汰。虽然 void main（）在很多系统都适用，但他毕竟不是标准的，所以应该避免这种用法, 应该使用这种 int main(void) 的写法比较妥当。是一种 整型 类型，里面保存的是一个整数，就像 int, long 那样。这种整数用来记录一个大小(size)。size\_t 的全称应该是 size type，就是说 一种用来记录大小的数据类型。通常我们用 sizeof(XXX) 操作，这个操作所得到的结果就是 size\_t 类型。

因为 size\_t 类型的数据其实是保存了一个整数，所以它也可以做加减乘除，也可以转化为 int 并赋值给 int 类型的变量。类似的还有 wchar\_t, ptrdiff\_t。

wchar\_t 就是 wide char type， 一种用来记录一个宽字符的数据, ptrdiff\_t就是 pointer difference type， 一种用来记录两个指针之间的距离的数据类型 。通常，size\_t 和 ptrdiff\_t 都是用 typedef 来实现的。你可能在某个头文件里面找到类似的语句：typedef unsigned int size\_t;而 wchar\_t 则稍有不同。在一些旧的编译器中，wchar\_t 也可能是用 typedef 来实现，但是新的标准中 wchar\_t 已经是 C/C++ 语言的关键字，wchar\_t 类型的地位已经和 char, int 的地位等同了。在标准 C/C++ 的语法中，只有 int float char bool 等基本的数据类型，至于 size\_t, 或 size\_type 都是以后的编程人员为了方便记忆所定义的一些便于理解的由基本数据类型的变体类型。

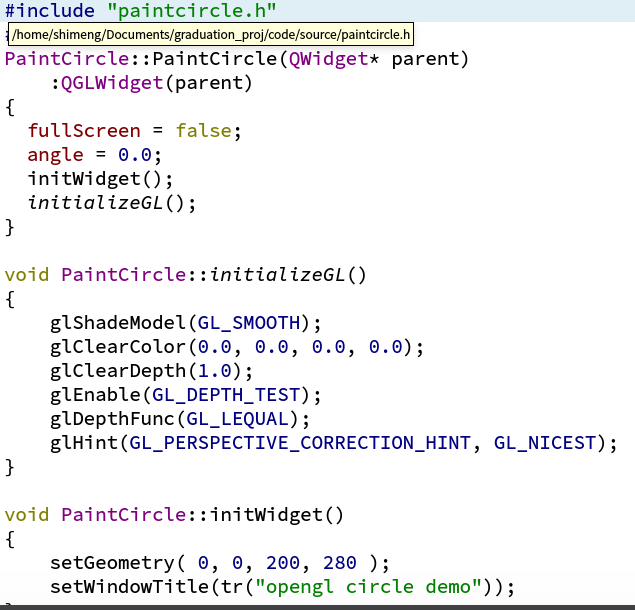


在程序中，局部变量和全局变量的名称可以相同，但是在函数内的局部变量和全局变量是两个独立的变量，互不影响。当变量间出现重名的情况下，作用域小的屏蔽作用域大的，但是由于在块中申请的变量作用域只限于当前块。C++提供了关键字explicit，可以阻止不应该允许的经过转换构造函数进行的隐式转换的发生。声明为explicit的构造函数不能在隐式转换中使用。C++中， 一个参数的构造函数(或者除了第一个参数外其余参数都有默认值的多参构造函数)， 承担了两个角色。 1 是个构造器 ，2 是个默认且隐含的类型转换操作符。所以， 有时候在我们写下如 AAA = XXX， 这样的代码， 且恰好XXX的类型正好是AAA单参数构造器的参数类型， 这时候编译器就自动调用这个构造器， 创建一个AAA的对象。这样看起来好象很酷， 很方便。 但在某些情况下（见下面权威的例子）， 却违背了我们（程序员）的本意。 这时候就要在这个构造器前面加上explicit修饰， 指定这个构造器只能被明确的调用/使用， 不能作为类型转换操作符被隐含的使用。std 是标准库函数使用的命名空间，是 standard（标准）的缩写。using namespace std ，它声明了命名空间 std，后续如果有未指定命名空间的符号，那么默认使用 std，这样就可以使用 cin、cout、vector 等。假设你不使用预处理 using namespace std;,就要加上 std::cin 或者 std::cout。cin 用于从控制台获取用户输入，cout 用于将数据输出到控制台。cin 是输入流对象，cout 是输出流对象，它们分别可以用 >> 和 <<，是因为分别在其类中对相应运算符进行了重载。



关于C++中this指针的理解：当你进入一个房子后，你可以看到房子内的桌子、椅子、地板等；但是你看不到房子的全貌；对于类来说，你可以看到成员函数、成员变量，但你看不到实例本身，但是应用this可以让我们看到这个实例本身。我的理解：class类就好比这座房子，this就好比一把钥匙，通过钥匙来打开了这座房子的门，那么里面的东西就随意你取用了。因为this作用域是在类的内部，自己声明一个类的时候，还不知道实例化对象的名字，所以用this来使用对象变量的自身。在非静态成员函数中，编译器在编译的时候加上this作为隐含形参，通过this来访问各个成员（即使你没有写上this指针）。C++ 中的 struct 对 C 中的 struct 进行了扩充，它已经不再只是一个包含不同数据类型的数据结构了，它已经获取了太多的功能。struct 能包含成员函数吗？ 能！struct 能继承吗？ 能！！struct 能实现多态吗？ 能！！！既然这些它都能实现，那它和 class 还能有什么区别？最本质的一个区别就是默认的访问控制，体现在两个方面：1）默认的继承访问权限。struct是public的，class是private的。2）struct 作为数据结构的实现体，它默认的数据访问控制是 public 的，而 class 作为对象的实现体，它默认的成员变量访问控制是 private 的。注意我上面的用词，我依旧强调 struct 是一种数据结构的实现体，虽然它是可以像 class 一样的用。我依旧将 struct 里的变量叫数据，class 内的变量叫成员，虽然它们并无区别。其实，到底是用 struct 还是 class，完全看个人的喜好，你可以将你程序里所有的 class 全部替换成 struct，它依旧可以很正常的运行。但我给出的最好建议，还是：当你觉得你要做的更像是一种数据结构的话，那么用 struct，如果你要做的更像是一种对象的话，那么用 class。

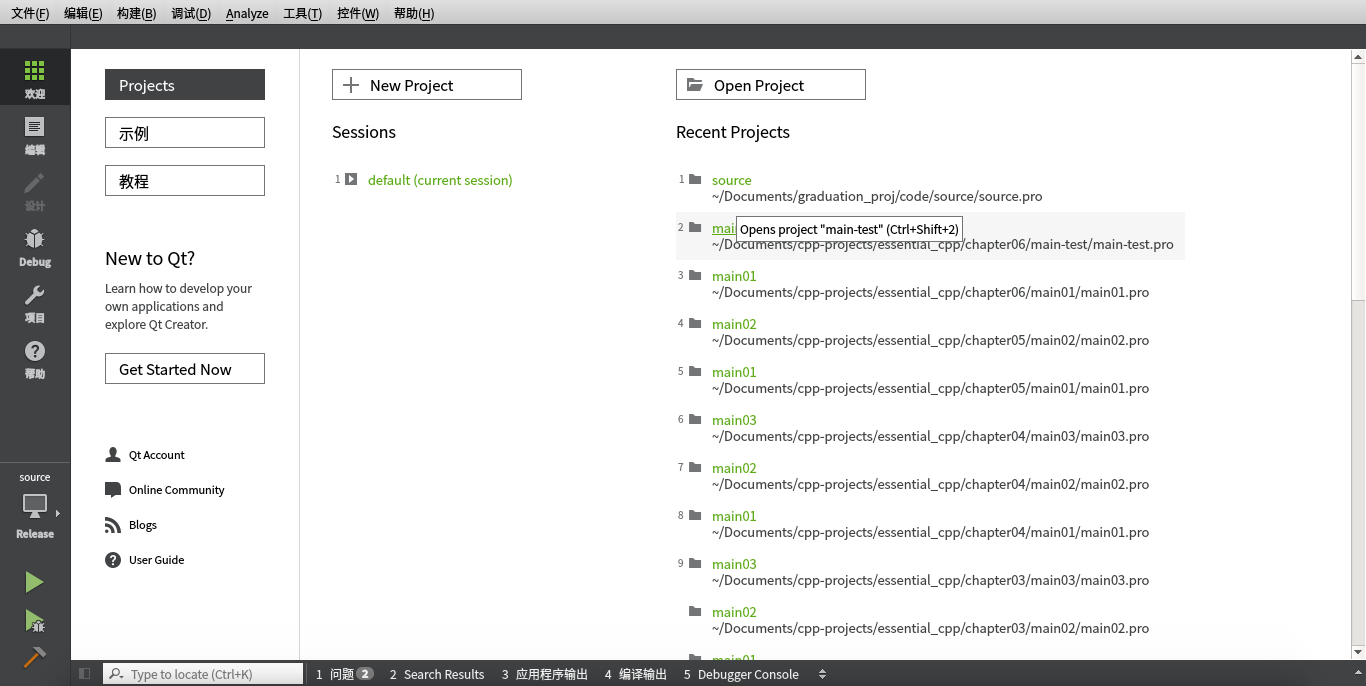
当然，我在这里还要强调一点的就是，对于访问控制，应该在程序里明确的指出，而不是依靠默认，这是一个良好的习惯，也让你的代码更具可读性。说到这里，很多了解的人或许都认为这个话题可以结束了，因为他们知道 struct 和 class 的“唯一”区别就是访问控制。很多文献上也确实只提到这一个区别。但我上面却没有用“唯一”，而是说的“最本质”，那是因为，它们确实还有另一个区别，虽然那个区别我们平时可能很少涉及。那就是：“class” 这个关键字还用于定义模板参数，就像 “typename”。但关键字 “struct” 不用于定义模板参数。这一点在 Stanley B.Lippman 写的 Inside the C++ Object Model 有过说明。问题讨论到这里，基本上应该可以结束了。但有人曾说过，他还发现过其他的“区别”，那么，让我们来看看，这到底是不是又一个区别。还是上面所说的，C++ 中的 struct 是对 C 中的 struct 的扩充，既然是扩充，那么它就要兼容过去 C 中 struct 应有的所有特性。也就是说 struct 可以在定义的时候用 {} 赋初值。那么问题来了，class 行不行呢？将上面的 struct 改成 class，试试看。报错！噢~于是那人跳出来说，他又找到了一个区别。我们仔细看看，这真的又是一个区别吗？你试着向上面的 struct 中加入一个构造函数（或虚函数），你会发现什么？对，struct 也不能用 {} 赋初值了。的确，以 {} 的方式来赋初值，只是用一个初始化列表来对数据进行按顺序的初始化，如上面如果写成 A a={'p',7}; 则 c1,n2 被初始化，而 db3 没有。这样简单的 copy 操作，只能发生在简单的数据结构上，而不应该放在对象上。加入一个构造函数或是一个虚函数会使 struct 更体现出一种对象的特性，而使此{}操作不再有效。事实上，是因为加入这样的函数，使得类的内部结构发生了变化。而加入一个普通的成员函数呢？你会发现{}依旧可用。其实你可以将普通的函数理解成对数据结构的一种算法，这并不打破它数据结构的特性。那么，看到这里，我们发现即使是 struct 想用 {} 来赋初值，它也必须满足很多的约束条件，这些条件实际上就是让 struct 更体现出一种数据机构而不是类的特性。那为什么我们在上面仅仅将 struct 改成 class，{} 就不能用了呢？其实问题恰巧是我们之前所讲的——访问控制！你看看，我们忘记了什么？对，将 struct 改成 class 的时候，访问控制由 public 变为 private 了，那当然就不能用 {} 来赋初值了。加上一个 public，你会发现，class 也是能用 {} 的，和 struct 毫无区别！！！做个总结，从上面的区别，我们可以看出，struct 更适合看成是一个数据结构的实现体，class 更适合看成是一个对象的实现体。



C++多态意味着调用成员函数时，会根据调用函数的对象的类型来执行不同的函数；形成多态必须具备三个条件：1、必须存在继承关系；2、继承关系必须有同名虚函数（其中虚函数是在基类中使用关键字Virtual声明的函数，在派生类中重新定义基类中定义的虚函数时，会告诉编译器不要静态链接到该函数）；

3、存在基类类型的指针或者引用，通过该指针或引用调用虚函数；纯虚函数声明如下： virtual void funtion1()=0; 纯虚函数一定没有定义，纯虚函数用来规范派生类的行为，即接口。包含纯虚函数的类是抽象类，抽象类不能定义实例，但可以声明指向实现该抽象类的具体类的指针或引用。虚函数声明如下：virtual ReturnType FunctionName(Parameter) 虚函数必须实现，如果不实现，编译器将报错，对于虚函数来说，父类和子类都有各自的版本。由多态方式调用的时候动态绑定。实现了纯虚函数的子类，该纯虚函数在子类中就编程了虚函数，子类的子类即孙子类可以覆盖该虚函数，由多态方式调用的时候动态绑定。虚函数是C++中用于实现多态(polymorphism)的机制。核心理念就是通过基类访问派生类定义的函数。在有动态分配堆上内存的时候，析构函数必须是虚函数，但没有必要是纯虚的。友元不是成员函数，只有成员函数才可以是虚拟的，因此友元不能是虚拟函数。但可以通过让友元函数调用虚拟成员函数来解决友元的虚拟问题。析构函数应当是虚函数，将调用相应对象类型的析构函数，因此，如果指针指向的是子类对象，将调用子类的析构函数，然后自动调用基类的析构函数。

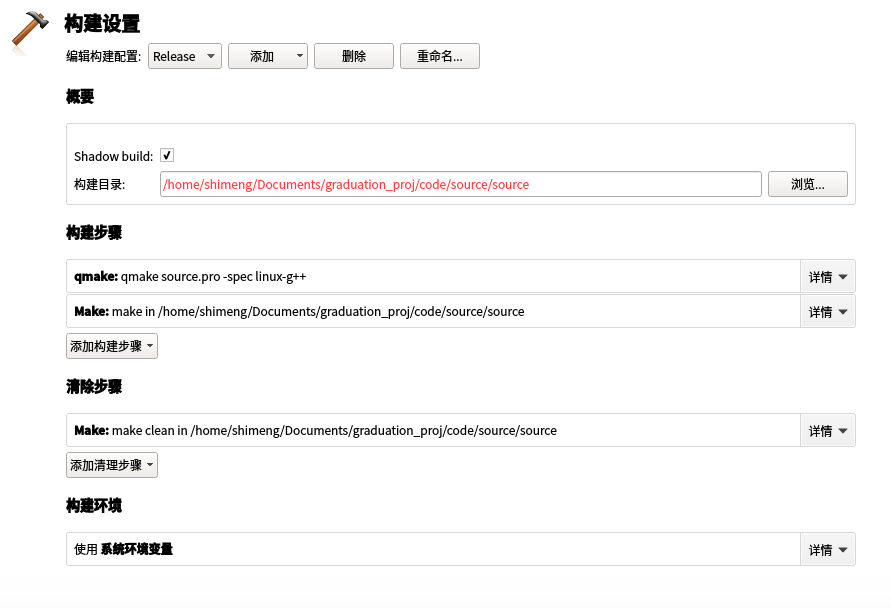
4.4 Qt框架



Qt是一个跨平台的C++图形用户界面库，是基于面向对象的C++语言，它提供了信号(signal)和槽(slot)的对象通信机制，具有可查询和可设计的属性，以及强大的事件和事件过滤器。Qt包括大量的联机参考文档，有超文本HTML方式，UNIX帮助页、man手册。其他的工具在应用时都存在一个普遍的问题，就是经常没有真正适合需求的控件，生成的自定义控件对用户来说，也是一个黑匣子。在Qt中，能够创建控件，具有绝对的优越性，生成自定义控件非常简单，并且容易修改控件。由于Qt是一种跨平台的GUI工具包，所以，它对编程者隐藏了在处理不同窗口系统时的潜在问题。为了将基于Qt程序更加方便，Qt包含了一系列的类，该类能够使程序员避免了在文件处理、时间处理等方面存在依赖操作系统方面的问题。Qt也为本地化应用提供了完全的支持，所有用户界面的文本或字符串都可以利用翻译工具将其翻译为各国语言。为了适合用户的需求，Qt提供了丰富的API函数，用于专门的GUI。Qt还提供了基于模板的初始化、文件和通用的I/O设备、目录管理、日期/时间类、常用表达式解析等。目的是利用这些类，建立或生成不同的功能，用它们来实现Qt的通用化。除此之外，也可以利用STL标准模板库。Qt编程的基本模块为控件，一个控件是一个用户界面的组成部分，Qt包含用于创建专业外观的用户界面所需的所有控件。Qt能执行一些基本的任务，比如图形的润色，比一般的基于平台的代码要快。大多数的GUI工具包是基于分层的方法。比如，工具包为本地窗口系统组件提供了很多C++类，这些结构使组件的继承性和通用性变得很差。在层次化的工具包中，GUI功能常成为所有使用的窗口系统所必须的最普遍的基础。Qt仿效本地串口系统的组件，这是一种非常复杂的技术。Qt还提供一些更有用的函数，使用于多平台。

4.4.1 Qt环境配置

Qt Creator作为Qt官方的开发环境，对Qt各方面支持都非常完善，并且使用体验非常好。Qt Creator主界面分为6个模式：欢迎模式、编辑模式、设计模式、Debug模式、项目模式和帮助模式，分别由左边的六个图标进行模式的切换。刚进去显示的是欢迎界面，有一些入门教程、开发的项目列表、Qt提供的示例程序，单击一个示例程序可以直接打开至这个项目。



4.4.2 Qt面向对象

Qt具有模块设计和控件或元素的可重用性的特点。一个控件不需要知道它的内容和用途，通过signal和slot与外界通信、交流。并且所有的Qt都可通过继承。Qt包括一些抽象的类，对象模型是框架的根本。C++标准中虽然有很多的对实时的对象模型的支持，但其静态的特性，导致其仍然缺乏灵活性，Qt提供了自己的对象模型。主要包括对象树，对象属性，以及元对象系统等。1）对象树（Object tree ），由于GUI的设计层次结构比较强，并需要兼顾效率，Qt设计了对象树，并支持了动态类类型转换，其中父对象与子对象相互指向，整体的结构关系相当于一个森林， 父对象与子对象为一对多的关系，并有多个平行的父对象。其中有两点需要注意，首先当父对象析构的时候将析构所有的子对象，如果子对象在栈中或者为全局变量，有可能导致重复析构，程序崩溃（ C++标准中，局部变量析构的顺序是构造过程的逆向 ）；其次如果事件在发送中将QObject( QObject作为其最终的基类，源自官方Qt文档中~QObject() ）析构，也可能导致多线程调用过程中的崩溃。2）对象属性（Object properties ），对象属性是基于元对象系统（ Meta-Class system ，信号槽机制 signals and slots 也基于该模块, 在后边的章节将会具体谈），Qt中的Q\_PROPORTY(...)宏标记了相关属性信息，将其注册到QMetaObject中，QMetaObject记录了所有注册过的属性信息，允许程序在编译时不知情的一些信息，在实时运行阶段能够动态添加进来。相当于一个盒子，假设我是盒子的设计者，我并不知道将来盒子里要放什么东西，设计的要求是，我可以放新的东西进来也可以添加已有的东西，也可以将盒子更改我自定义的样式。3）元对象系统（ meta-class system ），元对象系统负责信号槽机制，实时类型判断，以及对象属性。这个系统依赖于MOC（meta object complier），MOC“阅读”代码，将所有标记信息读入，整理，来满足上面所述3个方面的应用。在QObject所属子类中，所有拥有QMetaObject的类都支持反射模式（并不被C++标准支持），但Qt通过MOC间接支持了该设计模式，QMetaObject提供了类的属性以及方法描述信息。

4.4.3 Qt组件间的通信

Qt提供signal和slot概念，这是一种安全可靠的方法，它允许回调，并支持对象之间在彼此不知道对方信息的情况下，进行合作，这使Qt非常适合于控件编程。信号与槽是多对多的关系，通过connect将信号与槽链接在一起，这样的对应关系维护了指定的函数参数，信号emit后就不再关心是谁来处理，这个是一个单向的过程，不过槽函数可以继续emit信号，从而形成一个环，达到函数级通信的目的。因为信号机制无法返回结果，所以槽函数必须是统一的无返回类型。meta-object system维护了一个信号以及对应函数的关系列表，取代了回调函数，当发射信号signal1后，程序通过元对象系统定位到对应的函数，好比如说我有一个储物柜，分为很多的小格子，每个格子都有自己的编号，我现在将记录有格子编号的物品交给管理员，管理员自然就知道将物品放到哪个格子中了。在底层实现方面，meta-object complie( moc )编译期间，生成了专门的CPP文件，扩展meta class，记录函数所在偏移，并记录对应关系表。 整个信号槽机制依赖于MOC，并没有使用C++的模板的泛型编程，主要原因虽然效率要比模板稍差，但稳定性，扩展性都要比模板好。

4.4.3.1 槽

当信号被发出时，会调用与之相连接的槽。槽是普通的C++函数，可以用普通的方式来调用。它唯一特殊的地方在于可以与信号相连接。槽的参数不能有默认值。同样，信号的参数也不能有默认值。在槽的参数中尽量不使用自定义的数据类型，因为这样将会使通用性降低。 既然槽和普通的成员函数差不多，它们和普通成员函数一样有访问限制，根据槽的访问限制谁可以与它们相连接，能够分为以下三种情况。public slot: 任何信号都可以与之相连接。这在窗口部件编程中非常有用，用于创建一些对彼此一无所知的对象，只有通过信号和槽来交换信息。public slot就像是标准的铁路一样。protected slot: 只有该类及其子类所派生的对象的信号才可以与之相连接。这类槽的目的通常是为了类的完善，而不是类与外界的接口。private slot: 只有该类自己的信号才可以与之相连接。当然还可以将定义为virtual，这将非常有用。信号和槽是相当高效的。当然，它们与“实时”的回调函数相比，在增加了灵活性的同时也损失了一些速度，正所谓有利必有弊，但是这种速度的损失相当微不足道。因此，信号槽机制具有的简便性和灵活性的特性，使用信号和槽是用户交互的必然选择。

4.4.3.2 信号

当某个信号对其客户或所有者发生的内部状态发生改变，信号被一个对

象发射。只有定义过这个信号的类及其派生类能够发射这个信号。当一个

信号被发射时，与其相关联的槽将被立刻执行，就象一个正常的函数调用

一样。信号-槽机制完全独立于任何GUI事件循环。只有当所有的槽返回以

后发射函数(emit)才返回。如果存在多个槽与某个信号相关联，那么，当

这个信号被发射时，这些槽将会一个接一个地执行，但是它们执行的顺序

将会是随机的、不确定的，我们不能人为地指定哪个先执行、哪个后执行。例如，一个列表框可以发出highlighted()和activated()的信号，大多数对象也许只对activated()的信号感兴趣，但也许有些对象需要知道该列表框中的哪一项被选中了。如果有两个不同的对象对一个信号感兴趣，只要将该信号连接到这两个对象的槽上就可以了。当一个信号被发出的时候，与之相连接的槽就立即执行，就像通常的函数调用一样。信号/槽机制与任何图形用户界面的事件循环完全无关。当所有的槽返回之后，才返回到发出信号的地方。如果有多个槽与一个信号相连接，则这些槽将被一个接一个地执行，而且其执行顺序是随意的。信号是由MOC自动生成的，必须在.cpp文件中实现，而且永远没有返回值。

编写代码创建将要发送信号的类的实例。只有这个实例的存在，才能把槽和信号联系在一起。把槽和信号连接起来。这个工作通常在构造函数中完成，但是如果这个对象构造得比较晚，那么连接工作也可以晚点做。调用connect()方法把你的槽加入到方法列表中，每当指定的信号发出的时候，这个方法就会被调用。无论任何使用emit发送信号，就好象是你编写的程序直接调用每一个槽方法一样。也就是说，直到槽方法返回，你编写的程序才能继续执行。因此，通常应当保持在槽方法内部的处理过程中尽可能的简单，这样才不会因此中止信号的发送。发送信号的可能是用户接口过程，操作过程表现得比较慢或者缓慢。如果一个槽方法发送一个信号，此信号直接

或者间接地执行了发送一个信号的方法，而这个信号又被最开始的槽所接

收，那么信号将连续不断的调用槽，你编写的程序就会崩溃。比如：如果

名为firstfun()的方法发送了一个A信号，A信号被second()槽所接收，而

second()槽发送了信号B，最后，名为firstfun()的方法接收了信号B，这样

就产生了一个死循环。这种循环将一直执行，直到该程序崩溃为止(或者用

户进入长时间的等待)。还需要小心槽和信号方法在连接语句中的参数是否匹配。当程序运行的时候，直到试着去解决一个问题时，才可能得到出错的信息。为了避免这个问题的出现，必须确定每次增加内容的时候，都要进行测试，或者改变槽和信号部分。唯一的出错信息是当connect()方法找不到匹配对象的时候，输出一个写入控制台上的字符串。此后，程序就忽略了这个信号的存在。只有从命令行运行程序的时候，才能够看到控制台输出的信息。

4.3.4 Qt的组成部分

QtCore不但包含QString、QList和QFile等工具类，而且还包含QObject和QTimer等内核。因为QApplication类有refactored, 所以它能够在非GUI应用程序中。它将拆分为：QCoreApplication(在QtCore中)和QApplication(在QtGui中)。这种拆分将使用Qt开发服务器应用程序，无需连接所有多余的与GUI相关的代码，无需求与GUI相关的系统程序将要放到当前的目标机器中称为可能。qmake命令生成Makefile文件,qmake将默认连接到应用程序中依赖的Qtcore 和QtGui中。当想删除具有依赖关系的GUI时，可以在.profile文件中加入QT -= gui; 如果想使用其他的库文件，在.profile中加入 QT += network opengl; 如果要编译.ui文件，也需要在.pro文件中 加入 CONFIG += uic3。

QPushButton推动命令按钮是图形用户界面中最常用的一个窗口组件。它用来命令计算机去执行一些操作，或者回答一些问题。该按钮是一个矩形框，通常显示一个文本标签来描述它的操作。标签中有下划线的字母表明快捷键。QRadioButton组件为单选按钮提供了一个文本标签。QRadioButton是一个能够切换或者关的选择按钮。通常单选按钮是用于当前用户选择许多中的一个选项。同一时间，单选按钮组中仅仅只有一个按钮被选择；如果选择其他按钮，以前选择的按钮将要切换为关闭状态。当一个按钮切换开或关的时候，它就会发射toggled（）信号。如果任何时刻想触发一个行为来改变按钮的状态的时候，可以连接这个信号。否则，使用isChecked()来查看是否有其他按钮被选择。QCheckBox组件提供一个带文本标签的复选按钮，QCheckBox和QRadioButton的区别是在于对用户选择的限制。单选按钮定义了”多选一”的选择，而复选按钮定义了”多选多”的选择。QButtonGroup在视觉上能够把许多的复选按钮组合在一起，成为一个按钮组。无论复选按钮选中或者是未选中，它都会发射一个信号toggled()。在任何时刻，如果想要改变复选按钮的状态的时候，需要连接toggled()信号触发这个行为。除了常用的选中和未选中两个状态，QCheckBox还可以任意选择第三种状态来表明”没有改变”。当需要给用户一个选中或者未选中复选按钮的选择时，这是非常有用的。如果需要第三种状态，可以通过setTristate()来使它生效，利用checkState()函数来查询当前的状态。

标签组件QLabel提供了显示文本或者图像，没有用户交互的功能。因为标签的外观可以通过各种各样的方式来配置，而且可以制定焦点记忆键到另外的窗体上。QLabel是经常为交互式窗体制作标签。为了交互式窗体的使用。QLabel提供了添加助记符的有用机制，这种机制将设置键盘焦点到另外一个窗体上。 QLCDNumber窗口部件能够在LCD上面显示十进制、十六进制、八进制、二进制的数字。可以利用display()槽与数据源进行连接，能够重载五个任意类型的display()。虽然能够通过value()来检索数值，但是QLCDNumber对象的内容不能够被检索。如果您真正地需要文本，我们建议您使用信号-槽的机制来实现，可以将display()槽连接到其他槽中，并且将其存放在那里。

QSpinBox类提供了一个微调框的窗体。QSpinBox类是用来处理整数和设置离散集合的数值(例如：月份);利用QDoubleSpinBox可以处理浮点数值。绝大多数微调框是有方向的，但是QSpinBox也可以像使用一个圆形的微调框一样。例如：如果微调框的范围是0到99，并且微调框当前的数值为99，wrapping()函数设置为真，点击“向上”按钮，或者点击键盘向上的方向键，该微调框的数值就会变为0。如果想改变微调框具有这样圆形的功能，可以利用setWrapping()函数来实现。微调框有一个特殊的功能是：可以在需要实现数字的中添加任意的符号。例如：货币或者测量单位，可以参考setPrefix()和setSuffix()函数。如果您要获得微调框的文本信息，您可以使用text()函数(它包括所有prefix()和suffix())，或者使用cleantext()函数(它没有prefix()和suffix(),并且在开头和结尾没有空白)。每当微调框需要显示数值v,就要使用虚函数textFromValue()。该函数在默认的情况下，返回的是一个包含有参数v的字符串。如果需要将其输出，可以利用QLocale().toString(v)函数，将其标准的打印出来。

QLineEdit窗体组件是一个单行文本编辑框。单行文本编辑框允许用户通过有用的编辑函数来输入和编辑单行纯文本，该编辑函数包括撤消和重做、剪切和粘贴、拖动和放下。可以利用setText()或者insert()函数来改变文本，利用text()函数来获取文本编辑框的信息，也可以利用displayText()函数获取文本显示的信息。

QMenu类提供了一个菜单窗体组件，该组件用于菜单栏、上下文菜

单和弹出式菜单。菜单窗体组件能够选择菜单，在菜单栏中可以有一个下拉菜单，或者有一个独立的上下文菜单。当用户点击下拉菜单的各自选项，或者使用指

定的快捷键时候，就会显示菜单栏中的下拉菜单。使用QMenuBar::addMenu()函数将菜单添加到菜单栏中。上下文菜单通常是由某一个专门的键盘按键，或者通过使用点击鼠标右键来实现的。想要实现上下文菜单，就要执行popup()函数，或者exec()函数。菜单由一个选项列表组成，使用addAction()函数将选项添加到选项列表中。另外，选项也可以有文本标签，在文本标签的左边可以任意添加图标，也可以给选项添加快捷键，例如“Ctrl+X”。如果想将每个选项分隔开来，那应该如何去做呢？可以使用分割符。分隔符---显示一个子菜单的选项，并且可以在选项中执行某个操作。如果想插入一个分隔符，可以利用函数addSeparator()来实现。如果想添加子菜单，可以利用addMenu()函数来实现。当插入一个选项的时候，通常要指定接收者和槽。当选项发出信号

triggered()的时候，要告诉接收者。

Qt部件不同于其他用户交互方式的GUI工具包。用户交互方式是所有

GUI(Graphical user interface)应用程序关心的问题。通过将某种用户事件

(比如按下鼠标)与程序事件(比如退出程序)联系起来，使用户能够在图形界面中只使用鼠标来控制程序。而其他工具包是利用回调函数来进行用户交互的。所谓的回调是指：你自己定义一个函数，并告诉系统何时为何调用。你可以写一个特定数量和类型参数的函数，然后告诉系统何时使用，并传递给它所需的参数，系统就会调用你定义的函数，处理参数，并给你返回值。所谓的回调函数是指：按照一定的形式由开发人员定义并编写实现内容。使用回调函数，实际上就是在调用某个函数(通常是API函数)时，将自己的一个函数(也就是回调函数)的地址作为参数传递给那个函数。而那个函数在需要的时候，也就是某种事情发生的时候，利用传递的函数地址调用回调函数，这时开发人员可以利用这个机会在回调函数中处理消息或完成一定的操作。回调函数只能是全局函数，或者是静态函数，因为这个函数只是在类中使用，所以为了维护类的完整性，我们用类的静态成员函数来做回调函数。虽然回调函数能够实现用户之间的交互。但是，回调函数非常复杂，容易混淆，又难以理解(至少大部分编写Qt的工作人员或者程序员有过这样的想法)。因此，Qt的开发者使用另一种方法来完成这一工作。这种方法依赖

于Qt特有的两个功能，信号和槽。使用这种新方法是非常简单的，只需要

编写一行代码就能够将用户事件和程序事件连接起来。这种将用户事件连接到程序事件的方法要比回调函数，更加容易使用的两个重要原因是：1> 槽和信号是调用Qt库中的函数。2> 信号和槽不使用其他工具包。

信号和槽用于对象间的通信，该机制是Qt的一个中心特征，并且最能体

现Qt与其他工具包它们之间有什么不同。在图形用户界面编程中，我们经

常希望一个窗口部件的一个变化被告知给另外一个窗口部件。简单的说，

我们希望任何一类的对象可以和其他对象进行通信。例如：我们正在解析

一个XML文件，当我们遇到一个新的标签的时候，我们也许希望告知列表

视图，我们正在用来表达XML文件的结构。比较老的工具包使用一种被称做回调的通信方式来实现同一目的。回调是指一个函数的指针，所以如果你希望一个处理函数告知你一些事件，你可以把另一个函数(回调函数)的指针传递给处理函数。处理函数在适当的时候回调。回调有两个主要缺点：1> 它们不是类型安全的。我们从来都不能确定处理函数使用了正确的参数来调用回调。2> 回调和处理函数是非常强有力的联系在一起的，因为处理函数必须要知道调用哪个回调。

5.辅助工具

在开发的过程中，并非是将直接在编辑器中输入所有代码，之后编译生成等步骤。同时我们还需要版本控制、单元测试等其他辅助工具，只有当这些部分完成后，一个完整的软件开发才算是完成了。

5.1 版本控制

什么是版本控制？我为什么要关⼼它呢？版本控制是⼀种记录⼀个或若⼲⽂件内容变 化，以便将来查阅特定版本修订情况的系统。在本书所展⽰的例⼦中，我们仅对保存 着软件源代码的⽂本⽂件作版本控制管理，但实际上，你可以对任何类型的⽂件进⾏ 版本控制。 如果你是位图形或⽹页设计师，可能会需要保存某⼀幅图⽚或页⾯布局⽂件的所有修 订版本（这或许是你⾮常渴望拥有的功能）。采⽤版本控制系统（VCS）是个明智的 选择。有了它你就可以将某个⽂件回溯到之前的状态，甚⾄将整个项⽬都回退到过去 某个时间点的状态。你可以⽐较⽂件的变化细节，查出最后是谁修改了哪个地⽅，从 ⽽找出导致怪异问题出现的原因，又是谁在何时报告了某个功能缺陷等等。使⽤版本 控制系统通常还意味着，就算你乱来⼀⽓把整个项⽬中的⽂件改的改删的删，你也照 样可以轻松恢复到原先的样⼦。但额外增加的⼯作量却微乎其微。

许多⼈习惯⽤复制整个项⽬⽬录的⽅式来保存不同的版本，或许还会改名加上备份时 间以⽰区别。这么做唯⼀的好处就是简单。不过坏处也不少：有时候会混淆所在的⼯ 作⽬录，⼀旦弄错⽂件丢了数据就没法撤销恢复。 为了解决这个问题，⼈们很久以前就开发了许多种本地版本控制系统，⼤多都是采⽤ 某种简单的数据库来记录⽂件的历次更新差异。这种做法带来了许多好处，特别是相较于⽼式的本地 VCS 来说。现在，每个⼈都可 以在⼀定程度上看到项⽬中的其他⼈正在做些什么。⽽管理员也可以轻松掌控每个开 发者的权限，并且管理⼀个 CVCS 要远⽐在各个客户端上维护本地数据库来得轻松容 易。 事分两⾯，有好有坏。这么做最显⽽易见的缺点是中央服务器的单点故障。如果宕机 ⼀⼩时，那么在这⼀⼩时内，谁都⽆法提交更新，也就⽆法协同⼯作。要是中央服务 器的磁盘发⽣故障，碰巧没做备份，或者备份不够及时，就会有丢失数据的风险。最 坏的情况是彻底丢失整个项⽬的所有历史更改记录，⽽被客户端偶然提取出来的保存 在本地的某些快照数据就成了恢复数据的希望。但这样的话依然是个问题，你不能保证所有的数据都已经有⼈事先完整提取出来过。本地版本控制系统也存在类似问题， 只要整个项⽬的历史记录被保存在单⼀位置，就有丢失所有历史更新记录的风险。于是分布式版本控制系统（ Distributed Version Control System，简称 DVCS ）⾯世 了。在这类系统中，像 Git，Mercurial，Bazaar 以及 Darcs 等，客户端并不只提取最新 版本的⽂件快照，⽽是把代码仓库完整地镜像下来。这么⼀来，任何⼀处协同⼯作⽤ 的服务器发⽣故障，事后都可以⽤任何⼀个镜像出来的本地仓库恢复。因为每⼀次的 提取操作，实际上都是⼀次对代码仓库的完整备份。更进一步，许多这类系统都可以指定和若干不同的远端代码仓库进行交互。籍此，你就可以在同一个项目中，分别和不同工作小组的人相互协作。你可以根据需要设定不同的协作流程，比如层次模型式的工作流，而这在以前的集中式系统中是无法实现的。

5.2 单元测试

单元测试对框架的设计要求非常高，数据与代码与界面要尽可能分离，接口定义，输出与输出预期，代码覆盖度，单元测试是软件测试的基础，因此单元测试的效果会直接影响到软件的后期测试，最终在很大程度上影响到产品的质量。从如下几个方面就可以看出单元测试的重要性在何处。

时间方面：如果认真的做好了单元测试，在系统集成联调时非常顺利，因此会节约很多时间，反之那些由于因为时间原因不做单元测试或随便做做的则在集成时总会遇到那些本应该在单元测试就能发现的问题，而这种问题在集成时遇到往往很难让开发人员预料到，最后在苦苦寻觅中才发现这是个很低级的错误而在悔恨自己时已经浪费了很多时间，这种时间上的浪费一点都不值得，正所谓得不偿失。

测试效果：根据以往的测试经验来看，单元测试的效果是非常明显的，首先它是测试阶段的基础，做好了单元测试，在做后期的集成测试和系统测试时就很顺利。其次在单元测试过程中能发现一些很深层次的问题，同时还会发现一些很容易发现而在集成测试和系统测试很难发现的问题。再次单元测试关注的范围也特殊，它不仅仅是证明这些代码做了什么，最重要的是代码是如何做的，是否做了它该做的事情而没有做不该做的事情。

测试成本：在单元测试时某些问题就很容易发现，如果在后期的测试中发现问题所花的成本将成倍数上升。比如在单元测试时发现1个问题需要1个小时，则在集成测试时发现该问题需要2个小时，在系统测试时发现则需要3个小时，同理还有定位问题和解决问题的费用也是成倍数上升的，这就是我们要尽可能早的排除尽可能多的bug来减少后期成本的因素之一。

产品质量：单元测试的好与坏直接影响到产品的质量，可能就是由于代码中的某一个小错误就导致了整个产品的质量降低一个指标，或者导致更严重的后果，如果我们做好了单元测试这种情况是可以完全避免的。

6. 总结：

本文利用PLC与计算机通信，在上位机上开发一款基于圆柱测量数据显示系统。该系统通过数据采集、数据处理、数据计算、模型生成等一系列步骤，最终生成一个三维模型。在工业测量领域中，由于测量对象的千差万别，离散的测量数据很难进行描述分析，而三维显示系统可以让我们查看，研究，理解大量离散数据，从而可以有效的发现隐藏在数据内部的规律。通过拟合构建一个三维模型来逼近原型。通过给定的一组点，构造一个曲线，曲线逼近构造一个曲面，使之逐渐逼近给定的数据点。本文采用最小二乘法，拟合出圆柱的半径，从而构建出圆柱的三维模型。在实际的拟合中，数据点不可避免的引入噪声点，因此本文同样对数据进行预处理，减少噪点对拟合的影响。

本文为什么要用C++而不是C呢？对于我来说理由很现实：时间紧任务重，用C的话需要发明的轮子太多了，C++的抽象层次高，代码量少，bug相对就会更少，现代C++的内存管理完全自动，以至于从头到尾我根本不记得曾遇到过什么内存管理相关的bug，现代C++的错误处理机制也非常适合快速开发的同时不用担心bug乱飞，另外有了C++11的强大支持更是如虎添翼，其次，C++的异常不像C#的异常那样附带Callstack。例如你在某个地方通过.at(i)来取一个vector的某个元素，然后i越界了，你会收到vector内部抛出来的一个异常，这个异常只是说下标越界了，然后什么其他信息都木有，连个行号都没有。要是不抛异常直接让程序崩溃掉好歹还可以抓到一个minidump呢，这个因素一定程度上也限制了C++异常的被广泛使用。Callstack显然对于我们迅速诊断程序的bug有至关重要的作用，由于我们是一个不大的团队，所以我们对质量的测试很依赖于微软内部的dogfood用户，我们release给dogfood用户的是release版，倘若我们不用异常，用assert的话，固然是可以在release版也打开assert，但assert同样也只能提供很有限的信息（文件和行号，以及assert的表达式），很多时候这些信息是不足够理解一个bug的（更不用说还得手动截屏拷贝黏贴发送邮件才能汇报一个bug了），所以往往接下来还需要在开发人员自己的环境下试图重现bug。这就不够理想了。理想情况下，一个bug发生的时刻，程序应该自己具备收集一切必要的信息的能力。

时间过的很快，一晃大学四年的生活已接近了尾声在目，当初还是刚进大学的懵懂少年现在也长大了学到了很多，也懂得了很多。随着毕业日子的到，毕业设计也接近了尾声。经过两个月的奋战我的毕业设计终于完成了。在没有做毕业设计以前觉得毕业设计只是对这几年来所学知识的单纯总结，但是通过这次做毕业设计发现自己的看法有点太片面。毕业设计不仅是对前面所学知识的一种检验，而且也是对自己能力的一种提高。通过这次毕业设计使我明白了自己原来知识还比较欠缺。自己要学习的东西还太多，以前老是觉得自己什么东西都会，什么东西都懂，有点眼高手低。通过这次毕业设计，我才明白学习是一个长期积累的过程，在以后的工作、生活中都应该不断的学习，努力提高自己知识和综合素质。

刚拿到设计课题时有些茫然，经过了一个假期对所学知识已经淡忘了，以至于拿到课题时手足无措。经过指导老师对课题的分析，对整个方案的简单提示我们大体了解了课题内容。我们的课题主要是对单片机知识的巩固，通过单片机的软件编程来实现数字移相的控制，这是我们的课题任务。对整个课题我们进行分解，分期完成整个课题。首先是对总体方案的确定，有了明确的方案才能朝着这个方向去努力。其次是原理图的设计以及各部分流程图的绘制。最后就是各子程序及主程序的编写。

在设计期间，我们通过到图书馆、上网等方式查阅资料充实设计内容。通过整个项目的进行，对自己的这大学四年既是在原有学习上的一次总结，也是一次突破。整体从零开始，将整个毕业设计中设计到的知识学习一遍，同时有突破了自己原有的知识体系，学习了更多编程的知识。