

北京邮电大学

本科毕业设计（论文）



题目：基于三维触摸屏的素描板开发

姓 名 刘诗韵

学 院 软件学院

专 业 软件工程

班 级 07A02

学 号 072680

班内序号 34

指导教师 郭文明

2011 年 6 月

基于三维触摸屏的素描板开发

摘 要

时至今日，计算机的发展如火如荼。起初，计算机机如其名，是用来方便人类计算的机器。在日常生活中，总是充满了庞大的信息量和复杂的处理关系，计算机正是以这样一种方便快捷的服务者的身份进入时代。面对数据的繁复，运算的冗杂，计算机都可以做到得心应手，而且计算结果精准，是人类处理信息的强大帮手。而今，作为计算机创造者的人类，又开始了新的探索和寻觅。人机交互的研究又是一个神奇的新领域，人类可以通过与计算机各种接触的方式中，体会出不同的感觉和乐趣。在本次毕业设计中，我的任务就是基于三维触摸屏进行素描板软件的开发。运用 MFC 编程技术和触摸屏辅助代码完成实验。

本文即是针对于这次毕业设计的内容进行介绍。触摸屏素描板是提供用户进行素描绘画的工具。用户可以选择画笔的粗细，设置力度与深度的映射，通过指尖在屏幕界面绘画区域的按压，从而显示出素描的笔迹。

本文介绍了关于该课题的背景工具、需求分析、设计分析、实现代码以及在研究过程中所遇到的问题和解决的方法。在该系统中，设置了许多便捷的工具栏，如画图栏提供了九支不同宽度的快捷铅笔，绘画栏提供了自定义画笔及橡皮等等。并且，用户可以打开和保存 bmp 格式的文件，实现了应用程序与外界的联系和交互。另外，在性能方面，也做了考虑，譬如运用双缓冲技术存储图片信息至内存，譬如放弃画像素点而选择画线的方式。

目前为止，已经实现了基于三维触摸屏的素描板的基本功能。并且还进行了一小部分地界面美化。在对该软件的测试后，也可以确定在内存空间够用的情况下，其能稳定运行。该系统可以作为触摸屏领域研究的范例。

关键词 人机交互 MFC 触摸屏 素描板 双缓冲

Design and Implementation of Ontology-Based Network Management System

ABSTRACT

Today, the development of the computer in full swing. At first, the computer machine, as its name, is a tool which used to facilitate computing of human's life. In daily life, it is always full of massive amount of information and complex processing relations, and the computer goes into the era in such a convenient and efficient service identity. When face to complex data and jumbled computing, the computer can be handy and the results are also accurate, so it is a powerful helper for mankind to process information. Now, the human who is the creator of computers, begin to make a new exploration and searching. The research of Human Computer Interaction (HCI) is a new wonderful field. Though various ways of contacting with the computers, the humans can get a different feeling and pleasure. In this graduation project, my task is developing a drawing board software based on three-dimensional touch screen technology. MFC programming technology and touch screen supporting code is used to complete this experiment.

This article is designed to introduce the content of this graduation project. Drawing board which based on touch screen is to provide users a tool for drawing sketches. The user can choose the thickness of the brush, set the mapping relationship between intensity and depth by using finger to press the painting region on the screen, thus the sketch's handwriting is showed.

This article describes the background and tools of this subject, requirements analysis, design analysis, implementation code the problems and solutions in this course of the study. In this system, there are a number of convenient toolbars, such as pen bar provides nine quick pencils with different widths, drawing bar provides a custom brush and rubber and so on. Also, users can open and save picture files of bmp formats, and it will achieve the contact and interaction between application and outside world. In addition, in terms of performance, I also make a consideration. For example, the use of double buffering technique to store the picture information into memory, and choice of using lines to paint instead of pixel.

So far, I have implemented the basic functions of drawing board software based on three-dimensional touch screen technology. And also beautify a part of the dialog interface. After testing the software, it can also determine that the software will run steadily if the memory space is enough. The system can be used as an example of the research in the touch screen field.

KEY WORDS HCI, MFC, touch screen, drawing board, double buffering technique

目 录

第一章	引言	1
1.1	背景和意义	1
1.2	目的和任务	2
1.3	内容简介	2
1.4	文章结构	2
第二章	相关技术背景	4
2.1	VC6.0 下的 MFC 编程	4
2.2	触摸屏编程	4
2.3	本章小结	5
第三章	需求分析	6
3.1	用户分析	6
3.2	使用场景	6
3.3	功能性需求分析	7
3.3.1	新建文件	7
3.3.2	打开文件	7
3.3.3	编辑文件	8
3.3.4	保存文件	9
3.4	非功能性需求分析	9
3.5	本章小结	10
第四章	软件设计	11
4.1	体系结构	11
4.2	功能模块	12
4.2.1	新建文件	12
4.2.2	打开文件	12
4.2.3	编辑文件	13
4.2.4	保存文件	15
4.3	数据结构	16
4.3.1	画笔	16
4.3.2	触点	16
4.3.3	图片	16
4.4	本章小结	18
第五章	程序实现	19
5.1	开发环境	19

5.2 关键代码	19
5.2.1 触摸屏代码	19
5.2.2 工具栏代码	20
5.2.3 对话框代码	21
5.2.4 画笔栏代码	23
5.2.5 绘图栏代码	23
5.2.6 新建文件代码	24
5.2.7 编辑文件代码	25
5.2.8 打开文件代码	30
5.2.9 保存文件代码	31
5.2.10 全局变量	35
5.2.11 定时器	35
5.3 本章小结	37
第六章 程序测试	38
6.1 对程序进行单元测试	38
6.1.1 程序主窗体	38
6.1.2 菜单栏	39
6.1.3 工具栏	40
6.1.4 画笔栏	41
6.1.5 绘图栏	42
6.1.6 打开文件对话框	43
6.1.7 保存文件对话框	43
6.1.8 设置映射参数的对话框	43
6.1.9 设置画笔粗细的对话框	44
6.2 对程序进行整合测试	45
6.2.1 绘画成果	45
6.2.2 保存结果	46
6.3 本章小结	46
第七章 遇到的问题 and 解决方法	46
7.1 问题及解决方案	47
7.2 本章小结	48
第八章 结论	49
8.1 研究结论	49
8.2 进一步工作	49
参考文献	50

致 谢.....	51
附 录.....	52
附录 1 计划与执行	52

第一章 引言

本章主要对这次毕业设计的课题做出一个绪论，起到引出的作用。针对触摸屏素描板，讨论了这种技术的背景和意义，目的和任务，以及对这次课题的内容做出一个内容的阐述，并给出本论文的篇章结构，

1.1 背景和意义

而今，计算机技术纵横捭阖，短短数十载，已势如破竹。探其渊源，从 17 世纪巨型的自动进位加法器，到 18 世纪的乘法器，再到如今的电子计算机，这段历史正见证计算机的发展，凝结着探索者的不懈，更体现着人类的智慧。

计算机，顾名思义，是着重于计算的机器，其最主要的功能为处理数据。在日常生活中，需要历经许多人事，庞大的数据量确实让人错愕，人工手动难免过于繁复，所耗时间亦漫长久远。并且人是不定性元素，本身就会有所失误，在面对冗杂的事物时，心躁难耐，更不易精准，也许劳苦多日，结果却并不乐观和有用。计算机，一个可以高效正确处理大量数据的工具，一个为人造却在某些方面远甚于人的机器，将在信息社会中奠定举足轻重的地位。

在平日里，大多数计算机仍以鼠标作为操控界面的工具。或手指轻按，或莲步微移，界面的指针同步执行，完成了一桩又一桩心事。但是，计算机并不满足于仅仅处理数据，又开始涉足更宽广的领域。

计算机，探其根本，旨在为人类服务。我们有大量的数据需要处理，所以计算机帮我们快速精确地处理；我们需要收集大量的信息，所以计算机为我们提供大量的信息和便捷的搜索；我们需要在繁琐的工作之余获得乐趣，所以计算机又被开发成游戏平台，供我们娱乐。我们对计算机的需求标准已日渐上升，早就不是惊叹于他们处理数据的完美，从而沾沾自喜了。我们希望他们富有亲和力，富有亲切感，富有舒适度，富有美感，在我们与他们的交流中，得到欢乐。所以另一门学问诞生了——人机交互。

人机交互是一门研究系统与用户之间的交互关系的学问。系统可以是各种各样的机器，也可以是计算机化的系统和软件。人机交互界面通常是指用户可见的部分。用户通过人机交互界面与系统交流，并进行操作。而人类世界千姿百态，不同的人对于计算机的理解和需求也不一样，不同的场景对于计算机的需求也大相径庭。对于老年人以及一些特殊人群，需要更方便的人机交互程序和设备；对于像银行这样面向对象很广的服务机构，也需要尽量满足各种人群的机器，如触摸屏。

触摸屏又称为触控面板，是个可接收触头等输入讯号的感应式液晶显示装置，当接触了屏幕上的图形按钮时，屏幕上的触觉反馈系统可根据预先编程的程式驱动各种连结装置，可用以取代机械式的按钮面板，并借由液晶显示画面制造出生动的影音效果。触摸屏作为一种最新的电脑输入设备，它是目前最简单、方便、自然的一种人机交互方式。它赋予了多媒体以崭新的面貌，是极富吸引力的全新多媒体交互设备。触摸屏在我国的应用范围非常广阔，主要是公共信息的查询。

由此观之，计算机已展现出各种各样的功能，人们也正着手于挖掘计算机的潜力。

而基于人机交互研究的触摸屏技术，也逐渐崭露头角。基于触摸屏的软件开发，也成为炙手可热的工作。

本文将针对基于三维触摸屏技术的素描板开发进行一番探讨与介绍。

1.2 目的和任务

目的：

本课题来自于实际科研课题，本次毕业设计目的是利用三维触摸屏设计并实现一个电子素描板。该电子素描板可以捕获和记录触摸点的(x,y,z)，经过软件处理，将画笔的动作记录下来，显示成一幅图，从而达到在三维触摸屏下可以完成素描创作的功能。

任务：

- (1) 熟悉三维触摸屏接口程序；
- (2) 分析素描板的需求，设计电子素描版主要功能（用力大，笔画深，用力小，笔画浅）；
- (3) 调用三维触摸屏接口程序，编程实现素描板，并具备复制、粘贴、保存功能，支持画者在该素描板下完成创作；

1.3 内容简介

触摸屏素描板，是一个供用户进行素描绘画的平台。它基于触摸屏计算机，绘画时，用户以手代笔。

该素描板软件仿照真实美术中的素描绘画，只提供黑白的色调，深浅的笔迹，粗细的笔尖。用户可用不同的线条勾勒出所见，所思。

使用该素描板时，画笔宽度可以手动设置，设置后为固定值。绘画时，根据用户用力大小显示出笔迹颜色的深浅，用力大则颜色深，用力小则颜色浅。

1.4 文章结构

本文共分为八章，篇章结构如下：

第一章：主要介绍本次毕业设计的课题研究背景、任务目的、开发环境以及这篇毕业论文的篇章结构。

第二章：主要介绍与该课题相关的技术支持。

第三章：主要进行该触摸屏素描板的需求分析，包括用户，应用场景，功能用例分析以及非功能分析。

第四章：主要进行该触摸屏素描板的设计分析，包括系统的体系结构，模块划分，各模块的分析以及数据结构的简要介绍。

第五章：主要介绍该触摸屏素描板的实现，包括软件实现和应用需要的软件和硬件环境，以及重要实现代码的分析。

第六章：主要介绍该触摸屏素描板的测试，是一些界面和截图的介绍。包括

单元测试和整合测试。

第七章：主要介绍了在进行这次毕业设计的过程中遇到的问题以及解决的方案。

第八章：对整篇论文做出总结，并简要介绍下一步的计划以及可以再更加完善的地方。

第二章 相关技术背景

本章主要介绍这次毕业设计课题的相关技术背景，作为一次试验的蓄势铺垫，也是一种准备工作，技术背景更是软件开发的基础。本章阐述了 VC++6.0 下的 MFC 界面编程以及触摸屏编程的概括性知识。

2.1 VC6.0 下的 MFC 编程

MFC (Microsoft Foundation Class Library) 中的各种类结合起来构成了一个应用程序框架，它的目的就是让程序员在此基础上建立 Windows 下的应用程序，这是一种相对 SDK 来说更为简单的方法。因为总体上，MFC 框架定义了应用程序的轮廓，并提供了用户接口的标准实现方法，程序员所要做的是通过预定义的接口把具体应用程序特有的东西填入这个轮廓。Microsoft Visual C++ 提供了相应的工具来完成这个工作：AppWizard 可以用来生成初步的框架文件（代码和资源等）；资源编辑器用于帮助直观地设计用户接口；ClassWizard 用来协助添加代码到框架文件；最后，编译，则通过类库实现了应用程序特定的逻辑。

MFC 实现了对应用程序概念的封装，把类、类的继承、动态约束、类的关系和相互作用等封装起来。这样封装的结果对程序员来说，是一套开发模板（或者说模式）。针对不同的应用和目的，程序员采用不同的模板。例如，SDI 应用程序的模板，MDI 应用程序的模板，规则 DLL 应用程序的模板，扩展 DLL 应用程序的模板，OLE/ACTIVEX 应用程序的模板，等等。

这些模板都采用了以文档-视图为中心的思想，每一个模板都包含一组特定的类。典型的 MDI 应用程序的构成将在下一节具体讨论。

为了支持对应用程序概念的封装，MFC 内部必须作大量的工作。例如，为了实现消息映射机制，MFC 编程框架必须首先要保证首先得到消息，然后按既定的方法进行处理。又如，为了实现对 DLL 编程的支持和多线程编程的支持，MFC 内部使用了特别的处理方法，使用模块状态、线程状态等来管理一些重要信息。虽然，这些内部处理对程序员来说是透明的，但是，懂得和理解 MFC 内部机制有助于写出功能灵活而强大的程序。总之，MFC 封装了 Win32 API，OLE API，ODBC API 等底层函数的功能，并提供更高层的接口，简化了 Windows 编程。同时，MFC 支持对底层 API 的直接调用。

MFC 提供了一个 Windows 应用程序开发模式，对程序的控制主要是由 MFC 框架完成的，而且 MFC 也完成了大部分的功能，预定义或实现了许多事件和消息处理，等等。框架或者由其本身处理事件，不依赖程序员的代码；或者调用程序员的代码来处理应用程序特定的事件。

MFC 是 C++ 类库，程序员就是通过使用、继承和扩展适当的类来实现特定的目的。例如，继承时，应用程序特定的事件由程序员的派生类来处理，不感兴趣的由基类处理。实现这种功能的基础是 C++ 对继承的支持，对虚拟函数的支持，以及 MFC 实现的消息映射机制。

2.2 触摸屏编程

介绍：

GeneralTouch API 提供了与 GeneralTouch 串行和 USB 驱动程序，版本是 GTDrv4.2.1.73I 或更晚的版本。以下功能是可用于编程。在使用这些功能之前，你必须

使用一些窗口 API 来获得我们的设备句柄，如 **CreateFile** 函数。我们的设备的路径名是“\\\\.\\GENTOUCH”以及 COM 端口。

使用时

1. 把动态链接库 **GenTouchDll.dll** 放到你所建立的工程目录下；
2. 参照 API 文档里的说明调用这个动态链接库的函数；
3. 得到的数据是 32 为 **ULONG** 数据，由于驱动把 Z 坐标放在高 16 位，所以把得到的原始数据右移 16 位就可得到 0 到 255 间的 Z 数据；
4. 这里写了一个 **MyCalibrationApp** 的 VC 工程，可以参照里面的代码进行函数调用和 Z 轴数据转换，着重看 **MyCalibrationAppDlg.cpp** 里面的代码

2.3 本章小结

本章主要介绍了与本次毕业设计课题（基于三维触摸屏的素描板）开发相关的技术。介绍了基于 VC++6.0 编译器的 MFC 界面编程技术，以及触摸屏相关介绍，该触摸屏相关介绍来自于 **GeneralTouch** 的生产商提供的说明文档。

第三章 需求分析

本章主要是对基于三维触摸屏的素描板进行初始的需求分析。确定该软件的用户以及使用场景，从而体现软件的价值。还对该软件进行了功能性需求分析以及非功能性需求分析，使大家对该软件有个宏观上的了解。

3.1 用户分析

目标用户：

美术学校，建筑公司

最终用户：

美术学校，建筑公司，素描爱好者，其他有需求的公司。

3.2 使用场景

使用场景是该触摸屏素描板软件可以应用的领域分析，从而体现了软件的应用范围和自身的价值。

自娱自乐

- 1、用户或素描爱好者，可通过该程序实现电脑作画。
- 2、进行绘画时，可以调整画笔宽度，以及用力度控制颜色深浅。
- 3、会话结束后可方便地将绘画保存至电脑并发送。

课堂教学：

- 1、教师可以用该程序展示作画技巧。
- 2、运用力度控制画笔，从而模仿真实场景中的素描绘画，展示教学内容并进行示范。
- 3、教学结束后，易于将教学成果保留给学生。

草图绘制：

- 1、建筑公司或有需求的公司进行草稿设计时，可用该程序。
- 2、大致绘图，力度可用来辅助，不是必须。
- 3、绘图结束后，传阅方便，修改方便。

敏捷建模：

- 1、在面向对象分析中，小组讨论时，可用该画板进行敏捷建模
- 2、使用便捷，迅速。力度为辅，不是必须。
- 3、建模结束后，方便保存。

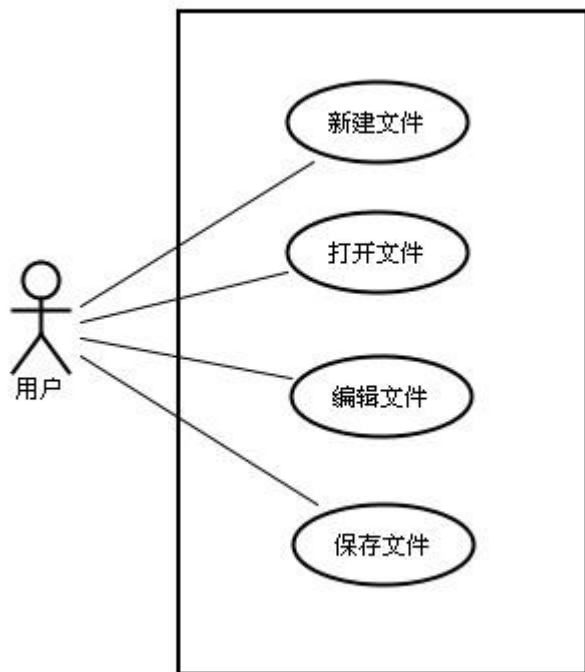
会议商讨：

- 1、企业开会中，也可使用该素描板。
- 2、会议草稿，记录文案。
- 3、便于记录会议中的临时提议，并易于整理。

3.3 功能性需求分析

该素描板程序为单机软件，且无数据库等复杂应用，因而用户只有一种类型，即使用该软件的使用者。

该素描板共有四个功能用例：新建文件，打开文件，编辑文件，保存文件。



3.3.1 新建文件

表 3-1 触摸屏素描板的新建文件用例

用户	使用者
概述	使用者可以新建一个绘图文件。
前置条件	使用者打开触摸屏素描板软件。 或已经进行绘画。
细节描述	使用者点选菜单中的“文件”->“新建” 或者点击工具栏的“新建”按钮。
后置条件	使用者会看到绘画区域清屏，重新变为空白画板。

3.3.2 打开文件

表 3-2 触摸屏素描板的打开文件用例

用户	使用者
概述	使用者可以打开一个绘图文件。

前置条件	使用者点选菜单中的“文件”->“打开” 或者点击工具栏的“打开”按钮。 成功打开了打开文件的对话框。
细节描述	使用者选择了需要打开的图片对象。 然后点击确认按钮。
后置条件	使用者会看到绘画区域内显示出了图片。

3.3.3 编辑文件

在该程序的编辑文件功能用例中，分为如下三个子用例：绘图用例，映射参数设置用例和画笔粗细设置用例

绘图用例：

表 3-3 触摸屏素描板的绘图用例

用户	使用者
概述	使用者可以进行绘画或编辑一个绘图文件。
前置条件	使用者打开触摸屏素描板软件， 或者新建了一个绘图文件， 或者打开了一个绘图文件。
细节描述	使用者触摸屏幕中的绘画区域进行绘画。 或在画笔栏或者绘画栏中选取需要的画图工具进行绘画。
后置条件	使用者会看到绘画区域出现所绘的图或者所打开的图片文件。

映射参数设置：

表 3-4 触摸屏素描板的映射参数设置用例

用户	使用者
概述	使用者可以设置力度到深度的映射关系。
前置条件	使用者点选菜单栏中“设置”->“设置映射参数”， 成功打开了映射参数设置对话框。
细节描述	使用者可以修改参数：最大力，最小力，最深色，最浅色。 或点击还原默认值按钮。 最后点击确认按钮完成修改，或点击取消按钮退出。
后置条件	使用者设置了力度到深度的映射关系， 从而将在绘画板上的线条中体现。

画笔粗细设置：

表 3-5 触摸屏素描板的画笔粗细设置用例

用户	使用者
概述	使用者可以设置画笔笔尖的宽度。
前置条件	使用者点击绘图栏中“削粗铅笔”按钮 成功打开了画笔粗细设置对话框。
细节描述	使用者可以通过滑块修改画笔笔尖的宽度， 并且在编辑框中显示出来。
后置条件	使用者会设置了画笔笔尖的宽度， 从而将在绘画板上的线条中体现。

3.3.4 保存文件

表 3-3 触摸屏素描板的保存文件用例

用户	使用者
概述	使用者可以打开一个绘图文件。
前置条件	使用者点选菜单中的“文件”->“保存” 或者点击工具栏的“保存”按钮。 成功打开了保存文件的对话框。
细节描述	使用者选择了需要保存的路径以及输入了需要保存为的图片名称。 然后点击确认按钮。
后置条件	使用者会看到图片成功输出。

3.4 非功能性需求分析

非功能分析是对系统性能和质量的考究，下面就该触摸屏素描板程序，进行如下七项非功能分析。

（1）可靠性

可靠性需要考虑的一些具体方面是：

- **安全性：**该触摸屏素描板程序为单机程序，无须联网，因此安全性比较高。
- **事务性：**该触摸屏素描板程序为计算机应用程序，无数据库，因此没有事务性顾虑。

（2）可用性

- 打开就可使用，无须其他支持，但须在此种触摸屏下使用。
- 单机用户只在一个界面上操作，没有多界面。
- 界面功能均有状态。

（3）有效性

- **性能：**该触摸屏素描板目前测试均可平稳运行。
并且运用了双缓冲技术绘图。双缓冲即在内存中创建一个与屏幕绘图区域一致的对象，先将图形绘制到内存中的这个对象上，再一次性将这个对象上的图形拷贝到屏幕上，这样能大大加快绘图的速度。双缓冲实现过程如下：
 - 1、在内存中创建与画布一致的缓冲区
 - 2、在缓冲区画图
 - 3、将缓冲区位图拷贝到当前画布上
 - 4、释放内存缓冲区
 - **可伸缩性：**此绘图程序不要求过快的运行速度，只要合适就行，达到要求。
可以在这个基础上进行其他的功能的扩展。
当内存空间不足时，绘画的双缓冲会受到一定影响。
- (4) **可维护性**
该触摸屏素描板功能较为简洁，可维护性体现得不明显。
- (5) **可移植性**
开发基于 GeneralTouch serial 以及 USB driver（版本是 GTDrv4.2.1.73I 或之后）的触摸屏以及 VC++6.0 编译器。
使用只要基于 GeneralTouch serial 以及 USB driver（版本是 GTDrv4.2.1.73I 或之后）触摸屏即可。
- (6) **完整性**
能够完成业务需求和系统正常运行本身要求而必须具有的功能。
- (7) **界面的美观性和人性化**
主应用程序界面简洁明了，工具栏设置易于操作。对话框的背景有图片支持，亦为黑白色调，符合软件主题。

3.5 本章小结

本章主要介绍了关于触摸屏素描板开发的需求分析。探讨了该软件的潜在用户和应用场景，并进行了功能分析及非功能分析。该程序总共有四个主要功能：新建文件，打开文件，编辑文件和保存文件。其中编辑文件又可分为绘图，笔画粗细设置，参数映射设置三个子用例。该程序的非功能分析从可靠性，可用性，有效性，可维护性，可移植性，完整性和界面的美观性和人性化这几个方面来阐述。

第四章 软件设计

本章主要阐述了该触摸屏素描板软件的设计方案。先介绍了该软件应有的体系结构以及各部分间的合作和交互。然后将其划分为功能模块，针对每一个功能模块，做出简要的分析。最后对部分重要的数据结构做出规划。

4.1 体系结构

基于三维触摸屏技术的素描板是一个单机软件的开发，不需要计算机与计算机之间的通信，没有网络关联。所以该软件的体系结构为界面与内存之间的交互，以及对图片的导出导入和触摸屏相关接口的调用。

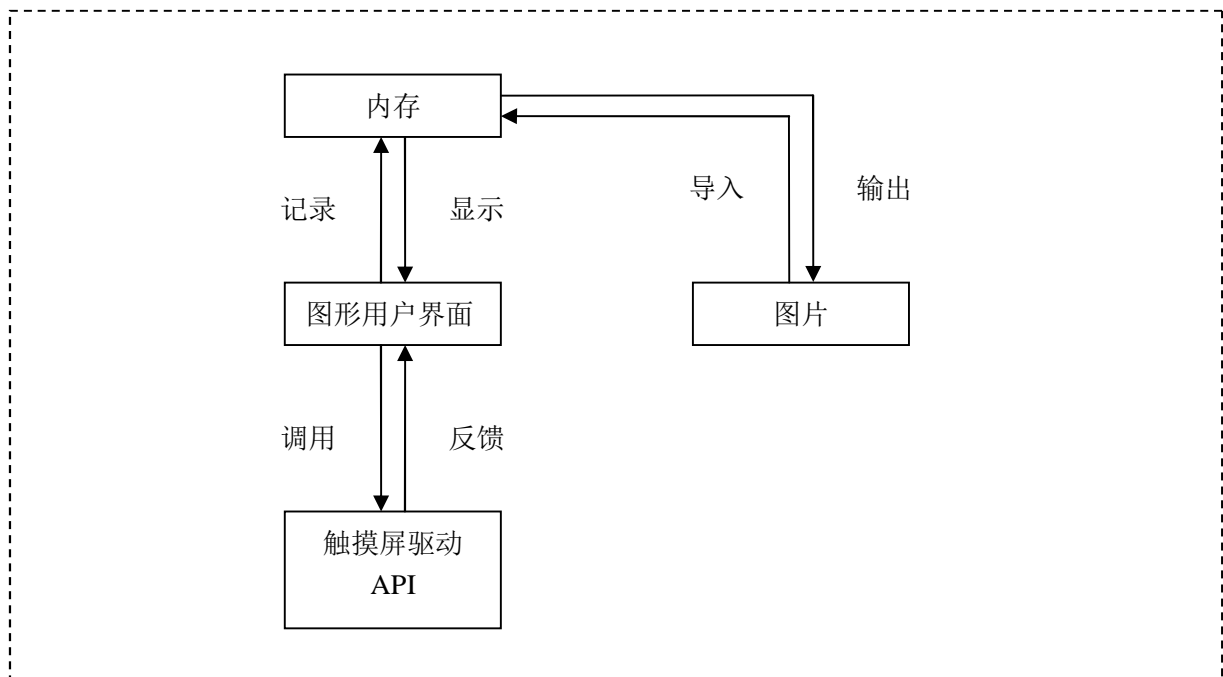


图 4-1 触摸屏素描板软件的体系结构图

1、图形用户界面：

图形用户界面是素描板软件与用户的界面，包括菜单，工具栏以及绘画区域。

2、内存：

内存是计算机的内部存储空间，对于该素描板，内存用来实时存储用户会话区域内容的，以备重新显示或者输出。

3、触摸屏驱动 API：

触摸屏驱动 API 是该素描板程序所基于的触摸屏的接口，该接口又由触摸屏开发商提供，软件的制作需要调用。

4、图片：

图片是素描板与外界交互的另一种形式，图片可以在素描板中打开，可以将素描板的内容保存。

4.2 功能模块

在程序设计中，为完成某一功能所需的一段程序或子程序；或指能由编译程序、装配程序等处理的独立程序单位；或指大型软件系统的一部分。本基于三维触摸屏的素描板程序就以其功能划分为四个模块：新建文件，打开文件，编辑文件和保存文件。

4.2.1 新建文件

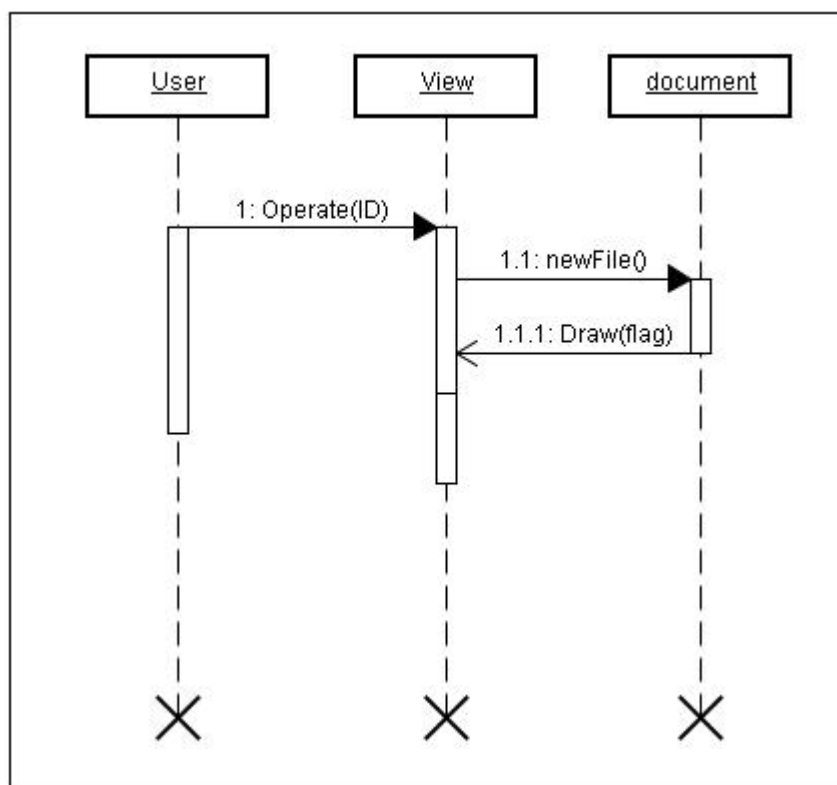


图 4-2 触摸屏素描板新建文件功能序列图

- 1、用户在界面上进行新建文件操作。
- 1.1、界面调用后台程序中的新建函数。
- 1.1.1、后台程序执行新建函数后，将结果返回，并调用前台界面的显示函数显示出来。

4.2.2 打开文件

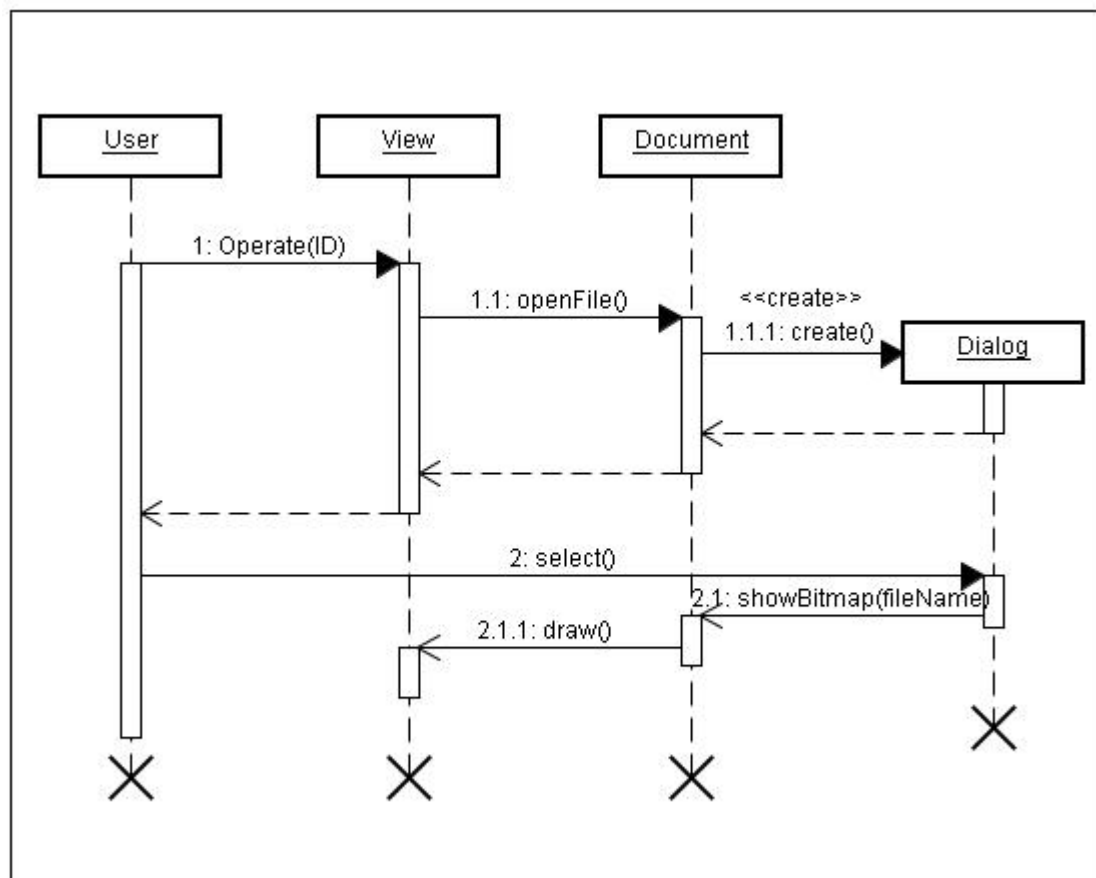


图 4-3 触摸屏素描板打开文件功能序列图

1、用户在界面中进行打开文件的操作。

1.1、界面调用后台程序的打开文件函数。

1.1.1、后台程序运行，并创建打开文件对话框。对话框显示。

2、用户在对话框中选择需要打开的文件。

2.1、对话框将打开文件的文件路径返回，并调用后台的图片复制流函数，将所选图片信息按照格式加载至内存。

2.2.1、最后界面的显示函数被调用，将内存中的图片信息显示到界面上的绘图区域。

4.2.3 编辑文件

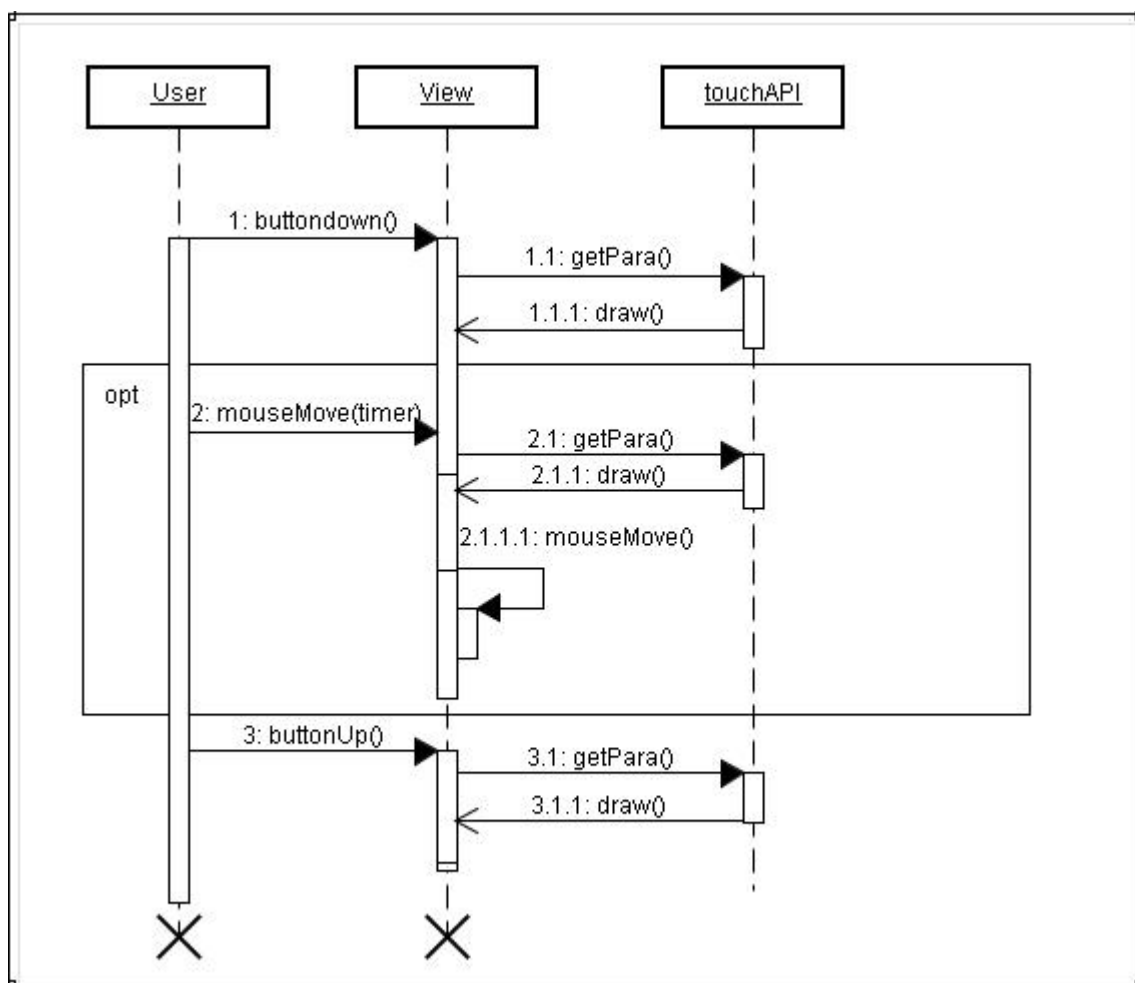


图 4-4 触摸屏素描板编辑文件功能序列图

- 1、用户在绘画区进行绘画时，先在区域中进行按压，从而调用界面的按压函数。
 - 1.1、界面的按压函数需要调用触摸屏提供的接口，并获取按压点数据结构中的值。
 - 1.1.1、触摸屏接口执行完毕，返回结果，并调用界面显示函数，在按压处显示绘画点的信息。
- 2、若用户按压后未曾抬起，继续在绘画区进行按压式移动，则前台界面的移动函数将被调用。（可选）
 - 2.1、界面的移动函数同样需要调用触摸屏提供的接口，并获取按压移动点数据结构中的值。
 - 2.1.1、触摸屏接口执行完毕，返回结果，并调用界面显示函数，在按压移动的地方显示绘画的信息。
 - 2.1.1.1、 这个过程一直持续到用户抬起手指。
- 3、当移动结束后，或按压后，用户抬手，完成这一绘画过程，则启动界面的抬起

函数。

3.1.、界面的移动函数再次调用触摸屏提供的接口，并获抬起点数据结构中的值。

3.1.1、触摸屏接口执行完毕，返回结果，并调用界面显示函数，在抬起移动的地方显示绘画的信息。

4.2.4 保存文件

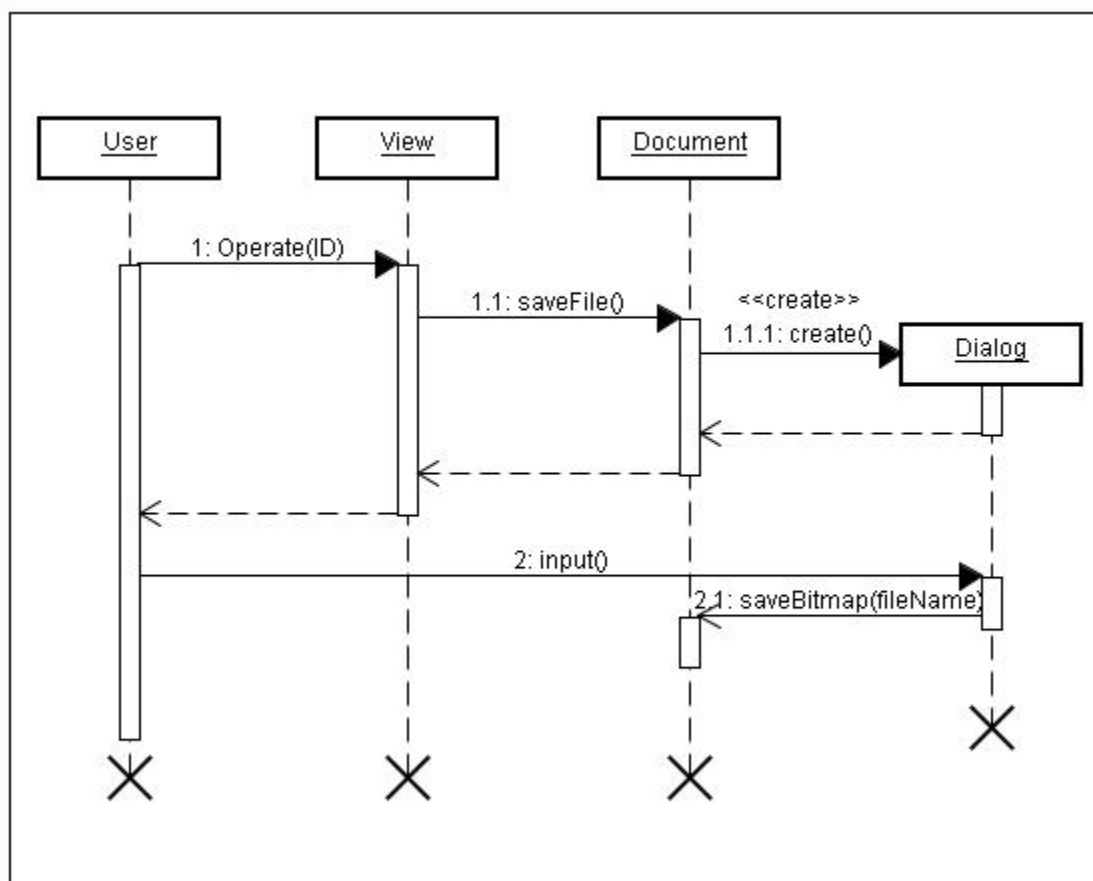


图 4-5 触摸屏素描板保存文件功能序列图

1、用户在界面中进行保存函数。

1.1、 界面调用后台保存文件函数。

1.1.1、后台程序运行，并创建保存文件对话框，对话框显示。

2、用户在对话框中输入要保存的文件名。

2.1、 对话框将保存文件的文件路径返回，并调用后台的图片复制流函数，将所选图片信息按照格式保存至图片中。

4.3 数据结构

4.3.1 画笔

画笔可通过 `CPen::CreatePen(int nPenStyle, int nWidth, COLORREF crColor)` 来创建。

数据结构 `pen` 中包含三个变量：

`int nPenStyle` 设置了画笔笔迹的风格。

`int nWidth` 设置了画笔笔迹的宽度。

`COLORREF crColor` 设置了画笔笔迹的颜色。

其中 `nPenStyle` 指名画笔的风格，可取如下值：

- **PS_SOLID** 实线 Creates a solid pen.
- **PS_DASH** 虚线，宽度必须为一 Creates a dashed pen. Valid only when the pen width is 1 or less, in device units.
- **PS_DOT** 点线，宽度必须为一 Creates a dotted pen. Valid only when the pen width is 1 or less, in device units.
- **PS_DASHDOT** 点划线，宽度必须为一 Creates a pen with alternating dashes and dots. Valid only when the pen width is 1 or less, in device units.
- **PS_DASHDOTDOT** 双点划线，宽度必须为一 Creates a pen with alternating dashes and double dots. Valid only when the pen width is 1 or less, in device units.
- **PS_NULL** 空线，使用时什么也不会产生 Creates a null pen.
- **PS_ENDCAP_ROUND** 结束处为圆形 End caps are round.
- **PS_ENDCAP_SQUARE** 结束处为方形 End caps are square.

4.3.2 触点

触点是用户在屏幕绘画区域进行绘画时与屏幕进行接触的点。

数据结构 `point` 中包含三个变量：

`x` 表示当前触点的设备横坐标的值。

`y` 表示当前触点的设备纵坐标的值。

`z` 表示当前触点的力度值。

（其中，坐标原点对应视左上角。）

4.3.3 图片

（1）BMP文件组成

BMP文件由文件头、位图信息头、颜色信息和图形数据四部分组成。

（2）BMP文件头

BMP文件头数据结构含有BMP文件的类型、文件大小和位图起始位置等信息。

其结构定义如下：

```
typedef struct tagBITMAPFILEHEADER
{
    WORD bfType; // 位图文件的类型，必须为BM
    DWORD bfSize; // 位图文件的大小，以字节为单位
    WORD bfReserved1; // 位图文件保留字，必须为0
    WORD bfReserved2; // 位图文件保留字，必须为0
    DWORD bfOffBits; // 位图数据的起始位置，以相对位图文件头的偏移量表
                      //示，以字节为单位
} BITMAPFILEHEADER;
```

（3）位图信息头

BMP位图信息头数据用于说明位图的尺寸等信息。

```
typedef struct tagBITMAPINFOHEADER
{
    DWORD biSize; // 本结构所占用字节数
    LONG biWidth; // 位图的宽度，以像素为单位
    LONG biHeight; // 位图的高度，以像素为单位
    WORD biPlanes; // 目标设备的级别，必须为1
    WORD biBitCount; // 每个像素所需的位数，必须是1(双色),
                     // 4(16色), 8(256色)或24(真彩色)之一
    DWORD biCompression; // 位图压缩类型，必须是 0(不压缩),
                          // 1(BI_RLE8压缩类型)或2(BI_RLE4压缩类型)之一
    DWORD biSizeImage; // 位图的大小，以字节为单位
    LONG biXPelsPerMeter; // 位图水平分辨率，每米像素数
    LONG biYPelsPerMeter; // 位图垂直分辨率，每米像素数
    DWORD biClrUsed; // 位图实际使用的颜色表中的颜色数
    DWORD biClrImportant; // 位图显示过程中重要的颜色数
} BITMAPINFOHEADER;
```

（4）颜色表

颜色表用于说明位图中的颜色，它有若干个表项，每一个表项是一个RGBQUAD类型的结构，定义一种颜色。

RGBQUAD结构的定义如下：

```
typedef struct tagRGBQUAD
{
    BYTE rgbBlue; // 蓝色的亮度(值范围为0-255)
    BYTE rgbGreen; // 绿色的亮度(值范围为0-255)
    BYTE rgbRed; // 红色的亮度(值范围为0-255)
    BYTE rgbReserved; // 保留，必须为0
} RGBQUAD;
```

颜色表中RGBQUAD结构数据的个数有biBitCount来确定：

当biBitCount=1,4,8时，分别有2,16,256个表项；

当biBitCount=24时，没有颜色表项。

位图信息头和颜色表组成位图信息，BITMAPINFO结构定义如下：

```
typedef struct tagBITMAPINFO
{
    BITMAPINFOHEADER bmiHeader; // 位图信息头
    RGBQUAD bmiColors[1]; // 颜色表
} BITMAPINFO;
```

（5）位图数据

位图数据记录了位图的每一个像素值，记录顺序是在扫描行内是从左到右，扫描行之间是从下到上。位图的一个像素值所占的字节数：

当biBitCount=1时，8个像素占1个字节；

当biBitCount=4时，2个像素占1个字节；

当biBitCount=8时，1个像素占1个字节；

当biBitCount=24时，1个像素占3个字节；

Windows规定一个扫描行所占的字节数必须是4的倍数(即以long为单位),不足的以0填充， $biSizeImage = (((bi.biWidth * bi.biBitCount) + 31) \& \sim 31) / 8 * bi.biHeight$;

4.4 本章小结

本章主要介绍了触摸屏素描板的设计分析。介绍了整个软件的体系结构及功能模块的划分。该程序依据其功能分析，同样划分为四大功能模块：新建文件功能模块，打开文件功能模块，编辑文件功能模块和保存文件功能模块。还具体介绍了各个功能模块的实现步骤。最后列举了需要用到的重要数据结构：包括画笔，触点和图片。

第五章 程序实现

本章主要是针对触摸屏素描板的具体实现做出一个讨论。介绍了程序实现所需的环境和使用的环境，以及浅谈了一些主要的代码并添加了一些注释，用以帮助实现和理解整个程序。

5.1 开发环境

开发软件要求：

VC++6.0;

开发硬件要求：

GeneralTouch serial 以及 USB driver （版本是 GTDrv4.2.1.73I 或之后）

使用硬件要求：

GeneralTouch serial 以及 USB driver （版本是 GTDrv4.2.1.73I 或之后）

5.2 关键代码

5.2.1 触摸屏代码

在触摸屏上创建窗口，需要使用触摸屏的驱动。在程序使用了触摸屏的驱动后，在其他非触摸屏以及非该类触摸屏的计算机上无法运行。运行后通常会提示找不到驱动（not open driver）。该触摸屏生产商提供了触摸屏的窗口驱动，以及相关接口，和名为 GenTouchDLL.dll 的文件。在编程时，需要将生产商提供的名为 GenTouchDLL.dll 的文件，复制到程序的目录下，并进行调用。

具体实现如下所示：

```
//使用触摸屏驱动。
```

```
HANDLE hMyDevice;
```

```
hMyDevice = CreateFile(_T("\\\\.\\GENTOUCH"), GENERIC_READ |  
GENERIC_WRITE, 0, 0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
```

```
//如果没有可用的驱动，则进行错误提示。
```

```
if(hMyDevice == INVALID_HANDLE_VALUE)
```

```
{
```

```
    MessageBox(TEXT("not open driver"), NULL, MB_OK);
```

```
    exit(0);
```

```
}
```

```
CloseHandle(hMyDevice);
```

```
//调用名为 GenTouchDLL.dll 的文件
hDll = (HINSTANCE)LoadLibrary(_T("GenTouchDLL.dll"));
if(hDll == NULL)
{
    AfxMessageBox("can't link this dll.");
    exit(0);
}
```

5.2.2 工具栏代码

工具栏，就是在一个软件程序中,综合各种工具,让用户方便使用的一个区域。工具栏是显示位图式按钮行的控制条，位图式按钮用来执行命令。

工具栏类似于菜单栏，工具栏中的按钮类似于菜单栏中的选项。按工具栏按钮相当于选择菜单项；如果某个菜单项具有和工具栏按钮相同的 ID，那么使用工具栏按钮将会调用映射到该菜单项的同一个处理程序。例如：创建 MFC 程序时，编译器自动生成了主菜单和工具栏，并且二者的名称均为：IDR_MAINFRAME。工具栏中的按钮和菜单栏中的选项（如：新建，打开，保存）都一一对应，有着相同的 ID 名称，可以映射相同的处理程序。

具体实现如下：

```
//定义工具栏变量：
CToolBar    m_wndToolBar;

//创建工具栏
if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD | WS_VISIBLE |
CBRS_TOP|    CBRS_GRIPPER    |    CBRS_TOOLTIPS    |    CBRS_FLYBY    |
CBRS_SIZE_DYNAMIC) || !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
//加载相应工具栏
{
    TRACE0("Failed to create toolbar\n");
    return -1;        // fail to create
}
```

竖排工具栏：

竖排工具栏的实现实在横排工具栏的基础上，将工具栏中的按钮一一纵放。

```
//获得工具栏按钮的个数
```

```

int nCount1 = m_penwide.GetToolBarCtrl().GetButtonCount();
//将按钮一一纵放
for(int i1=0;i1<nCount1;i1++)
{
    UINT nStyle=m_penwide.GetButtonStyle(i1);
    nStyle|=TBBS_WRAPPED; //分多行显示
    m_penwide.SetButtonStyle(i1,nStyle);
}

```

5.2.3 对话框代码

对话框是一种次要窗口，包含按钮和各种选项，通过它们可以完成特定命令或任务。对话框是人机交流的一种方式，用户对对话框进行设置，计算机就会执行相应的命令。对话框中有单选框、复选框等。

对话框类：在 MFC 程序中，可以使用向导帮助用户建立一个与对话框资源相关联的类，通常这个类由 CDialog 类派生。

Cdialog:

为了能够方便的操作对话框，MFC 为用户提供了 CDialog 类。它是在屏幕上显示对话框的基类，与对话框资源紧密相关，提供了管理对话框的接口，封装了一些对话框的相关操作。

（1）创建对话框：

需要创建相应的对话框类，并编写相应操作的函数。

本程序主要包括两个对话框：设置映射参数的 Set 对话框，以及设置笔画宽度的 PenWide 对话框。初始化对话框 BOOL OnInitDialog()中的重要代码：

（2）定义滑块：

```
CSliderCtrl m_slider;
```

当用户进行交互操作时，滑动条控制将向其父窗口发送消息 WM_HSCROLL，所以在应用程序中应重载父窗口的 OnHScroll()成员函数，以便对消息进行正确处理系统发送的通知代码、滑标位置和指向 CSliderCtrl 对象的指针等。

（3）初始化滑块 slider:

```

m_slider.SetRange(1,10); //设置滑块的滑动范围
m_slider.SetTicFreq(1); //设置滑块相邻间隔
m_slider.SetPos(5); //设置滑块的初始默认值

```

（4）滑块的操作事件函数：

OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)

当用户进行交互操作时，滑动条控制将向其父窗口发送消息 WM_HSCROLL，所以在应用程序中应重载父窗口的 OnHScroll()成员函数，以便对消息进行正确处理系统发送的通知代码、滑标位置和指向 CSliderCtrl 对象的指针等。

```
if(nSBCode==SB_THUMBPOSITION){
    m_eData.Format("%ld",nPos);    //在编辑框中显示滑块的位置
    CPublic::penwide=atoi(m_eData); //将滑块的位置赋值到画笔的宽度。
    UpdateData(FALSE);    //拷贝变量值到控件显示。(变量的最终运算结果
                          //值交给外部输出显示)即：变量—>控件显示。
}
else{
    CDialog::OnHScroll(nSBCode,nPos,pScrollBar);
}
CDialog::OnHScroll(nSBCode, nPos, pScrollBar);
```

（5）更换对话框的背景：

更换对话框的背景图片有许多方法，这里选用了较为便捷的方法，重载 OnPaint() 函数，在其中绘制背景

```
void PenWide::OnPaint()
{
    CPaintDC dc(this); // device context for painting

    // TODO: Add your message handler code here
    CRect rect;
    GetClientRect(&rect); //得到窗体的大小
    CDC dcMem;
    dcMem.CreateCompatibleDC(&dc);
    CBitmap bmpBackground;
    bmpBackground.LoadBitmap(IDB_BITMAP3); //加载背景图片
    BITMAP bitMap;
    bmpBackground.GetBitmap(&bitMap);
    CBitmap *pbmpOld=dcMem.SelectObject(&bmpBackground);

    dc.StretchBlt(0,0,rect.Width(),rect.Height(),&dcMem,0,0,bitMap.bmWidth,bitMap.bmHeight,SRCCOPY); //将图片拉伸至填满对话框窗口。
    // Do not call CDialog::OnPaint() for painting messages
```

```
}

```

(6) 编辑框显示内容:

```
itoa(CPublic::maxstr,a,10); //将 int 类型变为 char 类型
m_MaxStrength.SetWindowText(a); // 在编辑框中显示内容

```

(7) 将编辑框中的输入值记录到后台程序:

```
m_MaxStrength.GetWindowText(mas); //获取编辑框中的内容
CPublic::maxstr=_ttoi(mas); //将编辑框中的 char 型内容变为 int 型，然后赋
//值给变量保存。

```

5.2.4 画笔栏代码

画笔栏中定义了九根笔尖宽度不同的铅笔，4H~4B，4H 最宽，为 4.5，4B 最窄，为 1。依次以 0.5 的间隔递增。

以 4H 为例：

```
void CMiniDrawDoc::OnPW4H(){
    CPublic::penwide=4.5; //设置画笔宽度
    CPublic::tempPenwide=CPublic::penwide; //记录此时画笔宽度，以便之后调用
    CPublic::rubFlag=0; //将橡皮标记重置
}

```

5.2.5 绘图栏代码

(1) 削粗铅笔:

```
void CMiniDrawDoc::OnPenCut()
{
    CPublic::rubFlag=0;
    CPublic::lightFlag=0;

    //打开设置画笔的对话框
    PenWide* dlg=new PenWide; //定义设置画笔粗细的对话框
    dlg->Create(MAKEINTRESOURCE(IDD_PENWIDE)); //创建设置画笔
    //粗细的对话框
    dlg->ShowWindow(1); //显示设置画笔粗细的对话框
    CPublic::tempPenwide=CPublic::penwide;

}

```

（2）橡皮代码：

橡皮即是一种与背景颜色相同的画笔。本程序设置的画板背景为白色，因此橡皮画笔的颜色为白色。

```
void CMiniDrawDoc::OnRubberWipe()
{
    CPublic::rubFlag=1; //设置橡皮标记
    CPublic::lightFlag=0;
    CPublic::tempPenwide=CPublic::penwide;
    CPublic::penwide=10;    //将橡皮画笔的宽度设为定值 10
}
```

5.2.6 新建文件代码

新建文件就是将绘画区域填充为背景色，而该触摸屏素描板程序以白色为背景色，因此新建只需将绘画区域填充为白色。

```
void CMiniDrawDoc::OnFileNew()
{
    CPublic::clearFlag=1;    //设置新建标记
    UpdateAllViews(0); //刷新窗口
    SetModifiedFlag(); //设置修改标记
}
```

刷新时调用 onDraw 函数，onDraw 函数进行判断。

```
void CMiniDrawView::OnDraw(CDC* pDC)
{
    CMiniDrawDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    //若新建标记为 1
    if(CPublic::clearFlag==1){
        CPublic::clearFlag=0;    //重置新建标记
        RECT rect;
        GetClientRect(&rect);    //获取绘画区域的大小
        pDC->FillSolidRect(&rect,RGB(255,255,255));    //将该窗口填充白色的
                                                         //背景色
    }
}
```

5.2.7 编辑文件代码

编辑文件是当打开了窗口或打开了一幅外界图画，然后再在绘画区内编辑图片。用户需要手动触摸屏幕进行画线。

线是用点连接而成，所以画线的基础是画点。素描板的点我们用 `Cpoint` 类来定义。

`CPoint` 类与 Windows `POINT` 结构类似。它还包括用来操纵 `CPoint` 和 `POINT` 结构的成员函数。

只要 `POINT` 结构可以使用的地方，`CPoint` 对象也可以使用。这个类与“大小”有关的操作符可以接受 `CSize` 对象或 `SIZE` 结构，因为这两者是可以互换的。

（1）定义旧点和起始点：

```
CPoint m_PointOld;
CPoint m_PointOrigin;
```

（2）画线的基本函数有：

```
CDC::MoveTo( int x, int y ); 改变当前点的位置
CDC::LineTo( int x, int y ); 画一条由当前点到参数指定点的线
CDC::BOOL Arc( LPCRECT lpRect, POINT ptStart, POINT ptEnd );
画弧线
CDC::BOOL Polyline( LPPPOINT lpPoints, int nCount );
将多条线依次序连接
```

本素描板程序只用到前两个函数：`CDC::MoveTo(int x, int y);` 和 `CDC::LineTo(int x, int y)`。

（3）绘画函数：

当用户按压触摸屏素描板界面中的绘画区域时，`OnLButtonDown(UINT nFlags, CPoint point)`函数响应。

当用户在触摸屏素描板界面中的绘画区域结束一笔线条时，需要抬起手，离开触点，因此 `OnLButtonUp(UINT nFlags, CPoint point)`函数响应。

当用户按压触摸屏素描板界面中的绘画区域后继续按压并移动时，`OnMouseMove(UINT nFlags, CPoint point)`函数响应。

（4）画曲线的函数：

其中，画曲线的实现是建立在画直线的基础上的。增加了不断移动点并且连线的代码。如下所示：

```
CClientDC dc (this);
```

```
dc.SelectObject(&pen);

if(m_bDraw==TRUE){
    dc.MoveTo(m_PointOrigin);
    dc.LineTo (point);
    m_PointOrigin=point;
}
```

（5）绘图模式函数：

在绘画工程中，有用到 Windows API SetROP2(int nDrawMode)。

SetROP2(int nDrawMode)函数的主要的作用是：

根据 nDrawMode 设置的方式重新设定绘图的方式，下面就不同的 nDrawMode 值具体解释绘图模式是如何改变的。

首先就 nDrawMode 的取值有以下的情况：

- 1、R2_BLACK Pixel is always black. //所有绘制出来的像素为黑色
- 2、R2_WHITE Pixel is always white. //所有绘制出来的像素为白色
- 3、R2_NOP Pixel remains unchanged. //任何绘制将不改变当前的状态
- 4、R2_NOT Pixel is the inverse of the screen color. //当前绘制的像素值设为屏幕像素值的反，这样可覆盖掉上次的绘图，（自动擦除上次绘制的图形）
- 5、R2_COPYPEN Pixel is the pen color. //使用当前的画笔的颜色
- 6、R2_NOTCOPYPEN Pixel is the inverse of the pen color. //当前画笔的反色

总之，上述 api 的一个作用是在需要改变绘图的模式时，不需要重新设置画笔，只需要设置不同的绘图的模式即可达到相应的目的。

在此触摸屏素描板编程中用到了 R2_NOT 和 R2_COPYPEN。

（6）获取力度代码：

获取力度的函数及数据结构由触摸屏开发商提供。

```
typedef BOOL (WINAPI *lpGetExtraInfo)(HANDLE, PULONG);
lpGetExtraInfo GetExtraInfo;
```

```
typedef BOOL (WINAPI *lpfn)(HANDLE, PCALINFO);
lpfn GetDeviceRawData;
```

```
ULONG zcoordinate = 0;
CALINFO CalibInfo;
```



```

    UCHAR zdata = 0;

    GetDeviceRawData =
    (lpfn)GetProcAddress(hDll,(LPCTSTR)"GTGetDeviceRawData");
    GetExtraInfo =
    (lpGetExtraInfo)GetProcAddress(hDll, (LPCTSTR)"GTGetExtraInfo");

    else if(GetZInfo(&zcoordinate) && GetCOMRawData(&CalibInfo))
    {
        zdata = zcoordinate >> 16;    //zdata 为按压触摸屏的力度值。(zcoordinate
                                        //中包含该力度值，但需向后移动十六位。)

    BOOL GetZInfo(PULONG pZcoordinate)
    {
        BOOL      result;
        HANDLE     hMyDevice;

        _ASSERT( FALSE == IsBadWritePtr(pZcoordinate,sizeof(ULONG)) );

        if(TRUE == IsBadWritePtr(pZcoordinate,sizeof(ULONG)))
        {
            SetLastError(ERROR_INVALID_PARAMETER);
            return  FALSE;
        }

        hMyDevice = CreateFile(_T("\\\\.\\GENTOUCH"), GENERIC_READ |
                                GENERIC_WRITE, 0, \
                                0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);

        if(hMyDevice == INVALID_HANDLE_VALUE) return FALSE;
        result = GetExtraInfo(hMyDevice, pZcoordinate);
        return result;
    }

    BOOL GetCOMRawData(PCALINFO pRawInfo)
    {
        BOOL      result;

```

```

HANDLE      hMyDevice;

_ASSERTE( FALSE == IsBadWritePtr(pRawInfo,sizeof(CALINFO)) );

if(TRUE == IsBadWritePtr(pRawInfo,sizeof(CALINFO)))
{
    SetLastError(ERROR_INVALID_PARAMETER);
    return  FALSE;
}

hMyDevice = CreateFile(_T("\\\\.\\GENTOUCH"), GENERIC_READ |
GENERIC_WRITE, 0, \
0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);

if(hMyDevice == INVALID_HANDLE_VALUE) return FALSE;
result = GetDeviceRawData(hMyDevice, pRawInfo);
CloseHandle(hMyDevice);
return result;
}

```

（7）重绘函数：

当窗口最小化，最大化或者有其他窗口覆盖时，都执行了刷新，但用户在绘图区域内的画是没有保存的，之显示在了屏幕上。为了再进行刷新时，绘画不至于丢失，我们需要先将笔迹写入内存，然后刷新后，从内存中调用，再显示在屏幕上。

在抬笔函数 OnLButtonUp(UINT nFlags, CPoint point)中，需要添加写入笔迹至内存的函数。如下所示：

```

CDC *pDC = GetDC();
CMiniDrawDoc *pDoc = GetDocument();
CDC dcMem;

dcMem.CreateCompatibleDC(NULL);
CRect rect;
GetClientRect(&rect); //获取客户区域
pDoc->m_bmpBuf.DeleteObject();
pDoc->m_bmpBuf.CreateCompatibleBitmap(pDC,rect.Width(),rect.Height());
CBitmap *pOldBitmap = dcMem.SelectObject(&pDoc->m_bmpBuf); //将创建

```

//好的 m_bmpBuf 添加到临时的 CDC 的 Object 中，类似于在墙上先糊上墙
//纸，先在墙纸上作画，最后将最终形成的画拷贝到墙上。

//将 pDC 即当前客户区里面的内容拷贝到临时的 MEM 中，MEM 虽然过后
//会被 delete 掉，但是它更新了 CDocument 类中的 m_bmpBuf
dcMem.BitBlt(0,0,rect.Width(),rect.Height(),pDC,0,0,SRCCOPY);
dcMem.SelectObject(pOldBitmap);
dcMem.DeleteDC();

在窗口刷新后，会调用 onDraw (CDC* pDC) 函数来重绘，此时需要在该
函数中添加从内存中导入图片信息，并显示的程序。如下所示：

```
LPCTSTR filename="abc.bmp";

if (!pDoc)
    return;
CRect rect;
GetClientRect(&rect);
CDC dcMem; //以下是输出位图的标准操作
CBitmap *pOldBitmap = NULL;
dcMem.CreateCompatibleDC(NULL);
pOldBitmap = dcMem.SelectObject(&pDoc->m_bmpBuf);

//将 dcMem 中的 bitmap 拷贝到当前客户区
pDC->BitBlt(0,0,rect.Width(),rect.Height(),&dcMem,0,0,SRCCOPY);
dcMem.SelectObject(pOldBitmap); dcMem.DeleteDC();
```

（8）换算深浅代码

该触摸屏素描板中提供参数映射设置，用户可以定义最大力值 maxstr，最小力值 minstr，最深色值 maxclr，最浅色值 minclr。这四个值通过换算，可以将用户的按压力度与所对应的颜色深浅挂钩，得到该力度下笔迹的颜色深浅值，从而显示到绘画区域的按压处。（抬笔处的换算算法相同，按压移动的换算算法也相同，只不过利用定时器，间隔一段时间后获取力度，换算，然后显示）

$$\text{显示颜色灰度值} = \frac{\text{最大力值} - \text{最小力值}}{\text{最浅色值} - \text{最深色值}} \times (\text{按压力度值} - \text{最小力度值})$$

代码如下：

```
float t1,t2,t3;
```

```

float t4,t5;
t1=CPublic::minclr-CPublic::maxclr;
t2=CPublic::maxstr-CPublic::minstr;

//若该力度值大于设置的最大值，则颜色深度为设置的最深色
if(zdata>=CPublic::maxstr)
    CPublic::aa=CPublic::maxclr;
else{

    //如果该力度值大于设置的最小值，则进行区间换算
    if(zdata>CPublic::minstr)
        t3=zdata-CPublic::minstr;
    //反之，若该力度值小于设置的最小值，则颜色深度为设置的最浅色
    else
        t3=0;

    t4=t2/t1;
    t5=t4*t3;

    //画笔的灰度值等于最浅色与区间换算值之差
    CPublic::aa=CPublic::minclr-t5;
}

```

5.2.8 打开文件代码

在该触摸屏素描板中，打开文件为打开外部 bmp 格式的图片，并显示在绘画区域中。具体实现如下所示：

```

void CMiniDrawView::OnFileOpen()
{
    //跳出打开对话框，并返回输入值
    CFileDialog
    dlg(TRUE,NULL,NULL,OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT,
    T_T("位图文件(*.BMP)|*.BMP"));
    //可以通过修改这里实现打开其他格式的图片

    if (IDOK==dlg.DoModal())
    {
        state=0;
    }
}

```

```

        BmpName.Format(_T("%s"),dlg.GetPathName());
        ShowBitmap(BmpName);
        extname = dlg.GetFileExt();           //返回选定文件的扩展文件名
    }
}

```

打开后需要在绘图区域显示 bmp 格式的图片。

```

void CMiniDrawView::ShowBitmap(CString BmpName)
{
    if (state==0)
    {
        HBITMAP
        hBitmap=(HBITMAP)LoadImage(NULL,BmpName,IMAGE_BITMAP,0,0,LR_CREATEDIBSECTION|LR_DEFAULTSIZE|LR_LOADFROMFILE);
        m_bitmap.Detach();
        m_bitmap.Attach(hBitmap);
        state=1;
        Invalidate();
    }
}

```

5.2.9 保存文件代码

在该触摸屏素描板中，保存文件为保存到外部 bmp 格式的图片中，并显示在绘画区域中。具体实现如下所示：

```

void CMiniDrawView::OnFileSave()
{
    // TODO: Add your command handler code here
    //CFileDialog dlg(FALSE, "bmp","位图文件(*.bmp)|无标题.bmp");
    CFileDialog      dlg(FALSE,      NULL,"      无      标
    题 .bmp",OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT," 位 图 文 件
    (*.bmp)",NULL);

    if(dlg.DoModal() != IDOK){
        return ;
    }
    CString filePath = dlg.GetPathName();
}

```

```

// TODO: Add your command handler code here
CDC cdc;
HDC hDC1;
CBitmap bitmap;
RECT rect;CRect r;
GetClientRect(&rect);
CClientDC client(this);
int cx = rect.right - rect.left;
int cy = rect.bottom - rect.top;
bitmap.CreateCompatibleBitmap(&client, cx, cy);
cdc.CreateCompatibleDC(NULL);
//获取 BMP 对象
CBitmap * oldbitmap = (CBitmap* ) cdc.SelectObject(&bitmap);
//白色画布
cdc.FillRect(&rect,
CBrush::FromHandle((HBRUSH)GetStockObject(WHITE_BRUSH)));
OnDraw(&cdc); //画图
cdc.SelectObject(oldbitmap);
::OpenClipboard(this->m_hWnd);
::EmptyClipboard();
::SetClipboardData(CF_BITMAP, bitmap);
::CloseClipboard();

HBITMAP hBitmap = (HBITMAP)bitmap;
SaveBMPToFile(hBitmap,filePath);
}

bool SaveBMPToFile(HBITMAP hBitmap, CString lpFileName) //hBitmap 为刚才
的屏幕位图句柄,lpFileName 为位图文件名
{

HDC hDC; //设备描述表
int iBits; //当前显示分辨率下每个像素所占字节数
WORD wBitCount; //位图中每个像素所占字节数
//定义调色板大小, 位图中像素字节大小, 位图文件大小, 写入文件字节数
DWORD dwPaletteSize=0,dwBmBitsSize,dwDIBSize, dwWritten;
BITMAP Bitmap; //位图属性结构

```

```

BITMAPFILEHEADER  bmfHdr; //位图文件头结构
BITMAPINFOHEADER  bi;      //位图信息头结构
LPBITMAPINFOHEADER lpbi; //指向位图信息头结构

HANDLE fh, hDib, hPal;
HPALETTE hOldPal=NULL; //定义文件， 分配内存句柄， 调色板句柄

//计算位图文件每个像素所占字节数
hDC = CreateDC("DISPLAY", NULL, NULL, NULL);
iBits = GetDeviceCaps(hDC, BITSPIXEL) * GetDeviceCaps(hDC, PLANES);
DeleteDC(hDC);
if (iBits <= 1)  wBitCount = 1;
else if (iBits <= 4)  wBitCount = 4;
else if (iBits <= 8)  wBitCount = 8;
else              wBitCount = 24;

GetObject(hBitmap, sizeof(Bitmap), (LPSTR)&Bitmap);
bi.biSize      = sizeof(BITMAPINFOHEADER);
bi.biWidth     = Bitmap.bmWidth;
bi.biHeight    = Bitmap.bmHeight;
bi.biPlanes    = 1;
bi.biBitCount  = wBitCount;
bi.biCompression = BI_RGB;
bi.biSizeImage = 0;
bi.biXPelsPerMeter = 0;
bi.biYPelsPerMeter = 0;
bi.biClrImportant = 0;
bi.biClrUsed     = 0;

dwBmBitsSize = ((Bitmap.bmWidth * wBitCount + 31) / 32) * 4 *
Bitmap.bmHeight;

hDib      = GlobalAlloc(GHND,dwBmBitsSize + dwPaletteSize +
sizeof(BITMAPINFOHEADER));
lpbi = (LPBITMAPINFOHEADER)GlobalLock(hDib);
*lpbi = bi;

```

```

    hPal = GetStockObject(DEFAULT_PALETTE);
    if (hPal)
    {
        hDC = ::GetDC(NULL);
        hOldPal = ::SelectPalette(hDC, (HPALETTE)hPal, FALSE);
        RealizePalette(hDC);
    }

    GetDIBits(hDC, hBitmap, 0, (UINT) Bitmap.bmHeight, (LPSTR)lpbi +
    sizeof(BITMAPINFOHEADER)
        +dwPaletteSize, (BITMAPINFO *)lpbi, DIB_RGB_COLORS);

    if (hOldPal)
    {
        ::SelectPalette(hDC, (HPALETTE)hOldPal, TRUE);
        RealizePalette(hDC);
        ::ReleaseDC(NULL, hDC);
    }

    fh = CreateFile(lpFileName, GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
        FILE_ATTRIBUTE_NORMAL | FILE_FLAG_SEQUENTIAL_SCAN, NULL);

    if (fh == INVALID_HANDLE_VALUE)
        return FALSE ;

    bmfHdr.bfType = 0x4D42; // "BM"
    dwDIBSize = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) +
    dwPaletteSize + dwBmBitsSize;
    bmfHdr.bfSize = dwDIBSize;
    bmfHdr.bfReserved1 = 0;
    bmfHdr.bfReserved2 = 0;
    bmfHdr.bfOffBits = (DWORD)sizeof(BITMAPFILEHEADER) +
    (DWORD)sizeof(BITMAPINFOHEADER) + dwPaletteSize;

    WriteFile(fh, (LPSTR)&bmfHdr, sizeof(BITMAPFILEHEADER), &dwWritten,
    NULL);
    WriteFile(fh, (LPSTR)lpbi, dwDIBSize, &dwWritten, NULL);

```



```
GlobalUnlock(hDib);
GlobalFree(hDib);
CloseHandle(fh);
return TRUE;
}
```

5.2.10 全局变量

在多文档编程中，不同的文档需要调用，赋值或修改相同的参数。这些参数必须以全局变量的名义存在，并且属性为 `public`，才能被其他的文档所调用。

定义全局变量如下所示：

```
class CPublic{
public:
    CPublic();
    virtual ~CPublic();

public:
    static int rubFlag;
    static int clearFlag;
    static float penwide;
    static float tempPenwide;
    static int aa;
    static int maxstr;
    static int minstr;
    static int maxclr;
    static int minclr;
};
```

当一个文档需要调用全局变量时，需要在该文档头部先重定义全局变量。

例如：

```
int CPublic::rubFlag = 0;
```

使用时需要加上公共的全局变量类名，如下所示：

```
CPublic::rubFlag = 1;
```

5.2.11 定时器

用户在素描板的绘图区域进行绘画时，后台程序需要即时勘测力度，换算深浅，从而显示到屏幕界面的触点处。

```
//在窗口生成时，启用定时器函数
void CMainFrame::OnTimer(UINT nIDEvent)
{
    ULONG zcoordinate = 0;
    CALINFO CalibInfo;
    UCHAR zdata = 0;

    SetTimer(1,100,NULL); //设置定时器间隔

    CMiniDrawDoc* cmd;    //调用后台程序
    cmd->te();
    UpdateData(FALSE);
}

//后台程序
void CMiniDrawDoc::te()
{
    CMiniDrawView* c; //调用界面勘测触摸屏力度参数的函数
    c->timeStr();
}

//界面勘测触摸屏力度参数的函数
void CMiniDrawView:: timeStr ()
{
    ULONG zcoordinate = 0;
    CALINFO CalibInfo;
    UCHAR zdata = 0;

    //如果是橡皮，直接将颜色改变为白色。
    if(CPublic::rubFlag==1){
        CPublic::aa=255;
    }
    else if(StartFlag==1){
        GetZInfo(&zcoordinate);
    }
}
```

```

GetCOMRawData(&CalibInfo);
zdata = zcoordinate >> 16;    //获取力度值

//换算力度值到颜色的灰度值
float t1,t2,t3;
float t4,t5;
t1=CPublic::minclr-CPublic::maxclr;
t2=CPublic::maxstr-CPublic::minstr;
if(zdata>=CPublic::maxstr)
    CPublic::aa=CPublic::maxclr;
else{

    if(zdata>CPublic::minstr)
        t3=zdata-CPublic::minstr;
    else
        t3=0;

    t4=t2/t1;
    t5=t4*t3;

    CPublic::aa=CPublic::minclr-t5;

}
}
}

```

5.3 本章小结

本章着重介绍了该触摸屏素描板软件的最终实现的重要的程序代码包括：触摸屏代码，工具栏，竖排工具栏，对话框，画笔栏代，绘图栏，橡皮代码，新建文件，编辑文件，获取力度代码，换算深浅代码，打开文件，保存文件，全局变量，定时器，重绘函数。

第六章 程序测试

本章主要展示了程序测试的结果。对程序的测试包括单元测试和整合测试两部分。单元测试是对各个部分进行一个分开的测试和分析。整合测试是最终的运行结果，以及整个使用过程的测试。

6.1 对程序进行单元测试

对程序的单元测试展示了在试验完成后，触摸屏素描板软件最终实现的界面。包括主窗体，工具栏及对话框等。

6.1.1 程序主窗体

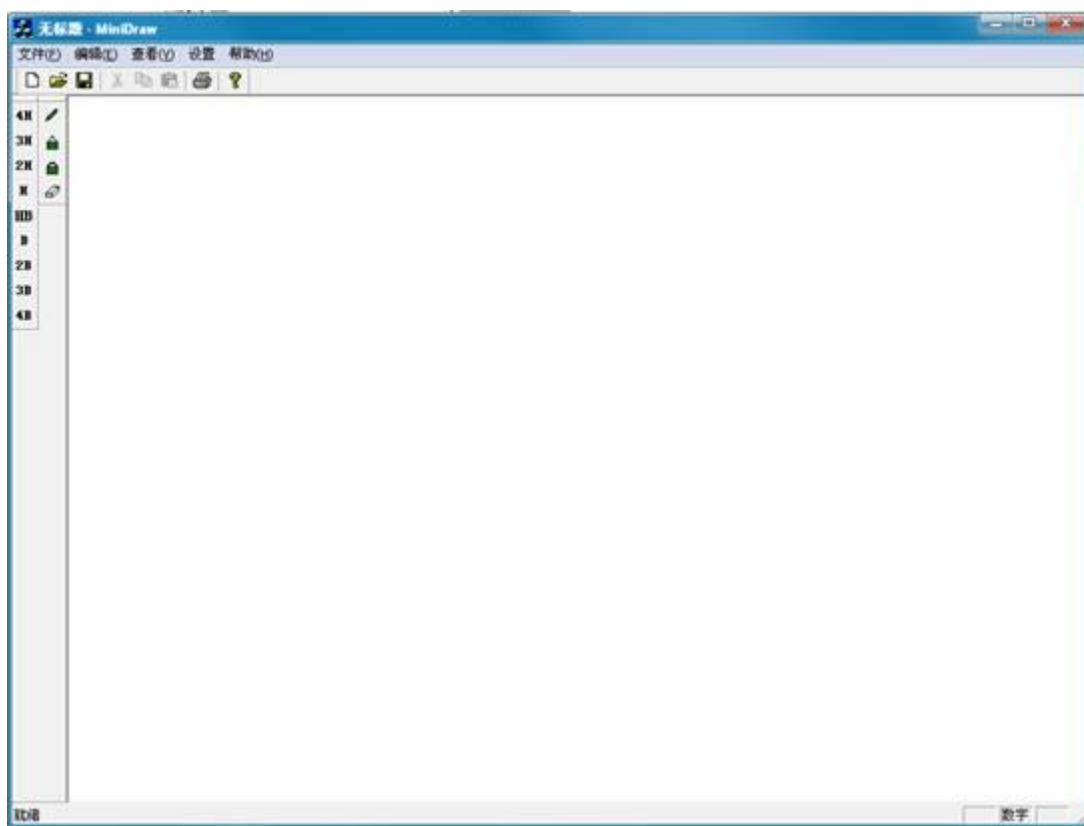


图 6-1 触摸屏素描板的程序主窗体界面

该主窗体分为如下五部分：

菜单栏：界面最顶一层

工具栏：菜单栏下方一层

画笔栏：左侧第一纵列

绘图栏：绘图栏右边纵列

绘画区域：界面中的空白区域

6.1.2 菜单栏

文件(F) 编辑(E) 查看(V) 设置 帮助(H)

图 6-2 触摸屏素描板菜单栏界面

(1) 文件:

文件菜单下包括的选项：新建，打开，保存，另存为，打印，打印预览，打印设置，退出。

文件(F)	编辑(E)	查看(V)	设置	帮助(H)
新建(N)		Ctrl+N		
打开(O)...		Ctrl+O		
保存(S)		Ctrl+S		
另存为(A)...				
打印(P)...		Ctrl+P		
打印预览(V)				
打印设置(R)...				
最近文件				
退出(X)				

图 6-3 触摸屏素描板文件菜单选项

(2) 编辑:

编辑菜单栏下包括的选项：撤销，全部删除。

编辑(E)	查看(V)	设置
撤销(U)	Ctrl+Z	
全部删除		

图 6-4 触摸屏素描板编辑菜单选项

(3) 查看:

查看菜单栏下包括的选项：工具栏，状态栏

查看(V)	设置	帮助(H)
工具栏(T)		
状态栏(S)		

图 6-5 触摸屏素描板文件查看菜单选项

（4）设置：

设置菜单栏下包括的选项：设置映射参数

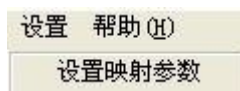


图 6-6 触摸屏素描板文件设置菜单选项

（5）帮助：

帮助菜单栏下包括的选项：帮助内容，关于 MiniDraw。

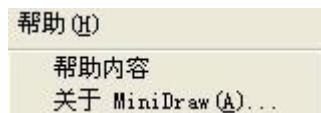


图 6-7 触摸屏素描板文件帮助菜单选项

6.1.3 工具栏

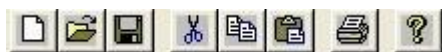





图 6-8 触摸屏素描板文件工具栏界面

 新建文件：新建画图，新建后画面为空白。

 打开文件：打开已有的 bmp 图画，打开后绘画区域显示该图片，并可进行编辑。

 保存文件：保存已经绘制好的图画，bmp 格式导出。

 剪切：复制并清除。

 复制：重新拥有已有部分的备份。



粘贴：粘贴已复制部分。



打印文件：打印所画图片。



帮助：打开程序帮助内容

6.1.4 画笔栏



图 6-9 触摸屏素描板画笔栏界面

该画笔栏为用户提供了九支不同的铅笔，每只铅笔都有不同但是固定的粗细。笔尖的宽度从 4B 到 4H 逐渐增大（4B 为 1，4H 为 4.5，每支相邻的铅笔宽度间隔为 0.5）。

用户可自选已定铅笔进行会话，方便使用。若想改为其他的宽度，可在铅笔的宽度设置中进行设置。

画笔的粗细由 4H-4B，如图所示

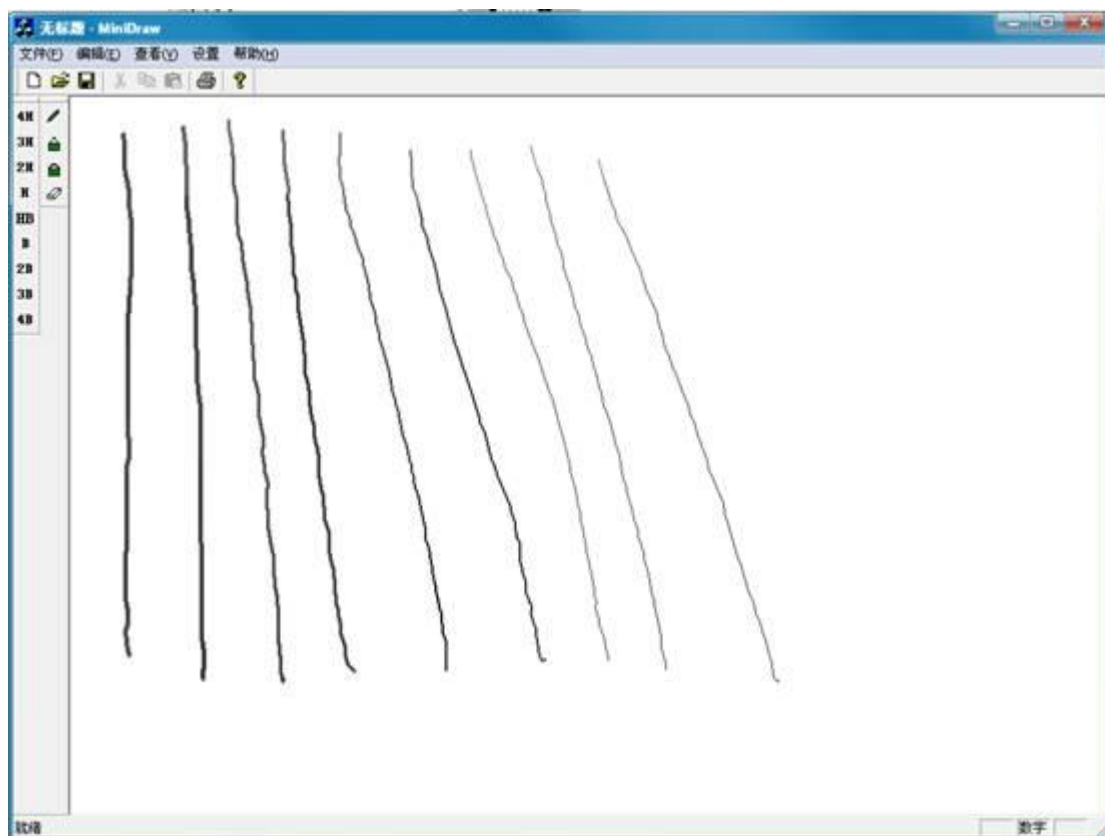



图 6-10 触摸屏素描板画笔粗细展示图


6.1.5 绘图栏





图 6-11 触摸屏素描板绘图栏界面

该绘图栏为用户提供了绘图工具：铅笔，削尖铅笔，改变铅笔宽度以及橡皮。

 铅笔： 该工具用于橡皮到铅笔的转换，并且可以保留在使用橡皮前所设置的笔尖宽度。

 削尖铅笔： 将铅笔的笔尖宽度变为最细，即 1。

 设置画笔粗细： 若画笔栏中没有想要的笔尖宽度，可在这里自行设置，由 1 到 10，间隔为 1。（关于画笔粗细的设置，见“设置”中的“设置画笔粗细”）

 橡皮： 擦除颜色。宽度为默认值 10。

6.1.6 打开文件对话框

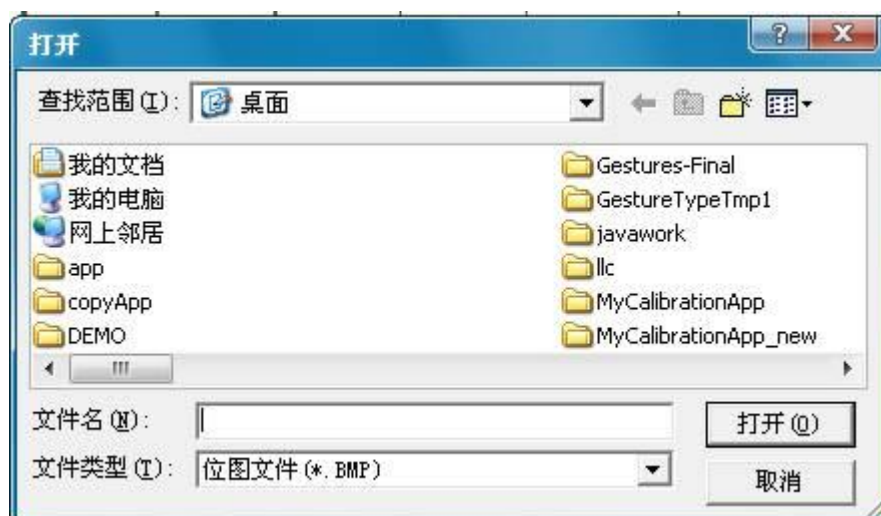


图 6-12 触摸屏素描板打开文件对话框界面

用户可以在对话框中选择需要打开的文件，但只允许打开 bmp 格式的图片。

6.1.7 保存文件对话框



图 6-13 触摸屏素描板保存文件对话框界面

用户可以在对话框中选择需要保存的路径以及保存文件的名称，但只允许保存为 bmp 格式的图片。

6.1.8 设置映射参数的对话框



图 6-14 触摸屏素描板设置映射参数对话框界面

最大力：范围是 1-255。值越大，力越大。

用户决定自己最大可以使用的力的大小。当实际用力超过最大力值时视为所设置的最大力值

最小力：范围是 1-255。值越小，力越小。

用户决定自己最小可以使用的力的大小。当实际用力小于最小力值时视为所设置的最小力值

最深色：范围是 1-255。值越小，色越深。

用户决定当自己使用超过或等于最大力值的力时，绘图线条颜色的深度。

最浅色：范围是 1-255。值越大，色越浅。

用户决定当自己使用小于或等于最小力值的力时，绘图线条颜色的浅度。

还原默认值：默认值为：最大力：150，最小力：5，最深色：20，最浅色：120。

6.1.9 设置画笔粗细的对话框

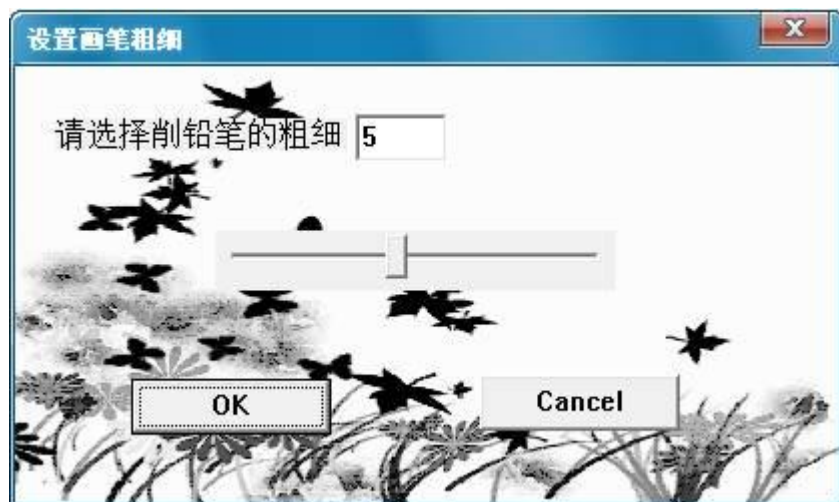


图 6-15 触摸屏素描板设置笔画粗细对话框界面

可以拖动滑块进行笔尖宽度设置。

画笔的宽度范围为 1-10，相邻间隔为 1。

默认的初始值为 5。

6.2 对程序进行整合测试

对程序的整合测试展示了在试验完成后，触摸屏素描板软件最终应用效果。包括绘画成果，保存导出等。

6.2.1 绘画成果

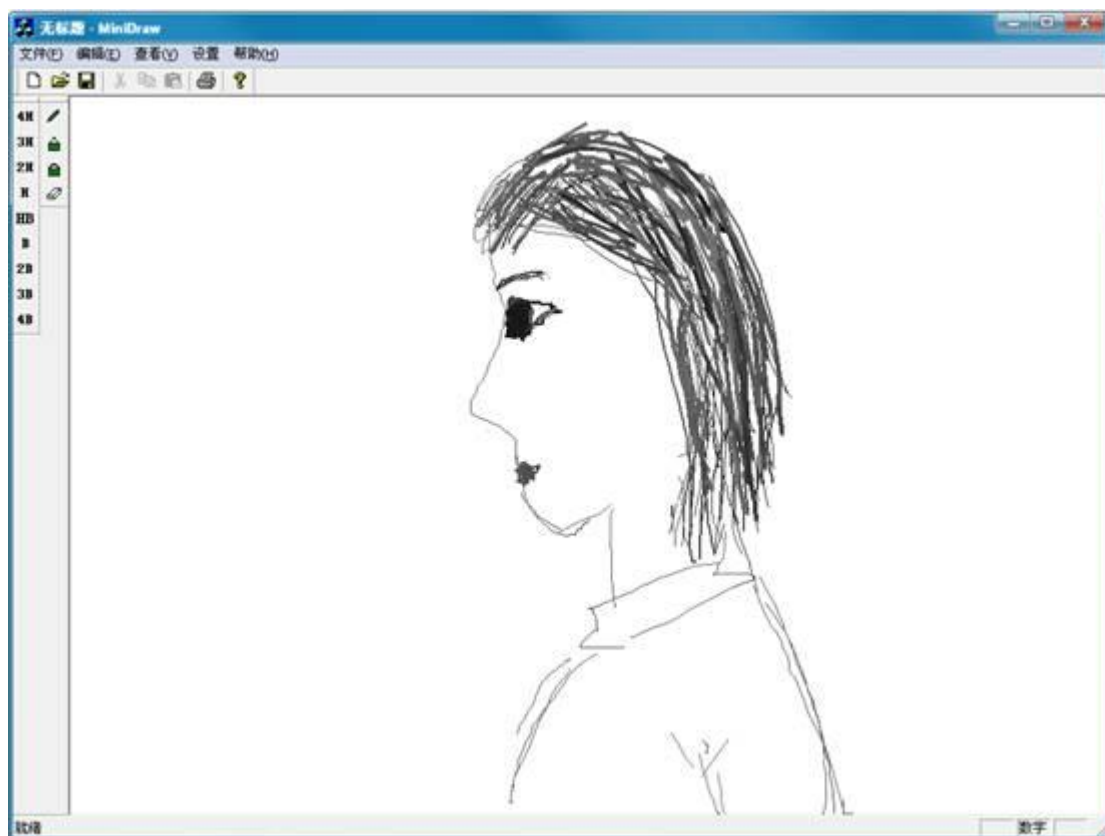


图 6-16 触摸屏素描板绘画结果截图

此图为在运行的触摸屏素描板程序下画出的简易人物素描。

6.2.2 保存结果



图 6-16 触摸屏素描板绘画内容保存导出结果截图

在绘画区域中绘画的内容通过保存可以导出为 bmp 格式的图片文件。

6.3 本章小结

本章着重介绍了该触摸屏素描板软件测试后的界面以及重要的程序代码。展示的界面包括整个主窗口，工具栏，画笔栏，绘图栏，设置画笔粗细和设置参数映射的窗口以及打开文件和保存文件的窗口。

第七章 遇到的问题解决方法

本章主要介绍了在进行本次毕业设计的课题——基于三维触摸屏的素描板软件的开发的过程中所遇到的问题，以及最终的解决方案。

7.1 问题及解决方案

1. 存在的问题：

触摸屏素描板的曲线无法绘画。

解决的方法：

抛开触摸屏，在普通的机器进行画板的研究。

2. 存在的问题：

画曲线有两种方式，一种方法是用像素点：`SetPixel(int x, int y, COLORREF crColor)`，这样做方便实时勘测用户的力度，从而反应绘画笔迹的深浅。另一种方法是使用画线函数 `moveTo()`和 `lineTo()`，这样做方便绘图，勘测力度可以用定时器间隔测量，并反馈至画板。

解决的方法：

选择第二种画线方式。

因为虽然在 Windows 中画点的方法很简单，只需要调用 `COLORREF CDC::SetPixel(int x, int y, COLORREF crColor)`就可以在指定点画上指定颜色，同时返回原来的颜色。`COLORREF CDC::GetPixel(int x, int y)`可以得到指定点的颜色。在 Windows 中应该少使用画点的函数，这样做的执行效率比较低，而且需要储存大量的点信息，也是一种压力。

3. 存在的问题：

当画笔笔尖宽度大于 1 时，两条黑色相交线交接处呈现白色。

解决的方法：

去掉取反色函数 `SetROP2(R2_NOT)`

4. 存在的问题：

当画笔颜色非黑时，中间有虚线。

解决的方法：

去掉重复的画图函数。因为 `moveTo()`和 `lineTo()`两个函数用了两次，第一次为用户画线，第二个默认画。

5. 存在的问题：

自建工具栏不能激活。

解决的方法：

添加消息响应函数和代码。

6. 存在的问题：

工具栏需要两列并排纵放。

解决的方法：

两列工具栏均纵列放置，然后排序即可。

7. 存在的问题：

窗口只要进行最大化最小化，线条消失，有其他窗口覆盖，覆盖处线条消失。

解决的方法：

在画图函数 `onDraw` 中添加重绘函数。如上章的重绘函数所示。

8. 存在的问题：

定时器加入存在问题，总在获取力度参数时提供错误，而其他函数调用力度参数时并没有问题。

解决的方法：

因为定时器在窗口创建时建立，而力度参数的获取需要在界面中完成触摸屏 API 的调用，因此不能成功，需在定时器的创建函数中创建界面实例，然后调用，从而获得力的参数。具体见上章定时器详述。

9. 存在的问题：

画第二条线的起笔点，总是和前一条线的抬笔点保持一致。

解决的方法：

之所以造成，是因为前一次绘画抬笔时力的值没有改变，而后一次的按压力度值的测量又是滞后的，因此会有重复。但添加了定时器后，可以间隔很短的时间即时获取力度，并显示。

10. 存在的问题：

保存图片或保存绘画区内容至内存中时，有两种方法，一种是采用保存点，即记录每条线中点的信息，当需要显示的时候，后台程序根据点的信息再连成线。另一种是采用保存位图的方式，当需要显示的时候，后台程序直接读取位图信息并显示到界面中来。

解决的方法：

选择后一种保存位图信息的方法。

因为前者需要保存大量的信息，画曲线不像直线，直线只须记录始末两点的信息即可，而曲线需要不断地记录信息，费时费力不科学。而位图此时方便写入和读取。

7.2 本章小结

本章大致介绍了在进行毕业设计过程中，进行基于三维触摸屏素描板的开发中，遇到的十个重要的问题以及解决方法。本章描述了一个软件开发的探索，显示了程序编辑中的不易，也是一种劳动成果的记录，这些问题也可为以后的编程或学习者提供思路和帮助。

第八章 结论

本章对整个文章以及毕业设计的课题进行总结，得出研究的结论以及这次课题的意义。最后提出对下一步工作的展望，以更加完善这种技术在计算机领域的发展情况。

8.1 研究结论

在计算机如此普及的今天，普通地处理数据已经不能满足大众的需求了。计算机的功能正在被努力挖掘，并且逐渐深入。各种软件层出不穷，各种技术接踵而至。计算机已经从自我运行的工具转变为与人类交流的工具。因此人机交互技术正在如日中天，触摸屏技术，声控技术，红外技术等等，都充分体现了人机交互的奇妙世界。这次的毕业设计就是基于三维触摸屏进行的素描板软件的开发。用户可以通过接触屏幕的绘画区域，选择笔尖的宽度，通过按压的力度，来体现线条的深浅，从而完成素描板的绘画创作。

本文针对这次的课题“基于三维触摸屏的素描板软件开发”，探讨了该素描板软件的需求分析（功能及非功能），设计模块（新建文件，编辑文件，打开文件以及保存文件），实现代码以及在开发过程遇到的问题以及解决方法。从而对触摸屏和 MFC 编程进行了进一步地详细介绍和探索。通过该程序与触摸屏计算机进行交互的用户可以是自娱自乐的使用者，也可以是教学授课的教师，更可作为敏捷建模和企业会议的记录工具。

8.2 进一步工作

第一、添加绘画场景。本次试验由于时间有限，虽然完成了基本的功能，并且添加了一些窗口界面的背景，但还可以在界面上进行其他实现。譬如，绘画场景的选择，绘画是一个需要灵感的事情，绘画场景可以提供给用户各种场景，激发灵感和创建一个良好的创作感觉。用户可以对绘画环境进行选择，比如下雪场景，有雪花在屏幕中飘落；或者火焰场景，屏幕变黑，角落有一团火焰跳动。再比如柳树场景，柳树的枝条在屏幕中不断摆动。

第二、开发毛笔画板。在研究该触摸屏素描板软件时，意外地做出了类似毛笔的效果，但是不是精准地毛笔感觉。下一步可以开发毛笔画板，也是黑白基调，通过力度、粗细、深浅来体现中国国画水墨风格。只不过需要在笔迹上进行处理，需要有种渲染和断续的效果。

第三、添加其他颜色。既然黑白可以实现，那么其他颜色也可以实现，彩色素描板也是一个不错的选择。提供各种彩色铅笔的选择，更适用于幼儿教育。

第四、添加渐淡效果。这次试验很希望可以添加一个渐淡工具，因为在实际美术中的素描绘画中，很多情况是绘画者用深色画好整个画后，用橡皮蘸需要变浅的地方，以突显光亮和层次感。

参考文献

- [1] 吉锐触摸(GENERAL TOUCH). GeneralTouch API Function（说明书）
- [2] Yellow. 李进久老师的《MFC 深入浅出》电子书（2005 年 2 月 17 日）
- [3] L3 工作室. 《Visual C++/MFC 入门教程》（2004 年 9 月 6 日）
- [4] 百度百科（人机交互，双缓冲）

致 谢

衷心感谢郭文明指导老师的谆谆教导，从做毕业设计到做人，都受益匪浅。

衷心感谢给予很多关照和帮助的软件学院的研究生姐姐，李良辰。

衷心感谢组内其他软件学院的研究生学长学姐，感谢他们的热心帮助和指导。

衷心感谢组内其他同学，感谢大家在一起开会的美好生活。

附 录

附录 1 计划与执行

需要在三个月中完成触摸屏素描板的基本功能，如果时间充裕，条件允许，还可增加一些美化或其他功能。

初步的计划与执行情况，如下表所示：

日期	计划完成内容	实际完成情况
1~2 周	1、分析项目的主要内容 2、书写开题报告；	1、完成 2、完成
3~4 周	1、熟悉三维触摸屏接口程序；	依据实际情况改为： 1、熟悉和钻研 MFC 画图后台编程。 2、查阅相关教程、文档、例子、程序等。 3、学习 MFC 界面编程。
5~6 周	1、分析素描板的需求 2、设计电子素描版主要功能， （用力大，笔画粗，用力小， 笔画细）；	依据实际情况改为： 1、完成普通电脑上的 MFC 画图程序，实现绘画曲线的功能。（基本实现，还有个别问题存在）’
7~8 周	1、产生一个基本的素描板样本，实现其最基本的功能 2、电子素描板可以捕获和记录触摸点的(x,y,z)，经过软件处理，将画笔的动作记录下来 3、撰写中期检查报告；	1、完成 2、完成（基本实现，还有个别问题存在） 3、完成 依据实际情况改动： （2）熟悉触摸屏 API （3）通过对实际素描的观察和探索，将“用力大，笔画粗；用力小，笔画细”的功能变为“用力大，颜色深，用力小，颜色浅”。 （4）将笔画的粗细设为定值，如同绘画用的

		<p>“HB”、“2B”铅笔。</p> <p>用户可以根据需要进行选择。</p>
9~10 周	1、进一步完善程序 2、使素描板具备复制、粘贴、保存功能；	1、完成 2、依据实际情况改为： <ul style="list-style-type: none"> （2） 将固定笔尖宽度的铅笔改为自定义宽度。 （3） 增设使宽度变为最细的快捷按钮 （4） 增加橡皮功能 （5） 使用力大小到颜色深浅的映射可以手动定义。 （6） 实现新建,保存和打开功能。
11~13 周	1、进行测试 2、改进不足	1、完成 2、依据实际情况改为： <ul style="list-style-type: none"> （2） 重新启用九根固定的铅笔,并保留自定义画笔。 （3） 安排工具栏界面 （4） 美化对话框界面 （5） 去掉不能在预期计划中实现的附加功能。 （6） 写帮助文档
14~15 周	1、工作整理 2、论文撰写	1、完成 2、完成