第十一章 事件驱动的程序设计

11.1 事件驱动程序设计概述

Windows是一种基于图形界面的多任务操作系统。

- 系统中可以同时运行多个程序。
- 每个程序通过各自的“窗口”与用户进行交互。
- 用户通过鼠标的单击/双击/拖放、菜单选择以及键盘输入来与程序进行交互。

Windows的功能以两种方式提供：

- 工具（程序）：资源管理器、记事本、画图、……，供用户（人）使用。
- 函数库：以C语言函数形式出现（在windows.h等头文件中申明），作为应用程序接口（API），供Windows应用程序（运行Windows系统上的应用程序）使用。

Windows应用程序的类型

- 单文档应用
  - 只能对一个文档的数据进行操作。
  - 必须首先等当前文档的所有操作结束之后，才能进行下一个文档的操作。
- 多文档应用
  - 同时可以对多个文档的数据进行操作。
  - 不必等到一个文档的所有操作结束，就可以对其它文档进行操作，对不同文档的操作是在不同的子窗口中进行的。
- 对话框应用
  - 以对话框的形式操作一个文档数据。
  - 对文档数据的操作以各种“控件”来实现。
  - 程序以按<确定>或<取消>等按钮来结束。

事件驱动的程序结构

Windows应用程序采用的是一种事件（消息）驱动的交互式流程控制结构：

- 程序的任何一个动作都是由某个事件激发的。
- 事件可以是用户的键盘、鼠标、菜单等操作。

每个事件都会向应用程序发送一些**消息**

每个应用程序都有一个**消息队列**

- Windows系统会把属于各个应用程序的消息放入各自的消息队列。
- 应用程序不断地从自己的消息队列中获取消息并处理之。

“取消息-处理消息”的过程称为**消息循环**

- 当取到某个特定消息（如：WM_QUIT）后，消息循环结束。

每个窗口都有一个消息处理函数。

- 大部分的消息都关联到某个窗口。
- 应用程序取到消息后将会去调用相应窗口的消息处理函数。

基于Windows API的事件驱动程序设计（过程式）

1. **主函数**

   每个Windows应用程序都必须提供一个主函数WinMain，其主要功能是：

   - 注册窗口类（定义程序中要创建的窗口类型）：
     - 窗口的基本风格、消息处理函数、图标、光标、背景颜色以及菜单等。
     - 每类窗口（不是每个窗口）都需要注册。
   - 根据注册的窗口类创建应用程序的主窗口（程序的其它窗口等到需要时再创建）。
   - 进入消息循环，直到接收到WM_QUIT消息时，消息循环结束。

2. **窗口的消息处理函数**

   负责处理发送到相应窗口的消息

   是一个可再入函数

11.2 面向对象的事件驱动程序设计

11.2.1 Windows应用程序中的对象及微软基础类库

- 窗口对象
  - 显示程序的处理数据。
  - 处理Windows的消息、实现与用户的交互。
  - 窗口对象类之间可以存在继承和组合关系。
- 文档对象
  - 管理在各个窗口中显示和处理的数据。
  - 文档对象与窗口对象之间可以存在关联关系。
- 应用程序对象
  - 管理属于它的窗口对象和文档对象。
  - 实现消息循环。
  - 它与窗口对象及文档对象之间构成了组合关系。

MFC提供的主要类

（1）窗口类

- 基本窗口类（CWnd）
  - 实现窗口的基本功能：一般的消息处理、窗口大小和位置管理、菜单管理、坐标系管理、滚动条管理、剪贴板管理、窗口状态管理、窗口间位置关系管理，等等。
  - 是其它窗口类的基类。
- 框架窗口类：提供对标题栏、菜单栏、工具栏、状态栏以及属于它的子窗口的管理功能。
  CFrameWnd：提供了单文档应用主窗口的基本功能
  CMDIFrameWnd：提供了多文档应用主窗口的基本功能
  CMDIChildWnd：提供了多文档应用子窗口的基本功能
- 视类（CView）
  视窗口（简称视）通常位于单文档应用主窗口（CFrameWnd）和多文档应用子窗口（CMDIChildWnd）的客户区（可显示区）。
  视用于实现程序数据的显示以及操作数据时与用户的交互功能。

（2）文档类

对程序要处理的数据进行管理，包括磁盘文件输入/输出

（3）应用框架类

- 文档模板类
- 应用类（CWinApp）
  提供了对Windows应用程序的各部分进行组合和管理的功能，其中包括实现消息循环等。

……

11.2.2 基于文档-视结构的应用框架

应用框架是一种通用的、可复用的应用程序结构，该结构规定了程序应包含哪些组件以及这些组件之间的关系，它封装了程序处理流程的控制逻辑。通过复用应用框架，使得应用的开发速度更快、质量更高、成本更低。

在一个应用框架中，各个组件以及它们之间的关系是固定的，应用的开发者通过给各组件添加具体的业务代码来实现不同的应用。

“文档 - 视”结构

- 文档：用于存储和管理程序中的数据。
- 视：显示文档数据以及实现对文档数据进行操作时与用户的交互功能。
- 文档与视结合构成了“文档-视”结构，它可以实现：
  - 数据的内部表示形式和数据的外部展现形式相互独立。
  - 一个文档对象可以对应一个或多个视对象，即，对于同一个文档数据可以用不同的方式进行显示和操作。

基于“文档--视”结构的应用框架

在该应用框架中，涉及以下的类：

- 视类：CView
- 文档类：CDocument
- 框架窗口类：CFrameWnd（包括CMDIFrameWnd和CMDIChildWnd）
- 应用类：CWinApp
- 文档模板类：CDocTemplate

其中，文档模板类CDocTemplate是应用框架的核心：实现对文档、视和框架窗口所构成的对象组的创建与管理功能。分为：

- 单文档模板类：CSingleDocTemplate
- 多文档模板类：CMultiDocTemplate

业务代码

应用框架规定了程序各部分之间的交互关系（流程控制），不同应用只需要添加各自的业务代码来完成各自的功能。

不同应用的业务代码（应用相关的代码）主要体现在：

- 菜单的设置与消息处理
- 鼠标、键盘消息的处理

- 文档的内部表示及文件输入/输出（Serialize）
- 视中对文档数据的显示（OnDraw）

为了体现“纯”面向对象特性，在应用向导建立的应用程序中隐藏了主函数WinMain。在隐藏的WinMain中，
首先调用theApp的成员函数InitInstance对应用程序进行初始化；
然后去调用theApp的成员函数Run进入消息循环；
消息循环结束之后，将会调用theApp的成员函数ExitInstance进行程序结束前的一些处理。

在应用向导建立的应用程序中对Windows消息处理函数进行了结构化处理：

- 通过“消息映射”机制把Windows消息与程序中相应类的成员函数关联起来
- 各个消息的处理分别由相应类的一个成员函数来实现。

可以用类向导为应用程序中从MFC派生的类增加/删除成员：

- 处理消息的成员函数（菜单、键盘、鼠标等）
- 基类中可重定义的成员函数
- 新定义数据成员（成员变量）
- 对话框类中与各个“控件”所对应的数据成员