时尚百例丛书

C++ Builder 6.0 时尚编程百例

网冠科技　编著

光盘包含本书素材、效果文件

机械工业出版社

C++ Builder 6.0 是 Borland 公司最新推出的功能强大、兼有 Visual C++ 的灵活性和 Delphi 的易学性的应用程序开发软件。本书以详尽的实例介绍了使用 C++ Builder 6.0 编写 Windows 应用程序的原理和方法，着重引导用户边制作边提高，在最短的时间内掌握 C++ Builder 6.0 的精华。

本书共分 6 篇，由浅入深地讲解了 C++ Builder 6.0 的基础知识、控件应用、多媒体开发技巧、高级开发技巧、网络与数据库程序的开发以及实用程序的开发，相信不同层次的读者都能从中学到需要的知识。

本书可供广大程序员、大学师生、计算机爱好者参考使用。

图书在版编目（CIP）数据

C++ Builder 6.0 时尚编程百例 / 网冠科技编著.
–北京：机械工业出版社，2002.4
（时尚百例丛书）
ISBN 7-111-10132-4

Ⅰ.C … Ⅱ.网… Ⅲ.语言-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2002）第 019605 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
策 划：胡毓坚
责任编辑：王 虹
责任印制：

印刷 · 新华书店北京发行所发行

2002 年 4 月第 1 版 · 第 1 次印刷
787mm × 1092mm 1/16 · 20.25 印张 · 498 千字
0001-6000 册
定价：36.00 元（1CD）

前　言

《C++ Builder 6.0 时尚编程百例》是“时尚百例丛书”中的一本。

在当今数字化的世界里，计算机已经深入到人们生活的各个领域中，应用软件的飞速发展已成为必然，最近崛起了第三、四代编程语言。其中 Inprise（原 Borland 公司）推出的 C++ Builder 是当今最优秀的 C++开发系统。C++ Builder 的最新版本 C++ Builder 6.0 支持许多业界的最新技术，最接近 ISO 的 C++标准。

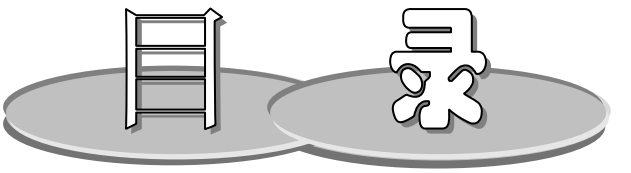
本书以实例的形式向读者讲解如何使用 C++ Builder 6.0。C++ Builder 6.0 是 Borland 公司的最新产品，代码自动完成的速度比以前快了很多；新增了一个 object treeview 窗口，可以方便的管理窗体上的控件；在数据库方面，增加了一个 dbexpress 标签页；在网络应用方面，增加了 websnap 和 datasnap 两个标签页，大大增强了网络应用程序的开发功能。

全书制作了 100 个实例，满足不同层次读者的需要。本书共分为六篇。各篇的内容如下：第一篇为基础篇，主要讲解 C++ Builder 的基本开发步骤，基本的工作流程，以便初学者对 C++ Builder 有一个大概的了解；第二篇为控件应用篇，C++ Builder 6.0 为用户提供了 100 多个控件，每个控件又包含若干个属性、事件和方法，要熟练应用这些控件，需要通过实例制作逐步掌握它们的用法；第三篇为多媒体应用篇，在 PC 时代，多媒体成为一个巨大的市场，多媒体软件有了前所未有的发展，C++ Builder 顺应历史潮流，提供了画布（Canvas）技术、对 Windos 的 MCI 接口的支持以及 OpenGl 编程的支持；第三篇中将详细介绍 C++ Builder 在多媒体方面的强大功能；第四篇为高级应用篇，主要讲解了 Windows API 函数的应用，怎样操纵注册表、编写 dll 文件、制作多线程应用程序以及获得硬件信息等内容；第五篇为网络与数据库篇，现今，在美国有 80%的应用程序使用这样或那样的数据库，事实上，大多数应用程序都是围绕着如何操作数据库的，在一个大型的企业中，为了协调管理，就需要开发网络数据库，以便能够实现异地同步管理，在本篇中以网络和数据库为中心，讨论了使用 C++ Builder 6.0 进行网络数据库开发的各方面的知识；第六篇为综合应用篇，这里以 5 个相对较大的实例讲解了如何综合使用 C++ Builder 的各项知识，编写出实用的应用程序。

本书主要由张增强编写，通过本书 100 个实例的学习，可以提高程序员开发 C++ Builder 6.0 应用程序的效率。真诚希望我们的讲解对读者有所帮助。

网冠科技

本书光盘含配套素材，技术支持请点击网冠科技站点 http://netking.163.com。E-mail：netking_@yeah.net。



第三篇　多媒体应用篇

第四篇　高级应用篇

第五篇　网络与数据库篇

第六篇　综合应用篇

第一篇

基础篇

本篇总览

本篇主要介绍使用 C++ Builder 6.0 的一些基本操作技巧，包括基本控件，如按钮、标签、文本框等的使用；基本的编程方法如定义局部变量、全局变量、调用 C++ Builder 6.0 中各控件的属性、事件、方法等。

本篇通过介绍链接效果的实现、按键检测、控制窗体最大化、右对齐文本、鼠标边框、小日历、计算器、对话框应用、大小金额转换、制作向导窗口、淡入淡出文字、关闭程序前进行提示、动态修改图像大小、Word 风格对话框等实例，读者可以了解使用 C++ Builder 6.0 编程的基本步骤，同时也会对 C++ Builder 6.0 的强大功能有所体会。

实例 1　网页链接效果

实例说明

本例程序使用 RichEdit 控件设置了一个链接效果，如图 1-1 所示。本例中，单击其中的网址：http://166.111.136.3，即可链接到相应的网址，就像 Office 2000 中的 Word 链接一样。

图 1-1　效果图

编程思路

本例使用了一个 Windows API 函数 SendMessage，通过调用该函数的方法可以设置 RichEdit 控件中带有特殊字符的文本，比如 http、www 等，该函数的原型如下：

```
LRESULT SendMessage(
    HWND hWnd,              // handle of destination window
    UINT Msg,               // message to send
    WPARAM wParam,          // first message parameter
    LPARAM lParam           // second message parameter
   );
```

创作步骤

1．启动 C++ Builder 6.0，打开一个新的标准工程。如果 C++ Builder 已经运行，则在 File 菜单中单击 New→Application 菜单项，打开一个新的标准工程。在新建的界面上加入一个 RichEdit 控件和一个 Button 控件。

2．设置 RichEdit 控件的关键属性如下：

Left = 8
Top = 8
Width = 305
Height = 137
Font.Charset = GB2312_CHARSET
Font.Color = clWindowText
Font.Height = -24

```
    Font.Name = 'MS Sans Serif'
    Font.Style = []
    ParentFont = False
    TabOrder = 0
```

3．为了使当用户点击有链接效果的文本时能链接相应的网页，需要添加如下一个处理函数。

```
void __fastcall TMainForm::WndProc(Messages::TMessage &Message)
// 链接到指定网页的处理函数
  {
  if (Message.Msg == WM_NOTIFY)
    {
    if (((LPNMHDR)Message.LParam)->code == EN_LINK)
      {
      ENLINK* p = (ENLINK *)Message.LParam;
      if (p->msg == WM_LBUTTONDOWN)
        {
        SendMessage(RichEdit1->Handle, EM_EXSETSEL, 0, (LPARAM)&(p->chrg));
        ShellExecute(Handle, "open", RichEdit1->SelText.c_str(), 0, 0, SW_SHOWNORMAL);
         //链接到指定的网页
        }
      }
    }
  TForm::WndProc(Message);
  }
```

4．在 Main.h **头文件的中添加代码如下；**

```
#include <vcl.h>
#pragma hdrstop
#include "Main.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TMainForm *MainForm;
__fastcall TMainForm::TMainForm(TComponent* Owner) : TForm(Owner)
  {
  Caption = Application->Title;
  unsigned mask = SendMessage(RichEdit1->Handle, EM_GETEVENTMASK, 0, 0);
  SendMessage(RichEdit1->Handle, EM_SETEVENTMASK, 0, mask | ENM_LINK);
  SendMessage(RichEdit1->Handle, EM_AUTOURLDETECT, true, 0);
  RichEdit1->Text = "欢迎访问酒井网站 "
                    "网址： http://166.111.136.3. ";
  }
```

```
void __fastcall TMainForm::WndProc(Messages::TMessage &Message)
    {
    if (Message.Msg == WM_NOTIFY)
        {
        if (((LPNMHDR)Message.LParam)->code == EN_LINK)
            {
            ENLINK* p = (ENLINK *)Message.LParam;
            if (p->msg == WM_LBUTTONDOWN)
                {
                SendMessage(RichEdit1->Handle, EM_EXSETSEL, 0, (LPARAM)&(p->chrg));
                ShellExecute(Handle, "open", RichEdit1->SelText.c_str(), 0, 0, SW_SHOWNORMAL);
                }
            }
        }
    TForm::WndProc(Message);
    }
void __fastcall TMainForm::CloseBtnClick(TObject *Sender)
    {
    Close();
    }
```

5．单击工具栏中的运行按钮，或者按下 F9 键，运行程序，将会出现如图 1-1 所示的效果。

说明：本书中的其他实例运行方法与本例相同，后面不再讲解。

实例 2　按 键 监 测

实例说明

本例程序可以实现对用户按下的键盘或鼠标键的监测。当用户按下<Shift>键时，在状态栏回显示<Shift>字样；同理，按下<Alt>键、<Ctrl>键、单击或双击鼠标时，也会显示相应的信息，如图 2-1 所示。

该实例集中介绍了响应 C++ Builder 中各种常用事件的方法。

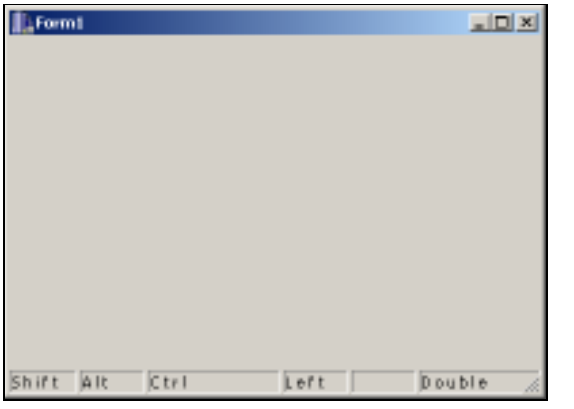


图 2-1　效果图

编程思路

C++ Builder 编写的程序都是事件驱动程序，即程序并不是按照流程依次运行各行代码，而是根据用户的交互响应来执行不同的代码。这样可以提高程序的执行效率和节省内存空间。

本实例中使用了几个基本的事件响应：

```
void __fastcall TForm1::FormKeyUp(TObject *Sender, WORD &Key,
        TShiftState Shift)
void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key,
        TShiftState Shift)
void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button,
        TShiftState Shift, int X, int Y)
```

每个响应在代码中都有详细说明，这里不再介绍。

创作步骤

1．启动 C++ Builder 6.0，打开一个新的标准工程。如果 C++ Builder 已经运行，则在 File 菜单中单击 New→Application 菜单项，打开一个新的标准工程。在新建的界面上加入一个 StatusBar。

2．双击 StatusBar 控件，出现如图 2-2 所示的 Editing StatusBar 对话框，单击其中的 Add New 按钮，可以向其中添加新的分栏，单击六次 Add New 按钮，在其中建立六个分栏，最后状态栏如图 2-1 所示。

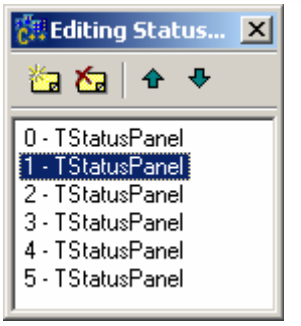


图 2-2　Editing StatusBar 对话框

3．本程序源代码如下：

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
if( Shift.Contains(ssShift) )                    // 如果按下了<Shift>键则在第一个面板上显示 Shift
StatusBar1->Panels->Items[0]->Text = "Shift";
if( Shift.Contains(ssAlt) )                      // 如果按下了<Alt>键则在第二个面板上显示 Alt
StatusBar1->Panels->Items[1]->Text = "Alt";
if( Shift.Contains(ssCtrl) )                     // 如果按下了<Ctrl>键则在第三个面板上显示 Ctrl
StatusBar1->Panels->Items[2]->Text = "Ctrl";
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FormKeyUp(TObject *Sender, WORD &Key,
      TShiftState Shift)
```

```
{
// 在<Shift>、<Alt>和<Ctrl>键弹起时清除状态栏中相应面板上的内容
if( !(Shift.Contains(ssShift)) )
StatusBar1->Panels->Items[0]->Text = " ";
if( !(Shift.Contains(ssAlt)) )
StatusBar1->Panels->Items[1]->Text = " ";
if( !(Shift.Contains(ssCtrl)) )
StatusBar1->Panels->Items[2]->Text =" ";
}
// ---------------------------------------------------------------------------
void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button,
      TShiftState Shift, int X, int Y)
{
if( Shift.Contains(ssLeft) )                    // 如果按下了鼠标左键则在第四个面板上显示 left
StatusBar1->Panels->Items[3]->Text = "Left";
if( Shift.Contains(ssMiddle) )                  // 如果按下了鼠标中键则在第五个面板上显示 Middle
StatusBar1->Panels->Items[4]->Text = "Middle";
if( Shift.Contains(ssDouble) )                  // 如果是双击鼠标则在第六个面板上显示 Double
StatusBar1->Panels->Items[5]->Text = "Double";
}
// ---------------------------------------------------------------------------
```

实例 3　控制窗体的最大化

实例说明

本例程序可以控制窗体最大化的程度，如图 3-1 所示，当窗体最大化时，窗体并没有占满整个屏幕，只达到 300×200 像素；同时，当用户拖动窗体时，也不能将窗体扩展到 300×200 像素以上。

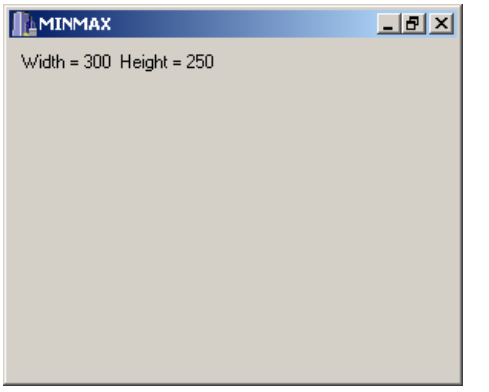


图 3-1　效果图

编程思路

要实现控制窗体的大小，关键是截获 Windows 的 TWMGetMinMaxInfo 信息，在 C++ Builder 中，对 TWMGetMinMaxInfo 的定义如下：

```
struct TWMGetMinMaxInfo
{
 unsigned Msg;
 int Unused;
 tagMINMAXINFO *MinMaxInfo;
 int Result;
} ;
```

在这个构造函数中，可以对窗体的最大化、最小化窗口等属性进行设置。

创作步骤

1．启动 C++ Builder 6.0，打开一个新的标准工程。如果 C++ Builder 已经运行，则在 File 菜单中单击 New→Application 菜单项，打开一个新的标准工程，向其中添加一个 Label 控件。

2．设置 Form 窗体的关键属性如下：

Left = 244	Top = 221
Width = 225	Height = 185
Caption = 'MINMAX'	Color = clBtnFace
Font.Charset = DEFAULT_CHARSET	Font.Color = clWindowText

Font.Height = -11　　Font.Name = 'MS Sans Serif'

Font.Style = []　　OldCreateOrder = True

OnResize = FormResize　　PixelsPerInch = 96

TextHeight = 13

3．**设置** Label **控件的关键属性如下：**

Left = 8　　Top = 8

Width = 32　　Height = 13

Caption = 'Label1'

4．**向头文件中添加代码如下：**

```
// ---------------------------------------------------------------------------
#ifndef MAINFORMH
#define MAINFORMH
// ---------------------------------------------------------------------------
#include <vcl\Classes.hpp>
#include <vcl\Controls.hpp>
#include <vcl\StdCtrls.hpp>
#include <vcl\Forms.hpp>
// ---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:                                    // IDE-managed Components
   TLabel *Label1;
   void __fastcall FormResize(TObject *Sender);
private:                                        // User declarations
   void __fastcall WMGetMinMaxInfo(TWMGetMinMaxInfo &Msg);   // prototype for msg handler
public:                                         // User declarations
   __fastcall TForm1(TComponent* Owner);
BEGIN_MESSAGE_MAP
   MESSAGE_HANDLER(WM_GETMINMAXINFO,TWMGetMinMaxInfo,WMGetMinMaxInfo)
END_MESSAGE_MAP(TForm)
};
//---------------------------------------------------------------------------
extern TForm1 *Form1;
// ---------------------------------------------------------------------------
#endif
```

5．**向单元文件中添加代码如下：**

```
// ---------------------------------------------------------------------------
#include <vcl\vcl.h>
#pragma hdrstop
```

```
#include "MAINFORM.h"
// ---------------------------------------------------------------------------
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::WMGetMinMaxInfo(TWMGetMinMaxInfo &Msg)
{
    Msg.MinMaxInfo->ptMaxSize.x=300;                // 设置最大化时的窗口水平方向大小
    Msg.MinMaxInfo->ptMaxSize.y=250;                // 设置最大化时的窗口竖直方向大小
    Msg.MinMaxInfo->ptMaxPosition.x=GetSystemMetrics(SM_CXSCREEN)-300;
                                                    // 设置窗体最大化时的 x 方向位置
    Msg.MinMaxInfo->ptMaxPosition.y=0;              // 设置窗体最大化时的 y 方向位置
    Msg.MinMaxInfo->ptMaxTrackSize.x=300;           // 设置拉伸最大化的窗口水平方向大小
    Msg.MinMaxInfo->ptMaxTrackSize.y=250;           // 设置拉伸最大化的窗口竖直方向大小
    Msg.MinMaxInfo->ptMinTrackSize.x=170;           // 设置拉伸最小化的窗口水平方向大小
    Msg.MinMaxInfo->ptMinTrackSize.y=150;           // 设置拉伸最小化的窗口竖直方向大小
}
void __fastcall TForm1::FormResize(TObject *Sender)
{
    Label1->Caption = "Width = "        + AnsiString(Width)    +
                                "    Height = " + AnsiString(Height);
}
// ---------------------------------------------------------------------------
```