计算机编程技巧

ABC 系列丛书

C++ Builder 编程技巧

多媒体与系统篇

清宏计算机工作室 编著

机械工业出版社
China Machine Press

.P312C
188

计算机编程技巧 ABC 系列丛书

C++ Builder 编程技巧

（多媒体与系统篇）

清宏计算机工作室　编著

机 械 工 业 出 版 社

本书以详尽的实例、丰富的内容深入系统地介绍了 C++ Builder 5.0 在多媒体和系统方面的编程技巧与方法。针对每一个技巧的主题，分关键所在（或原理方法）、实现与应用以及专家点评三个部分进行讲解，力求在内容讲解清楚的前提下，篇幅短小，使读者在有限的篇幅下学到更多技巧。本书按章节编写，各章由既相对独立，又相互关联的技巧主题组成，并且在顺序编排上考虑了内容的前后连续性。

本书适合已具有 C++ Builder 基本知识的读者学习，可供广大计算机编程人员、大专院校师生、计算机爱好者和各种培训班学员参考使用。

机械工业出版社（北京市百万庄大街 22 号　邮政编码 100037）
责任编辑：边　萌　　封面设计：姚　毅
责任印制：郭景龙
三河市宏达印刷厂印刷・新华书店北京发行所发行
2001 年 1 月第 1 版第 1 次印刷
787mm×1092mm 1/16・21.75 张・ 534 千字
0 001－5000 册
定价：39.00 元 （1CD，含配套书）
ISBN7-900043-45-4/TP・41
凡购本书，如有缺页、倒页、脱页，由本社发行部调换
本社购书热线电话（010）68993821、68326677-2527

前　言

随着计算机技术的飞速发展，计算机的开发迅速得到普及推广。当前有一大批计算机开发人员，虽然已经掌握了某种开发工具的基础知识、甚至达到了相当高的水平，但仍有很多盲区，或者急需更多的编程经验和技巧。为此，我们决定组织编写这套“计算机编程技巧 ABC 系列丛书”，以满足广大读者的需求，帮助读者快速成为计算机编程得行家里手。

这套丛书针对当前最流行的几种开发语言，介绍开发过程中的编程经验和技巧，并解决开发过程中的疑点和难点，主要突出“技巧”二字。针对某些开发语言，根据其内容相关性和独立性，我们编写了两个系列的书，即“网络与数据库篇”和“多媒体与系统篇”。这样可以使读者有更大的选择余地，并且在内容编排、读者阅读上都更为合理。

每本书按章节编写，各章由既相对独立，又相互关联的技巧主题组成，并且在顺序编排上考虑了内容的前后连续性。

针对每一个技巧主题，我们将其分为“ABC”三个部分进行讲解。

A——关键所在（或原理方法）　以简单明了的语言指出实现技巧的关键所在或原理方法。

B——实现与应用　以一个简单、完整实例向读者介绍技巧的实现方法或步骤。

C——专家点评　对技巧加以适当说明，包括补充解释，强调注意事项等。比如简要介绍实现该技巧的其他方法，或者介绍关键技术的扩展应用等。

针对每一个技巧主题，力求在内容讲解清楚的前提下，篇幅短小，使读者在有限篇幅内学到更多的技巧，整套书充分体现强调短小精悍、高效率的特点。

读者在阅读本套丛书之前，应该对相应开发语言有了一定了解，但并不要求读者开发水平一定很高，在章节编排和内容讲解过程中，我们都尽量照顾了水平还比较低的读者，因此初学者还是可以阅读这套丛书的。

希望这套丛书能使你很快成为软件开发的高手。

恳请广大读者对书中不足之处提出宝贵意见。

清宏计算机工作室

编 者 的 话

图形图像和多媒体技术是当今计算机行业中发展最为迅速、最具有吸引力的领域之一，也是二次开发平台最为注重的、不可缺少的部分。同时一个应用程序只运行在具有同样一种配置的系统上的情况是很少见的，大多数应用程序将运行在具有不同硬件和软件配置的系统上，几乎每个实用的系统都需要在运行期间获取关于系统的信息。本书将分八章，并以实例的方式介绍 C++ Builder 中有关多媒体和系统编程方面的技巧。

第 1 章“TCanvas 和图像控件的使用”首先介绍了 TCanvas。TCanvas 封装了应用程序在图形图像输出方面所需要的大多数 GDI 对象和绘图函数，使得图形图像的处理工作变得异常简单。然后介绍了图像处理的技巧，包括旋转位图、柔化图像、锐化图像、浮雕图像、扩散图像等，并介绍了如何使用其他一些图像控件，扩充 C++ Builder 提供的图像处理控件的功能。最后介绍如何利用 TImage 控件实现读、写文件到显示图像、操作像素等一系列过程。

第 2 章“应用 GDI 函数”介绍了如何运用 GDI 函数来处理图像控件不能直接处理的问题。诸如通过 TCanvas 提供的 Handle 属性访问设备描述表、映像模式、调色板处理、区域等一些高级话题。

第 3 章“使用 OpenGL”详细介绍了在 C++ Builder 中使用 OpenGL 绘制具有彩色及光照图形的基本内容、图形的特殊效果处理以及由位图生成三维图的方法等内容。

第 4 章“使用 DirectX”主要介绍了在 C++ Builder 中如何利用 DirectDraw 技术快速绘制高性能的 Window 图形，以及如何利用 Direct3D 发挥 Windows 3D 的编程功能和利用 DirectSound 实现播放声音功能。

第 5 章“多媒体”介绍了如何在应用程序中实现播放 MIDI、WAVE 等声音文件和 AVI 视频文件，以及动画制作等内容。

第 6 章“系统与硬件”介绍了如何检测并使用计算机上配置的硬件，包括内存、硬盘、显示器、打印机和游戏杆等。

第 7 章“系统与环境”介绍了如何在 C++ Builder 中使用获取关于系统配置、环境等信息的函数，包括在应用程序中启动另一个程序、关闭 Windows 系统、查找、移动和删除文件、保证只有程序的一个实例在运行，以及屏蔽系统热键和任务栏等。

第 8 章“系统安全与多线程”介绍了如何在使用多线程的同时又满足系统安全性的问题。

本书不但对 C++ Builder 中图形图像、多媒体和系统方面做了全面和详细的介绍，而且通过大量的实例和技巧，结合作者实际开发工作中的经验，对读者可能遇到的一些问题进行了深入透彻的分析，希望能给读者带来一些既切合实际又有用的知识。

本书的配套光盘中含有书中的全部源程序，这些程序都是经过严格调试和测试的。

由于编者时间关系和水平有限，书中难免出错，希望读者谅解和批评指正。

此为试读，需要完整PDF请访问：www.ertongbook.com

目　录

第 1 章　TCanvas 和图像控件的使用

1.1　巧用 TCanvas 的画线方法

关键所在

TCanvas 的方法较多，可以大致将它们分为四种类型：画线的方法、画填充图形的方法、文本输出方法和图形拷贝方法。Canvas 提供了强大的画线功能，利用 Canvas 的画线方法可以绘制直线、椭圆的弧线、Bezier 曲线以及组合线等。

绘制直线涉及到两个方法：MoveTo 和 LineTo。MoveTo(int startx, int starty)的任务是设置当前画笔的位置到(startx ,starty)，而不进行任何绘图工作。然后可以调用 LineTo（int endx , int endy）来画直线，它从当前画笔的位置画一条直线到点（endx ,endy），并将当前的画笔位置改变为(endx ,endy)。利用 MoveTo 和 LineTo 函数，通过一些简单的数学计算，就可以画出各种各样的优美图形。

实现与应用

选择菜单 File | New Application，创建一个新的项目文件。在对象监视器（Object Inspector）中双击 Form1 的 OnPaint 事件，创建此事件处理函数，在此函数中加入如下代码。注意在 Unit1.cpp 的最前面加入#include <math.h>一句代码，用于包含 math.h 的头文件。

```
//---------------------------------------------------------------------------
void __fastcall TForm1::FormPaint(TObject *Sender)
{
 double A,x1,y1,x2,y2;
 int D=100;
 double E;
 Width = 640;
 Height = 480;
 //设置画笔颜色
 Canvas->Pen->Color = clTeal;
 //设置画刷样式
 Canvas->Brush->Style = bsClear;
 //循环画直线
 for (int i=0; i<720; i++)
   {
```

```
        A = i*3.1415/360;
        E = D*(1+sin(4*A));
        x1 = 320+E*cos(A);
        x2 = 320+E*cos(A+M_PI/5);
        y1 = 240+E*sin(A);
        y2 = 240+E*sin(A+M_PI/5);
        Canvas->MoveTo(x1,y1);
        Canvas->LineTo(x2,y2);
      }
  }
  //---------------------------------------------------------------------------
```

编译并运行程序，就可以看到如图 1-1 所示的优美图形。

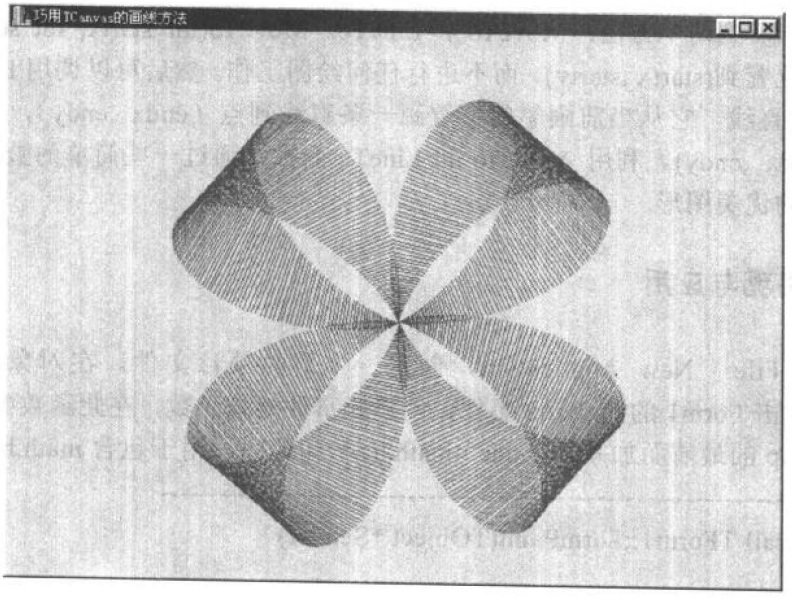

图 1-1 实现优美图形

专家点评

程序中使用了两个非常简单的函数：MoveTo 和 LineTo，加上了适当的运算，却绘制了另人心动的图形，可见程序的设计主要在于程序员的创新思维。有兴趣的读者不妨试一试下面一段程序代码，同样只利用了 MoveTo、LineTo 两个函数和相同的绘图机理，绘制的图形却又不一样。

```
//---------------------------------------------------------------------------
void __fastcall TForm1::FormPaint(TObject *Sender)
{
```

```
double A,x1,y1,x2,y2;
int D=150,E=50;
double L,M,O,P;
Canvas->Pen->Color = clTeal;
Canvas->Brush->Style = bsClear;
for (int i=0; i<240; i++)
   {
    A = i*M_PI/120;
    L = D+D/3*(1+cos(12*A)/2)*cos(A);
    x1 = 240+1.25*L*cos(A);
    M = E+E/3*(1+sin(12*A)/2)*cos(A);
    x2 = 240+1.25*M*cos(A);
    O = D+D/3*(1+cos(10*A)/2)*sin(A);
    y1 = 240-O*sin(A);
    P = E+E/2*(1+cos(15*A)/2)*sin(A);
    y2 = 240-P*sin(A);
    Canvas->MoveTo(x1,y1);
    Canvas->LineTo(x2,y2);
   }
}
//---------------------------------------------------------------------------
```

1.2　巧用 TCanvas 画椭圆的方法

关键所在

画布的绘制椭圆的方法很简单，形式如下：

Ellipse(int X1, int Y1, int X2, int Y2);

该方法在画布指定的矩形边界上画一个椭圆，（x1，y1）是矩形左上角的像素坐标，（x2，y2）是矩形右下角的像素坐标。如果矩形形成一个区域，将出现一个椭圆。当 x2-x1=y2-y1 时，画出的图形就是一个圆。如果加上适当的算法，也能绘制各种各样的优美图形。

实现与应用

选择菜单 File | New Application，创建一个新的项目文件。在对象监视器（Object Inspector）中双击 Form1 的 OnPaint 事件，创建此事件处理函数，在此函数中加入如下代码。注意在 Unit1.cpp 的最前面加入#include <math.h>一句代码。

此为试读，需要完整PDF请访问：www.ertongbook.com

```
//---------------------------------------------------------------------------
void __fastcall TForm1::FormPaint(TObject *Sender)
{
 double AL,x1,y1,x2,y2;
 int L=120;
 Width = 640;
 Height = 520;
 //设置画笔的颜色
 Canvas->Pen->Color = clTeal;
 //设置画刷的样式
 Canvas->Brush->Style = bsClear;
 //绘制 48 个椭圆
 for (int i=0; i<48; i++)
   {
     AL = i*3.1415/24;
     x1 = L*cos(AL);
     y1 = L*sin(AL);
     x2 = x1+320;
     y2 = -y1+240;
     Canvas->Ellipse(x2-90,y2-90,x2+90,y2+90);
   }
}
//---------------------------------------------------------------------------
```

编译并运行程序，显示效果如图 1-2 所示。

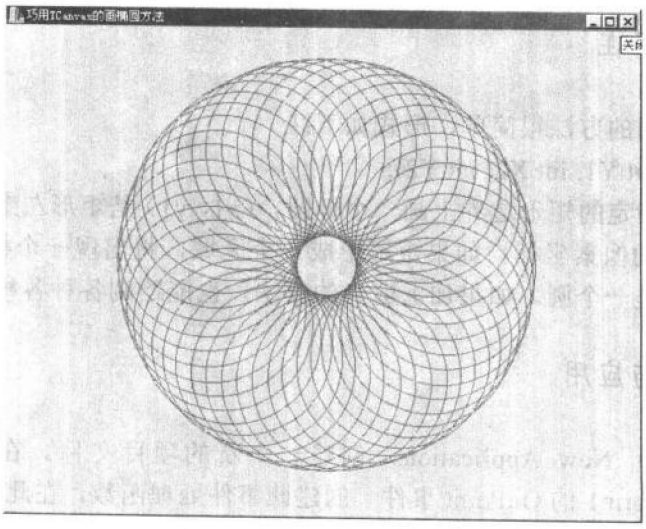

图 1-2 实现画椭圆

专家点评

在这个程序中，只用到了 Ellipse(int X1, int Y1, int X2, int Y2)方法，利用该方法在窗口上连续画出 48 个椭圆，就形成上面的图形。注意要将 Brush 的 Style 设置为 bsClear，这样就会使椭圆内部不会被填充。

细心的读者可能会注意到椭圆实际上被一个矩形所包含，是一个由矩形限定的图形。除了椭圆之外，这类图形还有圆弧、饼状图和圆角矩形，它们分别使用的方法是 Chord、Pie 和 RoundRect。

1.3　获取并使用系统的字体

关键所在

TCanvas 的字体（Font）属性是一个 TFont 对象。在 Windows 程序设计中，字体的处理曾经是一件很令人头痛的事情，在 C++Bulider 中，TFont 封装了 Windows 中的字体属性，使得改变字体的属性变得异常的简单。字体的属性主要有字体颜色（Color）、字体大小（Size）、字体样式（Style）和字体名(Name)等。

C++ Builder 中提供了名为 Screen 的全局变量，利用此变量中的 Fonts 属性就可以获取当前系统中可以使用的字型。

实现与应用

选择菜单 File | New Application，创建一个新的项目文件。在窗体 Form1 上放置一个 TListBox 控件。然后双击 Form1，创建窗体的 OnCreate 事件处理函数，并在其中加入如下代码。

```
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
 for(int i=0; i<Screen->Fonts->Count; i++)
    ListBox1->Items->Add(Screen->Fonts->Strings[i]);
}
//---------------------------------------------------------------------------
```

双击 ListBox1 控件，创建它的 OnClick 事件处理函数，在其中加入如下代码。

```
//---------------------------------------------------------------------------
void __fastcall TForm1::ListBox1Click(TObject *Sender)
{
 int i=ListBox1->ItemIndex;
```

```
 Canvas->Font->Name=ListBox1->Items->Strings[i];
 Canvas->Font->Size=20;
 Canvas->TextOut(0,0,"技巧");
}
//---------------------------------------------------------------------------
```

专家点评

程序在 Form1 的 OnCreate 事件处理函数中调用 Sreen->Fonts，获取系统中预先安装的字体，并显示在 ListBox1 中。当在 ListBox1 中选择了一字体，就将此字体名赋给画布的字体的 Name 属性，并用此字体在窗体的左上角显示“技巧”两字。

1.4 利用 TextOut 方法绘制立体文字

关键所在

TCanvas 最基本的文本输出方法是 TextOut 方法，TextOut 的具体形式如下：

TextOut(int X, int Y, const AnsiString Text);

该方法在指定的位置（X，Y）输出文本。利用该函数，可以非常容易地制作立体文字。制作立体字的基本原理就是在文字下面制作一些阴影，这样文字看起来就有立体感了。具体做法就是利用原来的字体颜色显示生成一字符串，接着再用灰色显示生成同一字符串，然后再将它们稍微错位叠加显示出来，这样就简单地生成了立体字。

实现与应用

选择菜单 File | New Application，创建一新的项目文件。在窗体 Form1 上加入三个 TLabel 控件、一个 TEdit 控件和两个 TTrackBar 控件，TEdit 控件用于设置要显示的文字，两个 TTrackBar 控件分别用于设置阴影和字体大小，并将它们的 Max 属性分别设置为 20 和 100。

双击窗体 Form1，创建它的 OnCreate 事件处理函数，用于初始化两个 TTrackBar 控件。

```
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
 TrackBar1->Position=10;
 TrackBar2->Position=50;
}
//---------------------------------------------------------------------------
```

创建 Form1 的 OnPaint 事件处理函数，用于绘制窗体。

```
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::FormPaint(TObject *Sender)
{
 int h;
 h=TrackBar1->Position;
 Canvas->Brush->Style=bsClear;
 Canvas->Font->Name="隶书";
 Canvas->Font->Style<<fsBold<<fsItalic;
 Canvas->Font->Size=TrackBar2->Position;
 Canvas->Font->Color=clBlack;
 Canvas->TextOut(h,h,Edit1->Text);
 Canvas->Font->Color=clRed;
 Canvas->TextOut(0,0,Edit1->Text);
}
//---------------------------------------------------------------------------
```

双击 TrackBar1 控件，创建它的 OnChange 事件处理函数，并加入如下的一句代码，用于要求重画窗体。

```
//---------------------------------------------------------------------------
void __fastcall TForm1::TrackBar1Change(TObject *Sender)
{
 Invalidate();
}
//---------------------------------------------------------------------------
```

在对象监视器中，将 TrackBar2 的 OnChange 事件处理函数设置为 TrackBar1Change。

编译并运行程序，显示效果如图 1-3 所示。

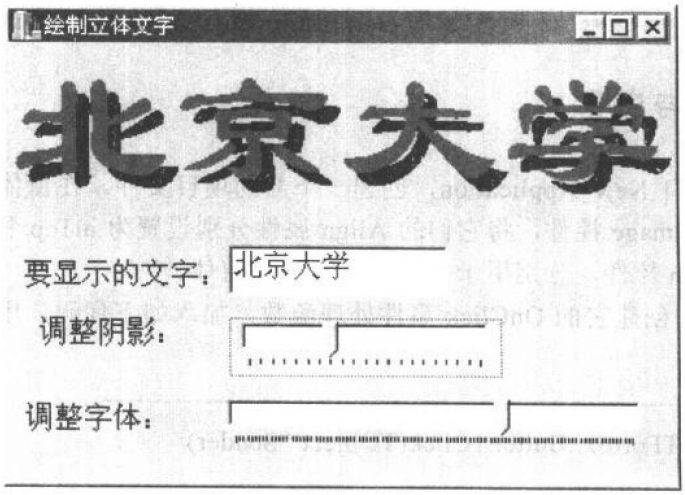

图 1-3　绘制立体文字

专家点评

在窗体的 OnPaint 事件处理函数中，用黑色显示在 Edit1 中输入的字符串，然后用红色在较上方显示同一字符串，就生成了立体字。注意不能调换两个 TextOut 方法调用的位置，应该先显示较灰色的字体，然后显示较亮色的字体。

当滑标移动时，触发 OnChange 事件，在 TrackBar1 和 TrackBar2 的 OnChange 事件句柄中调用了函数 Invalidate。此函数可以迅速擦除窗体，逼迫窗体重画（即调用窗体的 OnPaint 事件处理函数）。随着滑标的不断移动，字体不断发生变化，可以将字体调整到最佳状况。

Update、Repaint 和 Refresh 函数可以完成 Invalidate 函数同样的功能，但是它们之间的执行方法和步骤有些不同。Invalidate 函数不立即执行重绘操作，Windows 只是存储该请求，并在当前过程执行完成后没有其他事件有待系统解决时，再对它作出反映。利用 Update 函数要求 Windows 更新窗体的内容，立即重新绘制窗体，然而只有在 Invalidate 执行后该函数才执行。因此，常常在 Invalidate 调用后紧接着 Update 调用。这两步操作在 C++Builder 中完全可以用 Repaint 和 Refresh 来代替。Repaint 依次调用 Invalidate 和 Update，结果是立即激活 OnPaint 事件。该对象方法还有一个稍微不同的版本就是 Refresh。

1.5 拷贝屏幕和当前窗体

关键所在

利用 TCanvas 的图形拷贝方法 CopyRect 可以将一幅图像从其他画布拷贝到它的绘制表面。要拷贝整个屏幕，必须先获取桌面句柄，这可以使用 Win32 API 函数 GetDesktopWindow 函数来完成。而要拷贝当前窗体，方法比较多，这里介绍了两种完全不同的方法。一种是利用 CopyRect 方法，另一种是利用窗体的 GetFormImage 方法。

实现与应用

选择菜单 File | New Application，创建一个新的项目文件。在窗体 Form1 上放置一个 TPanel 控件和一 TImage 控件，将它们的 Align 属性分别设置为 alTop 和 alClient。在 Panel1 上放置三个 TButton 控件，分别用于拷贝屏幕和当前窗体。

双击 Button1，创建它的 OnClick 事件处理函数，加入如下代码，用于在 Image1 中拷贝整个屏幕。

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 TCanvas *DeskTop;
 DeskTop = new TCanvas;
```

```
 DeskTop->Handle= GetWindowDC(GetDesktopWindow()) ;
 Image1->Canvas->CopyMode=cmSrcCopy;
 Image1->Canvas->CopyRect(Rect(0, 0,ClientWidth,ClientHeight),
                          DeskTop,
                          Rect(0, 0,ClientWidth,ClientHeight));
 DeskTop->Free();
}
//---------------------------------------------------------------------------
```

分别双击 Button2 和 Button3，创建它们的 OnClick 事件处理函数，加入如下代码。通过两种不同的方法将窗体的图像拷贝到 Image1 中。

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
 Image1->Canvas->CopyMode=cmSrcCopy;
 Image1->Canvas->CopyRect(Rect(0, 0,ClientWidth,ClientHeight),
                          Canvas,
                          Rect(0, 0,ClientWidth,ClientHeight));
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
 Image1->Picture->Assign(GetFormImage());
}
//---------------------------------------------------------------------------
```

专家点评

CopyRect 方法与 GDI 函数中的 BitBlt 功能有些相似，但比 BitBlt 的使用要简单得多。该方法的具体形式如下：

CopyRect(const Windows::TRect &Dest, TCanvas* Canvas, const Windows::TRect &Source)

CopyRect 方法可以实现两个 TCanvas 之间任意两个矩形之间的拷贝。参数 Canvas 表示源画布，参数 Source 是源画布上要复制的图像区域，参数 Dest 表示目标画布上将接受复制图像的矩形区域。值得一提的是前面讲到的 TCanvas 的 CopyMode 属性，确定两个 TCanvas 之间如何进行拷贝。

程序在 Button1 的 OnClick 事件处理函数中，首先动态建立一个 TCanvas 对象（源画布），然后用 API 函数 GetDesktopWindow 获得桌面的句柄，利用 GetWindowDC 函数得到设备描述表的句柄，最后使用 CopyRect 方法将源画布拷贝到窗体（目标画布）上。

在 Button2 的 OnClick 事件处理函数中，将窗体 Form1 的 Canvas 属性作为参数调用

此为试读，需要完整PDF请访问：www.ertongbook.com