多媒体开发工具 Toolbook实用教程

● 康　宏　编著　● 张　祥　主审

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
URL: http://www.phei.co.cn

多媒体开发工具Toolbook实用教程

多媒体开发工具 Toolbook 实用教程

康宏 编著 张祥 主审

電子工業出版社

内 容 提 要

本书主要讲述了如何使用多媒体应用程序的开发平台 Toolbook 来开发 Windows 下的多媒体应用程序。Toolbook 是当今比较流行的面向对象的多媒体应用程序开发工具之一，同其它开发平台相比，用它开发的多媒体应用程序功能更强大，所花费的开发时间更少。

本书是通过三个部分（理论、应用和实践），从 Toolbook 程序最基本的编写原理到实际应用，穿插大量的实例，深入浅出的讲述了运用 Toolbook 开发多媒体应用程序的技术和技巧，并结合 Toolbook 工具阐述了开发多媒体应用程序的主要注意事项。

书　　名：多媒体开发工具 Toolbook 实用教程
编　　著：康宏
审 校 者：张祥
责任编辑：刘文玲　吴迪（特邀）
印 刷 者：
装 订 者：北京科技印刷厂 印刷
出版发行：电子工业出版社出版、发行
北京市海淀区万寿路 173 信箱　邮编 100036　发行部电话 68214070
URL：http://WWW.phei.co.cn
经　　销：各地新华书店经销
开　　本：787×1092 1/16 印张：11.25 字数：276 千字
版　　次：1997 年 2 月第 1 版　1997 年 2 月第 1 次印刷
印　　数：5000 册
书　　号：ISBN 7-5053-3833-1
TP·1646
定　　价：15.00 元

前　言

Toolbook 是由美国 Asymetrix 公司推出的一种面向对象的多媒体开发工具。该软件具有功能强大、容易掌握等特点。

Toolbook 是目前最流行的多媒体开发工具之一，同其它著名的开发工具 Authorware 和 Director 一样都是主流产品，它最大的特点是在 Windows 的集成环境下进行开发工作，开发者可以直接切入到用户层观看制作效果。Toolbook 本身除了具有内置的强大的面向对象的制作工具外，还能够直接支持动态数据链接(DOE)、对象的嵌入和链接(OLE)、对所有 Windows 的 API 库函数的链接和使用。由于 Toolbook 具有面向对象的编程结构和编程语言，因此，既使是没有编程经验的人，也可以在短时间内学习并掌握 Toolbook。

本书以最新版本 Toolbook 4.0 为基础，介绍了 Toolbook 程序设计的基本结构和思想，并附以实例向读者全面介绍了 Toolbook 开发多媒体软件的基本设计思路，从而可以熟练使用 Toolbook 进行实际开发工作。

本书由中国科学院计算所二室康宏、张祥、马郁、刘永波和初登平等同志编写，张祥主审了此书。

作者

1996.7.

序　言

本书共分为三个部分，它们分别是“理论篇”，“应用篇”和“实战篇”。本书的宗旨是帮助那些希望用多媒体软件开发平台，开发自己的多媒体应用程序的用户，在较短的时间内理解ToolBook的设计思想，掌握一定的技能，制作出能够令你满意的多媒体程序。之所以写这本书，并非为了把你推上“多媒体制作大师”的宝座，ToolBook的确具有强大的功能，但它充其量只是一种工具，正如同画家手中的笔与你手中的笔可能完全相同，可你却不能绘制出真正具有艺术价值的绘画作品来。

在阅读这本书之前，我还要劝告各位读者“不要轻易涉入多媒体制作”中，如果你没有完全准备好一颗能面对困难而不轻易破碎的心之前。请先把被绚丽多彩的多媒体世界所冲昏的头脑冷静一下。你会遇到各种困难，例如：界面设计，程序调试，对象安排，材料组织等。在美丽的多媒体世界背后是创作人员艰辛的劳动，工具只是用来提高你的工作效率，决不能把工具所提供的功能堆砌一番就称之为一个“多媒体Title”。我希望你能把更多的时间放在设计和思考上。

要学习制作多媒体，对你的要求很简单。首先你要具备一个正常人所具备的思维逻辑推理方式和一个正常人所具备的审美情趣（如果您的审美情趣已经达到了大众所自叹不如的地步，我奉劝您还是不要加入到这个行列中）；其次就是一点点程序员的思维方法；一点点美术工作者的眼光（至少你应该能分辨出鲜花放在家里的哪一个角落更能衬托出居室的温馨）；最后也是最重要的是要有一点点的耐心。可能你会觉得自己是世界上最有耐心的人之一，但当你在多次推翻自己的设计思想、程序不能通过而又苦于找不到错在哪里、甚至于在含辛茹苦的“创作”出一个作品后只换得了旁人的嗤鼻一笑时，你仍然能耸耸肩说“没关系”，才是真的有耐心。如果你已经满足了以上的条件，就请加入到多媒体制作的队伍中来。告诉你们一个制作多媒体的秘密，真正的快乐并非来自与第一次顺利完成一个作品的那一刻，而是来自于在历经痛苦和磨难后最终所获得的一丝安慰和喜悦。

笔者之所以选择“ToolBook”作为题目，是经过了一定的比较后决定的。相比较与其它的多媒体开发工具，例如：Authorware和Director等，ToolBook的确有许多优点，但其它的开发工具也有它们的长处。关键在于你怎样充分发挥某中开发工具的特点，根据不同的开发要求，选择不同的开发工具。

目　录

理　论　篇

每一门学科都有它的理论，何为 ToolBook 的理论，我在这里暂且把它定义为“组成 ToolBook 应用程序并使它能够正常运行的内在规则”。游戏有规则，更何况是多媒体开发工具呢？有许多多媒体开发工具的内在规则比较少，当然也比较容易掌握，但你所了解的规则越少，你所能“玩”出的花样也就越少。掌握 ToolBook 不算太复杂的理论，可以使你真正有一种创造者的感觉，这对于不愿拘泥于其它人思想的开发者来说却是很重要的。学习它，掌握它，并创造你自己的多媒体世界，如果你真的有这样的信心，那就耐心的学完理论部分吧！

1. ToolBook 的结构

一个家庭的组织结构，一个公司的组织结构，直至一个国家的组织结构都是保证这个家庭、公司和国家正常运转的基本条件。ToolBook 应用程序也是遵循了这个概念，把整个应用程序划分为多个层次和部分组成的总体框架。你在利用 ToolBook 创建多媒体应用程序的第一个工作往往也就是要搭出一个总体框架。暂时抛开应用程序中涉及的具体的模块部分，就 ToolBook 本身而言，最主要的结构如图 1－1 所示。

图 1-1　ToolBook 的结构图

“对象”(Object)：是 ToolBook 应用程序中的元素，如同公司里的每一个员工，没有他们所组成的群体，就不可能有一个健康运作的公司。ToolBook 中的每一种对象都极其具有人格的可比性，也就是说，每一个对象如同每一个人，都有他们各自的特长。例如：域(Field)对象是用来处理字符的，在多媒体中占有较大比重的解释、说明等文字都是建立在“域”的基础上。而

“剧场”(Stage)对象就是用来处理多媒体部分的动画。一个绘声绘色的多媒体应用程序缺少了动画是比较可惜的(但并非一定需要)。对象之间的不同不仅体现在它们的作用上,也体现在它们所具有的“属性”(Property)上。

“属性”也是 ToolBook 中的一个重要概念,简单而言,属性如同每一个员工的特长,有的人会使用电脑,有的人懂英语。属性和作用不同,作用决定了该对象所承担的功能(如同员工所承担的工作),而属性则决定了该对象在承担这种功能时所具备的一些特性。例如:“域”对象具有“文本”(text)属性,这种属性决定了在这个“域”对象中出现的字符。你可以用以下的语句来定义一个叫“解释”的“域”中的文本为“中华人民共和国”

text of fileld “解释” = “中华人民共和国”

虽然大部分情况下,你可以在 ToolBook 中直接输入文字,或用 Windows 的剪贴板,把整段的文字拷贝到域中,但无论你采用哪一种方法,你所输入的文字都是这个域的“文本”(text)属性的内容。“文本”属性是“域”所具有的,而按钮(Button)对象则不具有如上的“文本”属性。也就是说,按钮并不是用来处理文本的。如同一个担任会计职务的员工,他就必须具备处理会计事物的能力,而一个管理计算机的工作人员就不必知道怎样核算企业的成本。在编制 ToolBook 应用程序时,针对不同的目的,选择具备不同属性的对象也是一项最基本的任务。

但是不同的对象也可以具备相同的属性,例如:“域”和“按钮”对象都有“轮廓颜色”(StrokeColor)属性,这种属性决定了它们的外框的颜色。如同对于程序员和外贸员,懂得英语都是一个必备的条件。ToolBooK 中的对象并不多,一共十二种。它们分别是:

“书”(Book)	“背景”(Background)	“页”(Page)
“视窗”(Viewer)	“组”(Group)	“域”(Field)
“记录域”(RecordField)	“按钮”(Button)	“热字”(Hotword)
“组合框”(Combo Box)	“图形对象”(Graphic objects)	“链接和插入”(OLE)对象

每一类的对象都具有它的属性,正是这十二种对象,加上它们各自的属性,组成了变化多端的 ToolBook 的应用程序。值得一提的是几乎所有的对象都具有“脚本”(Script)属性,正是这种属性,才真正赋予这些对象以生命力,即它们对应用程序中其他对象的控制及接受其它对象控制的能力。

“组”(Group)

在这里,我把“组”定义成具有一致运动和功能特性的对象群。首先肯定“组”本身也是一种对象,它虽然不具有其它对象所具有的一些特定的功能性的属性和作用,但它对于组织对象的行为和功能上却起了重要的作用。类似于企业中的小组是通过集体配合来完成特定的工作一样,在一个“组”中的所有对象既保持了它们各自的特点,又具有共同的特性。例如:当你用程序或鼠标拖动由一个“图形”和一个“域”所组成的组时,这两个对象将同时移动,并在移动过程中保持它们之间的相对位置不变。以上的例子只说明了组的一个特点,其它的重要特点还有“组”的“脚本”,你如果在“组”的“脚本”中定义了处理一定消息的程序,则当该消息被“组”所

截获后，你就可以很方便的同时改变整个组中成员的属性(例如是否隐藏或显示这个“组”及“组”中所有的对象)。“组”在 ToolBook 应用程序中并不直接具有特定功能，却又是不可缺少的。例如：你要隐藏一个有三个域“记录 1”、“记录 2”和“记录 3”组成的组“记录”，你可以用以下的指令来把所有的对象一次都隐藏起来：

```
hide group “记录”
```

否则你就要逐个的隐藏三个域

```
hide field “记录 1”
hide field “记录 2”
hide field “记录 3”
```

“页”(Page)

之所以把这个多媒体开发工具称为 ToolBook(工具书)，可能正是出自于“页”这个概念。“页”同日常生活中书的页具有很大的类似，你在 ToolBook 中查询或学习知识，就如同在翻阅一本书，ToolBook 的组织结构就是基于这样的概念：“在某一个特定时间，在一个特定页上显示特定的信息”。

首先，“页”也是一个独立的对象，即使你没有创建任何“对象”(例如：“域”或“按钮”)，你也可以创建一个完全由空白的页所组成的 ToolBook 应用程序(可能没有人愿意为这样的应用程序掏钱)。在创建一个 ToolBook 应用程序时，ToolBook 系统就自动的为你添加了一个空白的“页”和“背景”，也就是说“页”其实是其它对象的归属，如同某一特定车间里的工人在许多关系上都是隶属于这个车间。现在，你们就可以理解，在 ToolBook 中，虽然存在十二种对象，但在它们之间不但具有功能上的差别，还具有级别上的区别。“页”的级别就高于某些对象(例如：“域”或“按钮”)，某一特定“页”上的对象同其它“页”上的对象是互相独立的，举一个简单的例子，在“A”页上有两个按钮分别叫“开始”和“结束”，在“B”页上也有一个按钮叫“开始”，在处理与按钮有关的程序时，就要使用按钮的名字：

```
visible of button “开始” = false          ——隐藏称为“开始”的按钮
```

这时你可能会疑惑，这个“开始”按钮到底是指哪一个页上的，解决这个问题的方法是在程序中，所有的对象后面都附加上对特定页的指代。例如：

```
visible of button “开始” of page “A” = false      ——隐藏“A”页上称为“开始”的按钮
```

这样，接受命令的按钮就一目了然了，但也会产生问题——“代码的通用性”，你可能发现这句程序只能用来处理页“A”上的按钮，而对于程序员而言，编写具有通用性的程序往往是他们的一个重要工作和责任。所以你可以再重写以上程序：

```
visible of button “开始” of this page = false      ——隐藏当前页上称为“开始”的按钮
```

这样,你所发出的命令只针对当前页上的按钮"开始",即使你在其它页上定义了成千上万个叫"开始"的按钮,ToolBook 也可以把命令准确的送到你所希望的对象上。以上的例子说明不同页上的对象是可以被区分的,即使它们使用相同的名字(但是,同一页上的对象,特别是同一类对象就应使用不同的名字来加以区分)。你所创建的对象,都是基于一个特定的"页",一旦"翻"到另一"页",这些对象也就不可见了;一旦该"页"被撤消(确切的说是被删除),你所创建的对象也就同时消失了(注意:不可见是指这些对象实际存在,但暂时不受程序命令的影响,而消失是指你永远的失去了它们)。

背景(Background)

"背景"的确是一个很确切的名称,背景上的对象是不受页的改变而改变。如同一出话剧中,对于每一个不同情节场景,可能有不同的布景,在舞台上的道具可能会不断的改变,但布景上的道具是不改变的。一个"背景"也是一个对象,但它的级别高于"页",所有的"页"都隶属于特定的背景。在"背景"上也可以象在"页"上一样创建特定的各种对象。但所有在"背景"上创建的对象都会出现在隶属于它的页上。这样你就可以把一些通用的对象和功能放在"背景"上,例如控制翻到前一页或后一页的控制按钮就可以放在背景上。对于程序员来说,这样的确减轻了很大的工作量。在创建一个 ToolBook 应用程序时,ToolBook 系统就自动的为你添加了一个空白的"页"和"背景",可以基于这个"背景"添加任意多的"页",也可以再增加一个新的空白"背景"和隶属于新"背景"的"页"。不能删除任何"背景",除非把隶属于他的页全部删除,当你删除了隶属于该"背景"的所有"页"后,该"背景"也就自动被删除了。

视窗(Viewer)

由于页是组成 ToolBook 的主要单位,那么用什么办法才可以在 ToolBook 应用程序中同时显示多个页呢? 使用视窗。如同它的名字一样,视窗只是一些可以定义的窗口。你可以自始至终只使用一个视窗(Main Window),这个视窗是不可以关闭的,当创建一个应用程序的时候就同时创建了。你也可以创建任意多的视窗,灵活使用。例如可以把所有演示部分都放在一个特定的视窗内,而只在主视窗内安排文字和装饰图形。

书(Book)

如果抛开"背景",把所有的"页"串接起来就形成了"书",如同日常生活中的书一样。一本 ToolBook"书"就是一个具有特定作用的应用程序。可以把整个应用程序编制成一本书,也可以按照各个不同的适用范围,划分并编制出不同的书,但同属于一个应用程序,并且可以在一本"书"中调用其它的"书"。好比一个大公司的庞大机构,可以由几个具体部门组成,而一个小公司只需要一个精简的机构就可以了。

ToolBook 系统

这不是由你来控制的一个部分。因为,你所创建的 ToolBook 应用程序正是基于这样的 ToolBook 系统,无论在编制应用程序的过程中或是在执行程序的过程中,都要使用 ToolBook 系统具体解释你的程序,控制各种对象的运作。

ToolBook 资源(Resource)

它是构成 ToolBook 的一类元素,这类对象不但属于 ToolBook 系统,更重要的是它还可以以

独立的形式存储在 ToolBook 系统外，成为通用的部件。有了它们，可以提高 ToolBook 应用程序的外观和内涵。例如：有了光标资源，可以提示用户当前的状态、有了调色板资源，可以使所显示的位图色彩更加逼真。总而言之，“资源”添加了 ToolBook 应用程序的连贯性和紧密性，并且还能够提高创作效率，在创作过程中的一些值得骄傲，并可能在以后被反复使用的东西，就可以资源的形式单独存储，这样，你既可以在以后的创作中重复使用它们，也可以很慷慨的把它们送给你的朋友使用。在 ToolBook 中，资源一共有五种，它们分别是：调色板、位图、光标、菜单条和图标。

ToolBook 作者层(Author Level)和读者层(Reader Level)

作者层和读者层的最大区别在于：前者是作者用的，后者是用户用的。由于无论你在开发或运行 ToolBook 的应用程序的过程中，都要基于 ToolBook 系统，所以使用这样的层次定义只是为了让创作者有一点自尊心而已。你可以在作者层创建和定义各种对象，然后立即在读者层测试它们的功能，唯一在它们之间切换的手续只是按一下 F3。可以通过定义，把在作者层的权利赋予用户，也可以使用你的特权，限制读者层的权利。对于其它的作者，还可以以各种手段把他们封锁在你的应用程序的作者层之外。简而言之，作者层是创作的园地，而读者层是用户真正要面对的。

综上所述，ToolBook 应用程序的整体可以视为：以对象为基础，以页为组织单位，以背景为线索，以视窗为显示方法，以书为运行单位，以资源为复用手段，以作者层和读者层作为界线的一种多媒体开发工具。

2. ToolBook 的事件和消息

事件(Event)

“面向对象，事件驱动”，这几个字，大家可能经常看到，Windows 本身就是一个面向对象的操作系统。什么是对象？姑且把上一部分所定义的对象做为这里的对象，例如：一个“按钮”或一个“域”。而事件就是外界对 ToolBook 应用程序的影响，套用生物学的话，也就是一种“刺激”。用户在打开你编制的应用程序后干的第一件事是什么，你不可能知道；你只知道，用户可能干的事情，例如：打开菜单或调节音量等。在编制程序时就要针对用户的行为而设计。在 ToolBook 中，用户所做的事情，例如单击按钮或输入文字等就是所谓的事件(当然不包括他干一些和计算机无关的事情)。事件本身并没有大的区别，也就是说，无论单击一个按钮还是单击一个图形，用户所做的事件都是“击键”，只不过针对的对象不同罢了。ToolBook 包括各种各样的事件，区别这些事件的就是这种事件所触发的“消息”。

消息(Message)

当所谓“事件驱动”的应用程序被事件“刺激”后，ToolBook 系统的首要任务就是发出同这个事件相配合“消息”。这种消息好比你同友人通电话是在电线中传输的电信号，信号本身只包含了一些特定的含义，而对于这些特定含义的处理还要依靠程序。ToolBook 应用程序的消息，严格的可以分为两大类，一类就是 ToolBook 系统提供的，例如“击键”(ButtonClick)消息或“拖曳”(Drag and Drop)消息，当操作特定的事件时，这一类消息就会自动的产生并进行传递。

而另一类消息是由用户自己创立的，把这种消息同一定的事件联系在一起，并强迫 ToolBook 系统发送。消息其实是联系用户和应用程序的一根纽带。对于消息的处理主要有两个方面，一个是在脚本中所编写的针对特定对象的处理程序，另一个就是 ToolBook 系统本身所定义的处理程序。

3. ToolBook 的对象层次

对象层次(Object Hierarchy)

这是一个比较抽象的概念。其实，简单的讲就是消息在传递过程中所经过的路径。这种路径并非可以自己选择，而是由 ToolBook 系统所决定的。你可以决定的只是在哪里处理和终止这个消息。就象一个员工所写的报告，可能交到他上级那里就可以得到及时处理，于是报告的使命也就完成了。如果报告内容涉及公司存亡，那么就会向上传送直至总经理的面前。在 ToolBook 应用程序中，第一个接到消息的就是引起该消息的用户事件所针对的某一个 ToolBook 对象。例如：用户单击一个按钮，“ButtonClick”消息就会由 ToolBook 系统传递到该按钮上，如果该按钮中有处理这个消息的程序(Handler)，该消息就会被自动截获，并及时处理，然后该消息自动取消。如果该按钮中不包含处理这个消息的程序(Handler)，该消息就会沿着“对象层次”自动的向上传递，直至被处理或到达 ToolBook 系统。ToolBook 中的消息传递的层次如图 1-2 所示：

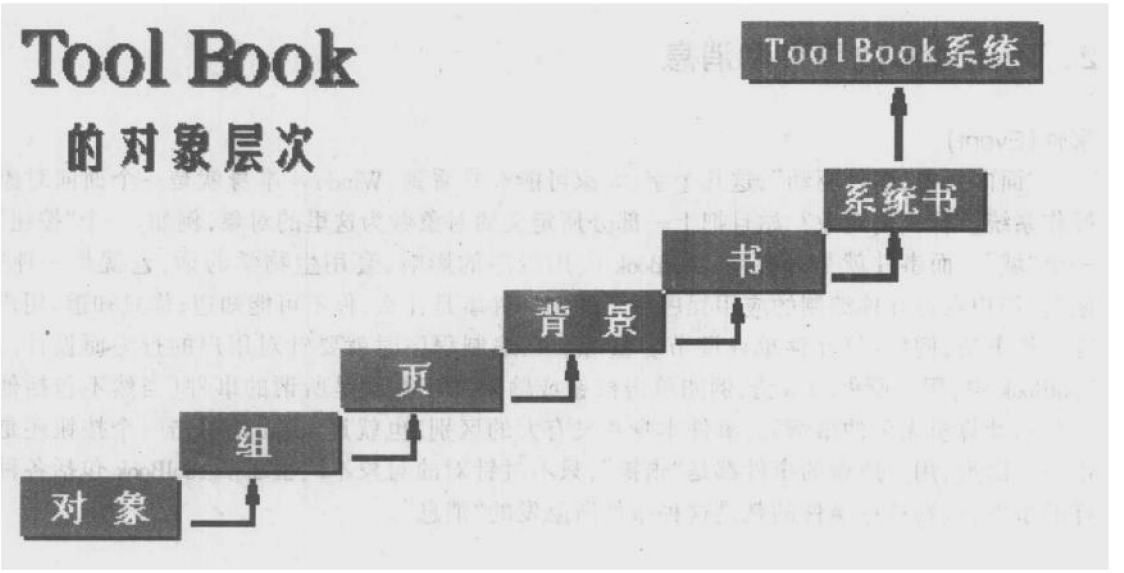

图 1-2　ToolBook 的对象层次

当然 ToolBook 消息在对象层次的具体传递过程中，并非一定要按部就班，也就是说，虽然在层次关系上由图 1-2 所决定，但如果缺少其中的某一个环节，消息依然沿剩余的层次向上传递。例如：如果被单击的按钮并没有隶属于任何一个组，那“ButtonClick”消息就会沿着层次，直接传递到该按钮所隶属的页上。一旦消息发送到某一个层次，ToolBook 就检查该层上具体对象的“脚本”(Script)属性中的处理程序中是否包含有处理这个消息的指令。如果有，该消息会

被就地处理，并自动消除；如果没有，消息就会被发送到当前对象的上一个层次的对象上去。

所以，在 ToolBook 层次上的每一个具体对象，并非都会收到最底层发送的消息。如果你希望消息在被处理后仍能够向上传递，就要使用 Forward(向前)语句，把该消息再发出一次。不过这一次，消息并非从底层发送，而是从使用该语句的对象上发出，所以接受到这个消息的对象中，不可能包括比发送消息的对象所处层次较低的对象了。

有了层次的概念，一些过去必须由程序员指定的关系，变成了 ToolBook 中内置的关系，遵循这种层次关系，就可以减少程序员的许多工作量，也使事件、消息和对象之间的关系更加清晰了。

4. ToolBook 的语言

ToolBook 的功能很强大！

之所以有这样的感慨，完全是因为 ToolBook 所使用的程序语言。同其它多媒体开发工具所使用的语言相比较，它简直可以说是一种“高级语言”了。除了提供丰富的内置函数外，还包含了流行的模块化语言所具备的语言结构，并且包含了函数和过程的概念，以及数组和参数的传递等。可以象使用其它语言(例如：C 或 C++)一样的编制程序，当然它不可能具有直接处理底层的能力，但通过调用 Windows 的库函数，也可以解决这个问题。你能分析简单的英语语法吗？如果你的回答是“是”，那你就可以用 ToolBook 所提供的 OpenScript 语言进行编程。找不到一个恰当的翻译方法，只能暂时用它的原文，若一定要翻，就把它定义成“开放型脚本”。这里的“开放”可能是指它那种与英语极其接近的语法。以下面的程序为例：

```
To Handle ButtonClick            —— 开始处理单击消息

    visible of paintobject "elephant" of this background = false
                                 —— 当前页的背景上的图形对象“elephant”的 Visible 属性的值为 False

    send next                    —— 发送“next”消息，翻到下一页
End ButtonClick                  ——结束处理单击消息
```

每一条语句都遵循一定的英语语法规范，您可以很快的用这种语言编程了。但是，“快”并非可以一蹴而就，学习一定的句法结构和语句是在所难免的。

5. 总结

理论篇到这里就算结束了，一共讲述了 ToolBook 的结构、消息、时间、层次和语言等问题，在结构部分，又对 ToolBook 中最主要的几个概念：对象、页、背景等进行了解释。对于这些概念和理论的具体实现和运用，将在以后的应用篇中讲述。如果你对理论篇中所提及的概念有不懂的地方，您可以有两种解决的方法，一种是把书翻到第一页，重新读一便，另一种方法就是带着这些疑问，继续看下去，“在游泳中学会游泳”！

应 用 篇

在这一部分，主要讲述 ToolBook 所提供的各种工具和他们的使用方法。“工欲善其事，必先利其器”，掌握这些界面，菜单条和命令的使用，是学习使用 ToolBook 的第一步，而且是你所必须牢固掌握的。第二步就是学习用 OpenScript 语言编写程序，这一部分，可以根据个人的情况，选择有用的部分学习。通过这一部分的学习，使你在使用 ToolBook 的过程中，进一步理解 ToolBook 的概念。第三部分就是学习如何管理和使用 ToolBook 的资源，以及如何利用这些资源创建多媒体应用程序。

1. ToolBook 的界面

ToolBook 的界面丰富多彩，甚至可以用“复杂”来形容了。其中最有特色的是它所提供的多种界面工具。这些界面工具使你能够更加简便快捷的操作，使命令更加容易被理解，加快用户编制应用程序的速度，当然只限于那些“老”用户。对于新用户，从菜单开始，仍然是他们的最佳选择。在 ToolBook 的作者层(创作区域)，除了传统的菜单条以外，还有其它六种辅助型工具。它们分别是工具条(toolbar)、状态条(status bar)、辅助设计板(palette)、尺(ruler)、网格(Grid)和右单击菜单(rightclick button)。下面先从菜单开始。

1.1 ToolBook 的菜单

一个标准的 Windows 应用程序往往拥有一个菜单，菜单上是系统中可以使用的菜单项，这已经成为了一种固定的模式，ToolBook 也不例外。ToolBook 不仅在自己的编程环境中包含菜单，也可以用它所提供的工具在你自己的应用程序中创建自定义的菜单。

1.1.1 菜单的组成和类型

通常所说的菜单，其实是指一个菜单条(Menu Bar)。菜单条是由菜单(例如：文件(File)或编辑(Edit))和菜单项(例如：打开(Open)或拷贝(Copy))组成的。当创建一个菜单条时，它的信息就以菜单资源的形式储存在 *.MNU 资源文件中。可以把菜单资源赋于任何视窗，其中包括 ToolBook 的主窗口(启动 ToolBook 新书时所显示的窗口)。当打开一本书时，它就包含了内置的作者层菜单和读者层菜单，它们都是菜单资源(menu Bar ID 100)。

每一个菜单条上的菜单项都有一个名字和别名。菜单项的名字就是直接显示在菜单条上的文本，可以通过菜单项的名字来推测菜单项的作用，例如：“存储”(Save)菜单项，代表可以用单击该菜单项存储当前的 ToolBook 书。菜单项除了一个显式的名字外，还有一个隐式的名字。这个隐式名字从不显示在菜单上，但可以用来在脚本程序中指代菜单项。之所以使用隐式的名字，是为了使应用程序更加容易维护。