Chapter 1

简介

本系统采用 C++ 11 、Python 3 、JavaScript 语言开发，在图形等跨平台领域，使用 Qt 5.2 、 Python 3.3 、 Boost 1.55 库。

本系统初期主要运行在 Windows XP 操作系统上，Qt 5 和 Python 3 的使用，很好地满足未来的跨平台性。

为了满足跨平台的编译管理，使用了 CMake 2.8.12 编译管理系统。

为了能在古老的 Windows XP 系统上运行使用 C++ 11 语言特性开发的程序，抛弃了 Visual C++ 2010 编译器，使用 MingGW/GCC 4.8 进行 C++ 11 语言的编译工作。

为了能在古老的 Windows XP 系统上解决嵌入 Python 3 解释器内存空间的问题，程序使用 Microsoft Visual C++ 2010 Redistributable ，将所有目标文件链接到 msvcr100.dll 和 msvcp100.dll。

为了便捷高效地进行编译，使用了 Ninja 编译系统。

当前系统可以良好的 Windows XP/2003/Vista/2008/7/2008R2/8/8.1 系统上。

通过使用 Microsoft Visual C++ 2012/2013 版本的编译器，可以编译出 64 位应用程序，进而良好运行在 Windows 7/8/8.1 64bit 系统上。

本系统可在 Mac OS X 下使用 XCode/Clang/llvm 编译，并运行在 10.6.8 以上的 Mac OS X 系统中。

本系统可在 Linux 下使用 GCC 编译器编译并运行。

本系统可通过交叉编译运行在 Android 2.3 以上版本中。

本系统可通过 XCode 编译运行在 iOS 4 以上版本中。

功能简介

基本功能

1. 读写整编数据文件。
2. 绘制“水位~流量关系线图”、“单~断沙关系线图”、“水位过程线图”、“流量过程线图”、“含沙量过程线图”和“断面高程图”。
3. 读写本程序专用项目工程文件。
4. 进行上述各种图的组合与套绘。
5. 打印本系统绘制的各种图。
6. 导出 PDF 、 SVG 、 JPG 、 PNG 、 TIF 格式的图。

扩展功能

1. 通过 Python 语言增加模块，从而可以读写更多种数据文件、计算数据并绘制更多种图。
2. 通过 JavaScript 定义用户脚本，将复杂重复的人工操作自动化。

Chapter 2

设计概览

考量1： 语言和类库的选择

即使到现在，在桌面程序领域， C#/Java 一类语言的效率和可移植性，还是比不上 C/C++ 的， 其他的编译语言像 Pascal 、 Fortran 都没落了，函数式语言不够全能，脚本语言因性能和用户友好性无法担当重任。 因此，主程序的重担必须落在 C/C++ 上了，这是最主流的选择。

全用 C 语言虽然很美，但不现实，现在众多的 C++ 库， 还有 C++ 11 是那么的吸引人。 用 C 的话， 意味着要自己完成一切， 尤其是 GUI 界面上。

但是，全用 C++ ， 灵活性是不够的，主流程序都有脚本系统，主要是 Python （如 Maya） ， JavaScript（如 Adobe 的设计软件）。 Lua 的应用也十分广泛，但由于功能库太少，主要用于游戏之中。 Ruby 较少见。 因此本系统使用 Python 作为功能模块扩展系统，使用 JavaScript 作为用户操作自动化脚本系统。 主要借鉴 Autodesk Maya 和 Adobe InDesign。

考量2： 运行库的选择和绿色免安装

上次做水文测验程序，由于使用了 .NET Framework ，老电脑 Windows XP 系统安装比较麻烦，需要耗时20~30分钟，需要给系统升级SP3的话就更耗时间了。 有的电脑还装不上去，需要重装系统。

这次起初打算用兼容上世纪 C/C++ 链接到 msvcrt.dll，用 MFC 4.2 做图形和一些基础功能库。 但是，由于这样太偏离时代，所以就放弃了这个想法。

Python 很早就抛弃了 msvcrt.dll，也没有详细的资料证明可以用 MinGW 编译并链接到 msvcrt.dll 上， Python 3 默认编译器是比较新的 Visual C++ 2010， 也是跟 Windows

XP SP2 兼容比较好的最后一个版本。 因此，应该用这个版本。程序自然也只需要让用户装上 Microsoft Visual C++ 2010 Redistributable 即可。

但是，Visual C++ 2010 的编译器对 C++ 11 的支持度不够，已经落伍了。 Visual C++ 2012/2013 又最多只能跟 Windows XP SP3 兼容，因此，被排除考虑范围。

于是，编译的重任落在第三方工具——也是开放免费的——MinGW/GCC 4.8身上，Qt 5.1.1 的 MinGW 版就附带 4.8 版。 通过实验，还是比较靠谱的。因此本软件系统打算使用它作为主要的编译器。

MinGW 编译器默认链接到 msvcrt.dll 库，涉及到同一块内存访问时， 会与 Python 默认链接到的 msvcr100.dll 冲突。 因此，所有用 MinGW 编译的二进制文件需要调整为链接到 msvcr100.dll。

考量3： 跨平台图形库的选择

这次希望做一个跨平台可移植的程序，自然 MFC 是不予考虑了。 迅雷的 Bolt 库能做出很炫的界面来，但是也不是跨平台的。 文档还不是很充分，而且对于这个绘图程序来讲，更重要的是效率，而不是界面。

至于 wxWidgets 和 Gtk+ ，因为不熟悉，在 Windows 上见过的用它们做的程序也少（ Code::Blocks、 GIMP、 Inkscape ），所以这次暂时不考虑了。

Qt 4 以前在做嵌入式的时候用过，比较熟悉，但是也比较烦它，原因有：

1. 类库太大了：占用内存大，在只有 64M 内存的嵌入式开发板上运行缓慢。
2. 在当年我的老电脑上编译慢，并且因为类库的问题（触摸屏、framebuffer等）很难编译通过。
3. 开发低效，有一种跟 C# 差不多的感觉。

这次还是选择了 Qt ，并且还是还未大规模应用的 Qt 5 版本，原因有：

1. Qt 5 支持 C++ 11 的特性，说明 Qt 开发人员的与时俱进。
2. Qt 本来就跨 Windows 、 Linux 、 OS X 平台，也有 BSD 等 Unix 系统的移植方案。 Qt 5.1 发布了 Android 的库； Qt 5.2 发布了 iOS 的库，也再次看到开发人员的努力， 也为今后程序移植到这些吸引人的平台提供可能。
3. 使用的人很多，从众心理嘛……，Qt 原来主要运用在 Linux/KDE 下。现在一些主流的产品也使用 Qt 了，尤其是 Autodesk 公司的产品， 除了 AutoCAD （.NET/WPF）和 3ds Max （MFC） 之外，什么 Maya、Softimage、Mudbox、

MotionBuilder 等等都使用了 Qt 库。我常用并经常称赞的免费的 TortoiseHg。还有 Sibelius ，Evernote 出品的 Skitch 等等数不尽的主流大型应用程序都在用 Qt。

考量4： 编译自动化

项目初始阶段主要用 qmake 进行管理，后来发现 qmake 不足以满足完全自动化的需要，于是 cmake 重装上阵。

但是只有 cmake 是不够的，还得需要一些脚本来操纵编译过程。 一开始在 Windows 下开发，图省事用了 bat。 随着项目推进， bat 语法怪异，自动化工作越来越力不从心，因此新的语言提上日程。 看到[GitHub]上，BYVoid 用 OCaml 写了一个Batsh， 本来打算应用。 但是还是担心新项目的稳定性，尤其是最近一个月以来又没有更新，时间也不允许我去读它的源代码， 因此编译管理打算用 Ruby 。 为什么不用 Python 呢，主要还是从众……

持续集成领域， Ruby 圈子里已经有了很好的项目，所以，我这小小的编译自动化，当然也要超时代前沿看齐了。

考量5： 测试自动化

尽量能用上，但就我一个人开发，又牵扯三种语言，比较费劲。先推后。

考量6： 源代码管理

一定要用 DSCM，分布式的好处太多了。项目之初使用 Bitbucket/Mercurial(hg)私有仓库管理， 等到可以开源的时候，再用 GitHub/Git 管理。

考量7： 文档管理

这个就全部用最主流新的方式——用最流行的 markdown 格式写文档，并用源代码管理这些文档。 通过 pandoc 转换成 wiki 网页、 pdf 文档和 doc 文档。

pandoc 可以完成 markdown 格式到 html 的转换，用于 wiki 页，支持模板。 这个最好全自动，提交一次代码，就生成一次 wiki。转换成 html 很快，很灵活。便于通过 wiki 查阅文档。

pandoc 可以完成 markdown 格式到 pdf 的转换，支持 latex 模板。成为正式的文档。

pandoc 可以完成 markdown 格式到 doc 的转换，提供给单位的同事用于制作报审、验收、评职称的材料。

考量8： 互联网互操作

现在都讲云计算，是否应该开发 Web 应用呢。 SVG 已经作为本程序里组图功能以来的格式了。 如果能在互联网上传输，结合 html5， 通过浏览器来操作， 一定会很炫。但现在本局没有任何使用了解云计算的规划， 资料保密的考量，暂时没有这种需求，所以暂不考虑。

考量9： 文件格式

关于项目文件： XML 和 JSON 是主流的，比二进制文件操纵简单的格式。C++ 操作它们的类库多且先进。

配置文件尽可能的主流，无论是ini、cfg、XML、JSON，还是 sqlite 数据库加ORM，都可以。

项目文件应该是一种档案文件，打包许多原始文件，像整编原始数据、绘图的指令序列、生成的中间过程图（SVG）。

整编原始数据大多是文本文件，读写还算容易，就是容易出错，容易混乱，处理起来需要小心谨慎。

考量10： 发布管理

Windows 平台

Windows 平台上主要要解决系统安全性 UAC 设置后无法读写 Program Files 目录的问题。

1. 二进制文件：
    1. 主程序（HydroCurve.exe）。
    2. 依赖的类库（*.dll）。
    3. Python Lib库的py们编译成pyc、pyo后打压缩包 （pythonXX.zip）。
2. 脚本文件：

1. Python 脚本文件（位于程序目录的 modules 目录下）。
2. 全局 JavaScript 脚本文件 （位于程序目录的 scripts 目录下）。
3. 用户 JavaScript 脚本文件（位于系统 AppData\HydroCurve\scripts 目录下，必须提供打包导出功能才行）。

3. 配置文件：

1. 全局配置文件 （位于程序目录的 config 目录下）。
2. 用户配置文件（位于系统 AppData\HydroCurve\config 目录下， 必须提供打包导出功能才行）。

(本页为有意留白)

Chapter 3

主程序设计

设计的场景：用户看着屏幕，通过键盘鼠标进行操作。

考虑程序尽可能的容易上手，键盘主要用于录入数据，控制主要用鼠标。 因而并不需要设计复杂的快捷键控制逻辑。 现在没有应用触摸屏和绘图板，所以操作上要求尽量的符合鼠标操作的特点。

界面设计

界面设计尽可能的简洁。

由于程序大范围的区域需要显示图纸，因此功能模块主要放到菜单中，工具条尽可能的少。

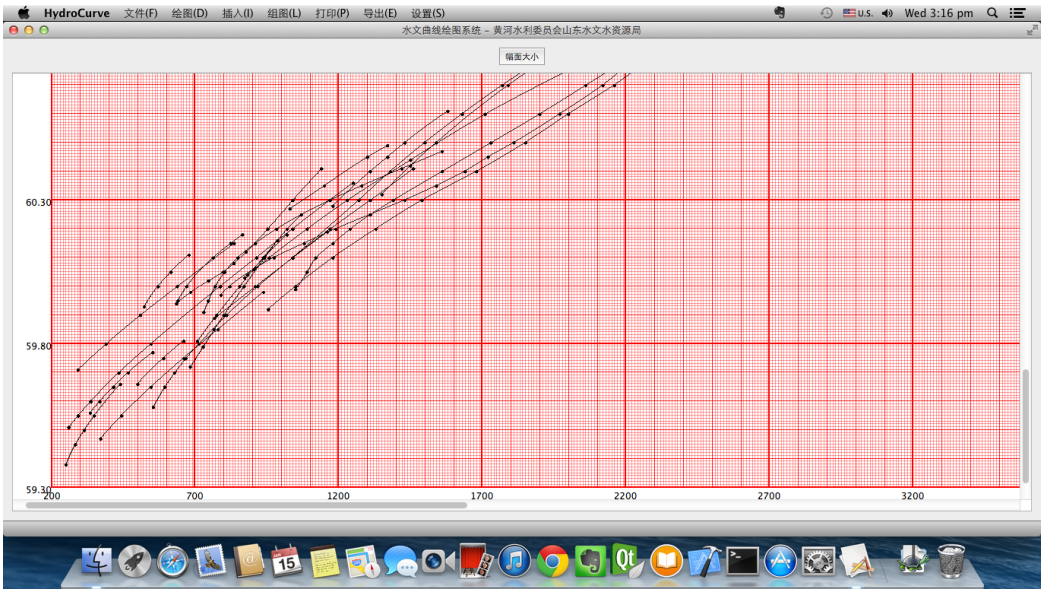

Figure 3.1: OS X 系统上主界面截图 （分辨率1366x768）

Chapter 4

附录

开发环境配置

为了方便他人增进本程序功能，特专门写一下如何配置开发环境。

开发时主要用的是 Windows 7 SP1 64bit 和 Ubuntu 13.04、13.10。分开说。

Windows 平台

一定要用 Windows 7 吗？一定。

Windows 7 SP1 是呈上启下的一个成熟的系统， 结合 Visual Studio 2010 SP1 开发的程序对于早前的 Windows XP 和后继的 Windows 8 有着非常好的兼容性。 建议用 64 位，因为未来很快就是 64 位的天下。

当然编译本程序 Windows XP/Vista/7/8 都可以，只是不那么方便灵活罢了。

一台装好的 Windows 7 ，成为开发机，需要经过以下步骤的折腾。

开发目标为 32 位，能在 Windows XP 以上系统运行的程序

1. 安装 Microsoft Visual C++ Express 2010 SP1。
2. 安装 Windows SDK 7.1。
3. 安装 Python 3.3.3 32位 到 C:\Python33 下。
4. 安装 Ruby 2.0 32或64位 到 C:\Ruby200(x64)下。
5. 安装 Git for Windows。

6. 安装 TortoiseHg。
7. 安装 Qt 5.2.0 for MinGW。
8. 安装 cmake 2.8.12。
9. 使用 Git 克隆 Ninja，并用 VC2010 编译 Ninja 源代码。
10. 下载或克隆 boost 1.55 库。
11. 配置环境变量，详见下节。

开发目标为 64 位，能在 Windows 7 64bit 以上系统运行的程序

1. 安装 Microsoft Visual Studio 2012 Express for Desktop
2. 安装 Windows SDK 8。
3. 安装 Python 3.3.3 64位 到 C:\Python33 下。
4. 安装 Ruby 2.0 64位 到 C:\Ruby200x64 下。
5. 安装 Git for Windows。
6. 安装 TortoiseHg。
7. 安装 Qt 5.2.0 for MSVC 2012 (OpenGL版即可)。
8. 安装 cmake 2.8.12。
9. 使用 Git 克隆 Ninja，并用 VC2010 编译 Ninja 源代码。
10. 下载或克隆 boost 1.55 库。
11. 配置环境变量，详见下节。

Ubuntu/Linux

Unix/Mac/Linux 跟 Windows 不同，我们永远只用为最新或比较新的系统考虑，不像 Windows 存在古董 XP 系统这样的问题。因此环境配置非常简单，以 Ubuntu 13.04 以上系统为例。命令前 sudo 或 sudo -i 随便。

1. apt-get install build-essential

2. apt-get install git
3. apt-get install tortoisehg
4. apt-get install python3.3
5. apt-get install ruby2.0
6. apt-get install cmake
7. apt-get install ninja
8. 下载或克隆 boost 1.55 库。
9. 安装 Qt 5.2.0 for Linux x86/x64。

文档环境配置

文档环境主要由字体、pandoc和Latex软件组成。

为了让 Windows 和 Linux、Mac 下生成效果一致，使用免费开源的文泉驿字体。 主要有两种印刷字体，都是黑体。 正黑涵盖字数多，微米黑少些，但也够覆盖GB18XXX。正黑典雅复杂，微米黑现代简约。 设计上，正文、代码、小字用正黑，标题、大字用微米黑。

Windows

1. 下载“文泉驿正黑”和“文泉驿微米黑”的 ttf 字体并安装。
2. 下载并安装 pandoc。
3. 下载并安装 MikTex。
4. 如果没有搭建开发环境的话，需要安装 Python 3 和 Ruby 2。

Ubuntu/Linux

这里还是以 Ubuntu 13.04 为例。

1. apt-get install ttf-wqy-zenhei ttf-wqy-microhei
2. 安装 pandoc。

3. 安装 texlive。
4. 如果没有搭建开发环境的话，需要安装 Python 3 和 Ruby 2。