"九五"国家重点电子出版物规划项目.计算机知识普及系列

Borland C++ Builder

高级编程技术

北京希望电脑公司　总策划
龙启铭　编　著

北京希望电脑公司

Borland C++ Builder 高级编程技术

北京希望电脑公司　总策划

龙启铭　编著

谢建勋等　审校

北京希望电脑公司　出品

1998

内 容 简 介

Borland C ++ Builder 美国 Borland 公司最新推出的快速程序开发（RAD）工具，它具有 RAD 环境下的 C ++全部功能，其重要特点是有很强的数据库和网络应用程序开发能力，而且能快速、简单地实现。

本书的重点是应用程序开发，尤其是数据库和网络应用程序的开发。全书由十一章构成，主要内容包括 Borland C ++ Builder 集成开发环境、事件响应、Borland C ++ Builder 与 Delphi 共享代码、图形图像、动态链接库 DLL、数据库开发、网络数据库开发、组件创建、Internet 应用程序开发、利用 QuickReport 组件创建报表和发布应用程序。

本书既是从事 Borland C ++ Builder 应用和开发的所有人员的参考书，同时也可作为大专院校相关专业师生的自学和教学读物。

需要本书和配套电子书或需技术支持的读者可直接与北京海淀 8721 信箱书刊部联系，电话：010-62562329，62531267，或传真：010-62579874，62633308 联系。

“九五”国家重点电子出版物规划项目.计算机知识普及系列

Borland C ++ Builder 高级编程技术

北京希望电脑公司　总策划

龙启铭　编著

谢建勋等　审校

责任编辑　战晓雷

北京希望电脑公司　出品

北京海淀路 82 号（100080）

北京东升印刷厂　印刷

新华书店、新华书店音像发行所、各地书店、软件专卖店经销

* * * * *

1998 年 9 月第 1 版 . 1998 年 9 月第 1 次印刷

开本：787×1092 1/16　印张：13.75

字数：320 千字　　印数：1-5000

新出音管[1997] 96 号

ISBN 7-980008-40-5/TP • 11

定价：30.00 元（1CD，含配套书）

前　言

C++ Builder 是 Borland 公司的快速应用程序开发(RAD)工具，它具有 RAD 环境下的 C++ 全部功能。其一个重要特点是有很强的数据库和网络应用程序开发能力，而且能快速、简单地实现。

本书重点是应用程序的开发，尤其是数据库和网络应用程序的开发，而不是 C++语言的讲解，所以要求读者有一定的 C、C++语言基础。

本书所有的程序均在 C++ Builder 3.0 环境下调试通过。

龙启铭

1998 年 6 月

目　录

第1章　C++ Builder

C++ Builder 是Borland公司最新系列产品之一，它是为C++应用程序而开发的快速应用开发(RAD)产品。使用C++ Builder创建Win32应用程序时，具有RAD环境下的C++全部功能。可以快速地为应用程序创建用户界面（如主视窗、菜单和对话框等），还可以快速地创建各种实际应用的程序。Borland公司最新系列产品（如Delphi和J Builder等）一个主要的特点就是有很强的数据库开发能力，因此它们都是数据库开发的首选。现在的应用程序开发，几乎80%包含数据库。数据库的掌握与开发显得越来越重要，数据库开发人员的需求也越来越大。

本书的读者要求有一定的C、C++语言基础，因为本书的重点不是语言讲解，而是应用程序的开发。从后面的内容可以看出，在应用程序的开发中，尤其以数据库的开发为重点。

1.1　C++ Builder集成开发环境

运行C++ Builder，出现C++ Builder IDE（集成开发环境）。如图1-1所示，包含有主窗口菜单、属性与事件浏览器、组件调色板、一个空白表单以及速度条等内容。

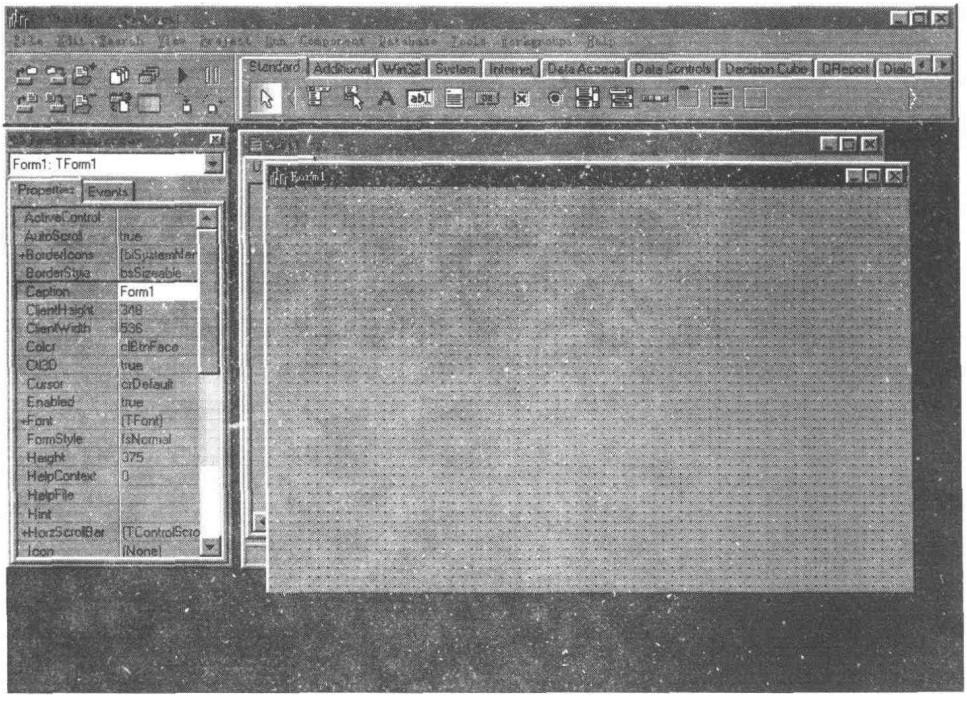

图1-1　C++ Builder集成环境

1.2 第一个应用程序

传统的编程都是从编写一个“Hello World”程序开始。但这对于C++ Builder来说，不用编写一行代码就能实现，我并不想从它开始，因为这样就显示不出C++ Builder的方便、快速了。我的第一个程序是多媒体播放器，可以播放.wav、.mid和.avi文件，另外，还可以调入.bmp文件来更换窗口的底图。应用程序运行结果如图1-2所示。

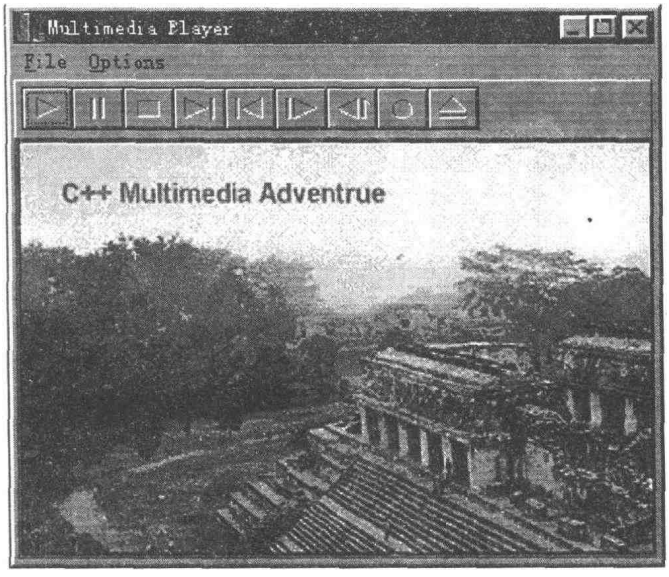

图1-2 运行结果

下面是项目文件的建立步骤：

1. 选取File->New...,弹出对话框图1-3，选取Application，生成一个新项目；

2. 从Standard组件调色板选取Panel控件，在窗体上生成一个TPanel对象，缺省名为Panel1，将它放置窗体的顶部。从System组件调色板选取MediaPlayer控件，在窗体上生成TMediaPlayer对象，缺省名为MediaPlayer1使它位于Panel1上。从Additional组件调色板选取Image控件，生成一个TImage对象，缺省名Image1。在Image1的Picture属性中装入所要的图形文件，用作背景图案。从Dialogs组件调色板选取OpenDialog控件，生成一个TOpenDialog对象，缺省名OpenDialog1。

3. 主菜单制作

从Standard组件调色板选取MainMenu控件，生成一个TMainMenu对象，缺省名MainMenu1。在窗体上双击MainMenu1图标，弹出菜单编辑器，如图1-4所示。

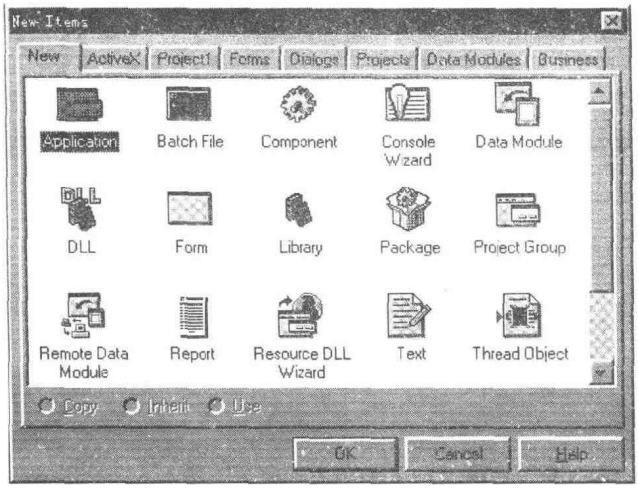

图1-3 新建项目对话框

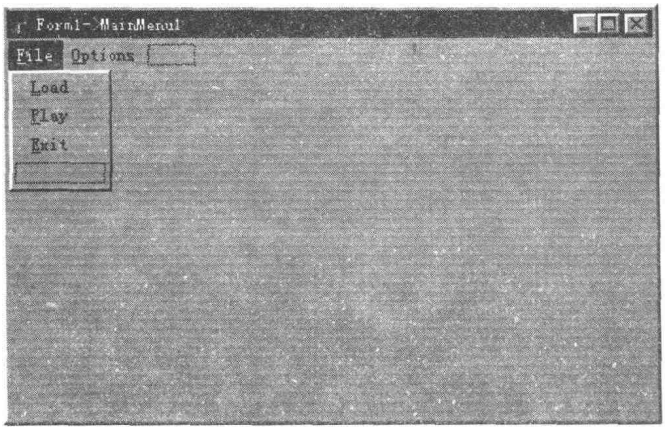

图1-4 菜单编辑器

设计两个菜单项File和Options，其中File下有Load、Play和Exit三个菜单选项，Options有ChangeBackGrounds一个菜单选项。

```
TMenuItem *File1;     // Popup menuTMenuItem *Load1;   // menu item
TMenuItem *Play1;     // menu item
TMenuItem *N1;        // (separator made by entering
                      //a dash in the Caption field)
TMenuItem *Exit1;      // menu item
TMenuItem *Options1;                  // Popup menu
TMenuItem *ChangeBackground1;   // menu item
```

4．设置主窗口Form1的Memu属性为MainMenu1。设置OpenDialog1的InitialDir属性为“C:\”，Filter属性为“Movies,Sound,Midil*.avi,*.wav,*.mid”。

5．给菜单项添加代码

单击主菜单MainMenu1的File->Load菜单选项，弹出代码编辑器，为Load菜单项加入如下代码：

```
void_fastcall TForm1::Load1Click(TObject *Sender)
{
if(OpenDialog1->Execute( ))
    {
    MediaPlayer1->FileName=OpenDialog1->FileName;
      //将对话框的文件赋给媒体播放器
    MediaPlayer1->Open( );
      //打开媒体播放器，使播放器置于播放状态
      //至于媒体播放的播放、暂停等功能，由控件自己实现，
      //无须编写任何代码
    }
}
```

给File->Play菜单选项加入代码：

```
void_fastcall TForm1::Play1Click(TObject *Sender)
{
try
    {
    MediaPlayer1->Play( );
      //调用媒体播放器的播放函数
    }
catch(EMCIDeviceError &E)
    {
    AnsiString S("\r Use the File|Open Menu item to select an AVI File.");
    ShowMessage("Bummer:"+E.Message+"."+S);
    }
}
```

给File->Exit菜单选项加入代码：

```
void_fastcall TForm1::Exit1Click(TObject *Sender)
{
Close( );
        //关闭应用程序
}
```

给Opitions->ChangeBackGrounds菜单选项加入代码：

```
void_fastcall TForm1::ChangeBackGrounds1Click(TObject *Sender)
```

```
{
    AnsiString RootDir(ParamStr(0));
    AnsiString SaveDir=OpenDialog1->InitialDir;
    AnsiString SaveFilter=OpenDialog1->Filter;
    OpenDialog1->InitialDir=ExtractFilePath(RootDir);
    OpenDialog1->Filter="Picture|*.bmp";
if (OpenDialog1->Execute( ))
   {
    Image1->Picture->LoadFromFile(OpenDialog1->FileName);
      //装入图片，设置为播放器背景图
    Image1->Stretch=True;
    }
OpenDialog1->InitialDir=SaveDir;
OpenDialog1->Filter=SaveFilter;
}
```

6．好了，所有工作都完成了，运行应用程序，即可进行.avi、.wav和.mid文件的播放，并可以改变播放器的背景图案。

要用其它C++语言设计实现上面的功能，要比这复杂得多。从这个程序可以看出C++ Builder的快速应用开发（RAD）优点。

下面保存所做的工作，将主窗体保存为mainfrm.cpp，将项目文件保存为player.bpr。另外C++ Builder还为自动保存了一个窗体头文件mainfrm.h。

为了进一步了解C++ Builder的工作，以及它与一般的C++语言的异同点，让我们来看看C++ Builder 为我们生成的代码。其中所有的关键处都有注释，请读者自己体会C++ Builder一般的C++语言异同。其它认为读者能理解的地方就不一一注释，因为本书的读者是要有一定的C++编程基础。

主窗口头文件mainfrm.h源代码

```
//////////////////////////
//mianfrm.h
//////////////////////////////
#ifndef mainfrmH
#define mainfrmH
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Dialogs.hpp>
```

```
#include <ExtCtrls.hpp>
#include <Menus.hpp>
#include <MPlayer.hpp>

class TForm1 : public TForm    //定义主窗体类
{
_published:    // IDE-managed Components
    //定义窗体中包含的控件对象
    TPanel *Panel1;
    TMediaPlayer *MediaPlayer1;
    TOpenDialog *OpenDialog1;
    TMainMenu *MainMenu1;
    TMenuItem *File1;
    TMenuItem *Load1;
    TMenuItem *Play1;
    TMenuItem *Exit1;
    TMenuItem *Options1;
    TMenuItem *ChangeBackGrounds1;
 TImage *Image1;

     //函数定义
     void_fastcall Load1Click(TObject *Sender);
     void_fastcall Play1Click(TObject *Sender);

     void_fastcall ChangeBackGrounds1Click(TObject *Sender);
  void_fastcall Exit1Click(TObject *Sender);

  private:  // User declarations

  public:          // User declarations

     _fastcall TForm1(TComponent* Owner);
  };
  extern PACKAGE TForm1 *Form1;

  #endif
```

主窗口源程序mainfrm.cpp代码

```
//////////////////////////
//mainfrm.h
//////////////////////////
#include <vcl.h>
#pragma hdrstop
```

```
#include "mainfrm.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

_fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}

void_fastcall TForm1::Load1Click(TObject *Sender)
{
if(OpenDialog1->Execute( ))
    {
    MediaPlayer1->FileName=OpenDialog1->FileName;
    MediaPlayer1->Open( );
    }
}

void_fastcall TForm1::Play1Click(TObject *Sender)
{
try
    {
    MediaPlayer1->Play( );
    }
catch(EMCIDeviceError &E)
    {
    AnsiString S("\r Use the File|Open Menu item to select an AVI File.");
    ShowMessage("Bummer:"+E.Message+"."+S);
    }
}

void_fastcall TForm1::ChangeBackGrounds1Click(TObject *Sender)
{
AnsiString RootDir(ParamStr(0));
AnsiString SaveDir=OpenDialog1->InitialDir;
AnsiString SaveFilter=OpenDialog1->Filter;
OpenDialog1->InitialDir=ExtractFilePath(RootDir);
OpenDialog1->Filter="Picture|*.bmp";

if (OpenDialog1->Execute( ))
    {
```

```
        Image1->Picture->LoadFromFile(OpenDialog1->FileName);
        Image1->Stretch=True;
        }
    OpenDialog1->InitialDir=SaveDir;
    OpenDialog1->Filter=SaveFilter;
    }
    void_fastcall TForm1::Exit1Click(TObject *Sender)
    {
    Close( );
    }
```

上面的函数都是为菜单选项的响应代码，前面已有注释，就不再罗嗦了。

项目文件player.cpp源代码

```
//////////////////////////
//player.cpp
//////////////////////////
#include <vcl.h>
#pragma hdrstop
USERES("mainfrm.res");
USEFORM("mainfrm.cpp", Form1);
//应用程序的WinMain函数，与其它C++的WinMain函数作用与结构都类似
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
    try
    {
        Application->Initialize( );
        Application->CreateForm(_classid(TForm1), &Form1);
        Application->Run( );
    }
    catch (Exception &exception)
    {
        Application->ShowException(&exception);
    }
    return 0;
}
```

从上面的代码可以看出，C++ Builder建立好了应用程序的总体框架，开发人员只要用C++语言编写所要实现的功能的代码，而不必去考虑其它重复性的工作。重复性的工作都交给了C++ Builder。

第2章　事 件 响 应

所谓事件响应就是指某一事件被触发后，将会发生的操作。下面是最常见的公共事件及其描述。

表2-1　一些公共事件

事　件	描　　述
OnChange	当一个控件按某种方式改变时发生的事件
OnClick	单击鼠标时发生的事件
OnDblClick	双击鼠标时发生的事件
OnEnter	被激活时发生的事件
OnExit	失去焦点时发生的事件
OnKeyDown	具有输入焦点时按键按下发生的事件，这些键包括字母、数字键、方向键、Home、End以及Ctrl键等
OnKeyPress	具有输入焦点时按键按下发生的事件，但只包括字母、数字、Tab、Backspace、Enter及Esc等键
OnKeyUp	按键被释放时发生的事件
OnMouseDown	鼠标按下发生的事件
OnMouseMove	鼠标移动发生的事件
OnMouseUp	鼠标释放发生的事件
OnPaint	窗体或控件重画时发生的事件

C++ Builder能够很容易地处理鼠标和键盘事件。

2.1　鼠标与键盘事件响应

下面举一个简单的例子来说明鼠标事件响应。

具体步骤：

1．选取File->New...菜单选项，建立一个新项目文件event1.bpr；

2．在Object Inspector的event页中的OnClick属性双击，弹出代码编辑器，输入下面代码：

```
void _fastcall TForm1::FormClick(TObject *Sender)
{
  MessageDlg("The Mouse is Clicked.", mtInformation,
             TMsgDlgButtons( ) << mbOK, 0);
}
```

每次鼠标在窗体上单击，弹出如图2-1所示对话框。

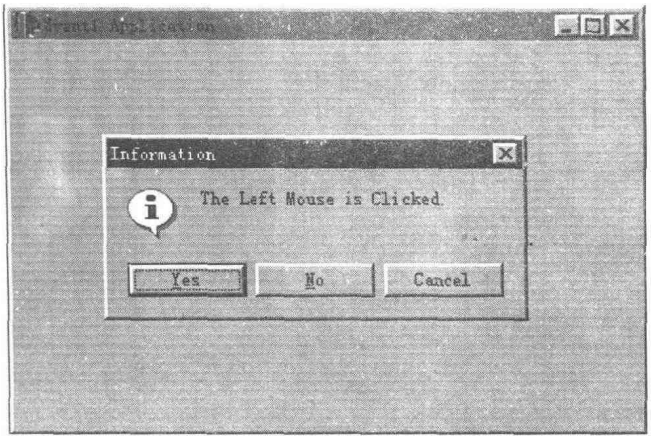

图2-1 鼠标事件响应

3. 在窗体的OnMouseDown属性右边双击，弹出代码编辑器后加入代码：

```
void _fastcall TForm1::FormMouseDown(TObject *Sender,
    TMouseButton Button,TShiftState Shift, int X, int Y)
{
  if (Shift.Contains(ssRight))
  {
    Canvas->Brush->Style = bsClear;
    Canvas->TextOut(X, Y, "* Button");
  }
}
```

当鼠标右键在窗体上单击，在鼠标所在位置显示“Button”字符。

以上是几种鼠标事件响应。

同样，C++ Builder可以很方便地处理键盘响应事件。

例如，在窗体的event页的OnKeyDown 属性加入如下代码：

```
void _fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key,
  TShiftState Shift)
{
  MessageDlg(Key, mtInformation, TMsgDlgButtons( ) << mbOK, 0);
}
```

上面代码的意义是：当用户按下一个键时，弹出一个包含所按键的ASCII值的对话框。例如，按下“A”，弹出一个含有“65”的对话框。

当然，也可以弹出包含按键本身字符的对话框，其代码如下：

```
void _fastcall TForm1::FormKeyPress(TObject *Sender, char &Key)
{
```

```
    AnsiString S("OnKeyPress: " + AnsiString(Key));
    MessageDlg(S, mtInformation, TMsgDlgButtons( ) << mbOK, 0);
}
```

2.2 直接处理事件响应

从上面的例子可以看出，你可以不用考虑是这样得到鼠标和键盘等的事件发生，你的注意力是事件发生后，所要做的工作。

C++ Builder也可以让程序员直接得到事件响应消息。例如，对于下面鼠标移动事件消息的处理，C++ Builder是先由VCL得到消息，然后将消息送给OnMouseMove事件，程序员只须将相应代码写入OnMouseMove事件，而不必关心事件消息的得到，只关心事件的处理。下面来看看不通过VCL而直接进行鼠标移动消息事件处理的方法。

首先在窗体类中定义鼠标消息事件：

```
class TForm1 : public TForm
{
_published:
  ... // Declarations omittedprivate:
MESSAGE void MyMouseMove(TWMMouse &Message);
public:
  virtual _fastcall TForm1(TComponent* Owner);
BEGIN_MESSAGE_MAP
  MESSAGE_HANDLER(WM_MOUSEMOVE,
 TWMMouse, MyMouseMove);
END_MESSAGE_MAP(TForm);
};
```

上面代码告诉C++ Builder当鼠标移动时，你可以直接对此事件作反应。

然后，给MyMouseMove函数添加相应的代码：

```
void TForm1::MyMouseMove(TWMMouse &Message)
{
  TForm::Dispatch(&Message);
  LSpecialMouse->Caption = "X " + IntToStr(Message.XPos) +
                           " Y " + IntToStr(Message.YPos);
}
```

代码先是调用TObject类的Dispatch( )函数。如果没有调用此函数，则程序的其它地方就永远得不到鼠标移动消息，但程序还是能运行。