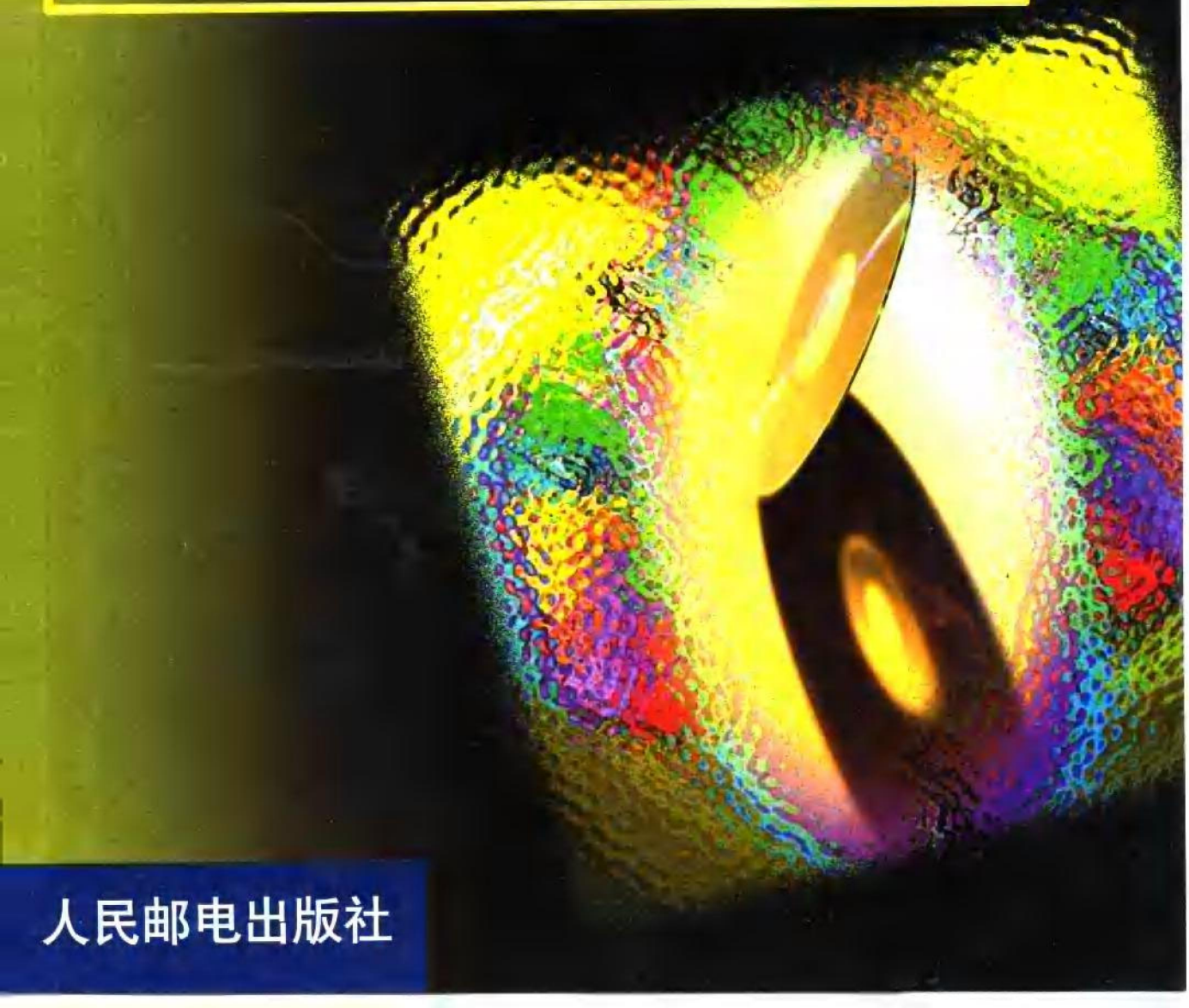
C++ Builder 4.0 多媒体开发技术

翟焱 冉欣 等编著



人民邮电出版社

C++ Builder 4.0 多媒体开发技术

翟 焱 冉 欣 等编著

人 民 邮 电 出 版 社

内 容 提 要

本书主要介绍应用 Borland C++ Builder 进行图像处理、动画制作以及视频处理的技术。全书共分为八章，内容包括 Borland C++ Builder 的图形图像处理技术、第三方类库 TeeChart 的应用方法、在 Borland C++ Builder 中使用 Windows 图像处理接口、多媒体应用程序特殊效果的实现、在 Borland C++ Builder 中使用 OpenGL 和 DirectDraw 接口进行多媒体编程，以及 Borland C++ Builder 的视频处理技术和基本游戏编程。

本书内容丰富，实用性强，适合具有一定的 Borland C++ Builder 编程基础的读者阅读，也可供 Borland C++ Builder 的初学者学习和参考。

C++ Builder 4.0 多媒体开发技术

◆ 编　著　翟　焱　冉　欣　等
责任编辑　马　嘉
◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销
◆ 开本：787 × 1092　1/16
印张：23.25
字数：578 千字　　2000 年 1 月第 1 版
印数：1 – 6 000 册　　2000 年 1 月北京第 1 次印刷
ISBN 7-115-08348-7/TP·1490

定价：35.00 元

前　言

自从 Borland 公司推出了新一代的集成化开发工具 Delphi 以来，这个快速高效的应用程序开发工具奠定了 Borland 公司在 RAD（Rapid Application Development）领域的领先地位，Delphi 也成为在 PC 机上唯一可以和工作站级的 Nextstep IB (Interface Builder) 相提并论的开发工具。几年前，Borland 公司开始研制一套代号为 Ebony 的应用程序开发工具，它被人称为 Delphi for C++，也就是现在的 Borland C++ Builder。

Borland 提出的程序设计思路已经被越来越多的程序编写人员所接受，其完全可视化的开发环境使多媒体应用程序的开发变得容易起来。Borland C++ Builder 使用了 Delphi 的程序开发风格，它使用 C++编程语言生成程序代码，因此能够进一步深入到操作系统核心，开发出更加优秀的应用程序。

Delphi 和 Borland C++ Builder 的出现使应用程序开发真正实现了工程化。以前，开发一个稍具规模的多媒体应用程序需要若干专业程序员花费几天的工作时间，而使用 Borland C++ Builder 开发，一个程序员仅仅需要几个小时就可以完成。

随着计算机技术以及信息产业的不断发展，特别是网络的日益普及，多媒体技术的应用和推广已成为计算机应用领域中一个非常重要的方面，在人们的日常生活与工作中占有重要的地位。

本书详细讲述了应用 Borland C++ Builder 进行多媒体程序开发的技术，其中包括开发图形图像处理应用程序的方法、动画编程以及在 Borland C++ Builder 中应用 DirectDraw、OpenGL 等应用程序接口。此外，本书也介绍了一些比较基本的图形处理技巧和应用程序特殊效果的实现。本书中重点讲述了应用 Borland C++ Builder 进行图像处理的技术，因为它的发展已经非常成熟，在实际中也被广泛地应用。

本书主要面向基本掌握 Borland C++ Builder 编程，有一定多媒体程序制作基础的读者。对于 Borland C++ Builder 的初学者，建议在使用本书时参考一些编程基础类书籍，以达到更好的学习效果。

本书第 1、2、3、5 章主要由翟焱编写，第 4、6、7、8 章主要由冉欣编写，全书内容的规划及审核由张宏林完成。参加本书编写工作的还有李家富、王辉、郑旭、王孟博、高畅、侯飞、马小力、刘森林、庞国华、武福军、马大永、刘明明、王建军、刘峰、赵昆、罗飞、唐虎、赵炎炎、张宇航、李长顺、刘光、李亚军、张惠、李欣、宋小宁、谷玉、张冉、郑则成、张国良，等等。

由于作者水平有限，加之时间仓促，书中的疏漏之处在所难免，希望广大读者批评指正。

作　者

目　录

第一章　VCL 对图像处理的支持 …… 1
1.1　图形对象概述 …… 1
1.2　功能强大的 TCanvas …… 2
1.2.1　TCanvas 介绍 …… 2
1.2.2　TCanvas 的属性 …… 2
1.2.3　TCanvas 的方法 …… 5
1.2.4　TCanvas 的事件 …… 9
1.3　Cananvas 对象的属性 …… 9
1.3.1　TPen 对象 …… 10
1.3.2　Tbrush 对象 …… 18
1.3.3　TFont 对象 …… 21
1.4　使用 Canvas 的方法绘图 …… 22
1.4.1　画点 …… 22
1.4.2　画线 …… 22
1.4.3　画椭圆 …… 23
1.4.4　绘制矩形 …… 24
1.5　运行期间图形的打印输出——TPrinter 对象 …… 24
1.6　TShape 组件 …… 27
1.7　Canvas 的综合应用 …… 31
1.7.1　响应鼠标事件 …… 31
1.7.2　建立快捷按钮 …… 33
1.7.3　绘图功能的实现 …… 35
第二章　图像处理 …… 45
2.1　图像对象概述 …… 45
2.2　TPicture 对象 …… 45
2.3　TGraphic 对象 …… 56
2.3.1　复制、装入和保存图像 …… 57
2.3.2　剪贴板和粘贴剪贴板图像 …… 59
2.4　TBitmap 对象 …… 62
2.4.1　建立 TBitmap 对象 …… 67
2.4.2　平移图像 …… 67
2.4.3　图像颠倒 …… 72
2.4.4　渐变效果 …… 74
2.4.5　柔化效果 …… 77

2.4.6 锐化效果 …… 79
2.4.7 浮雕效果 …… 80
2.5 TMetafile 对象 …… 81
2.5.1 建立 TMetafile 对象 …… 82
2.5.2 打开 Metafile …… 84
2.5.3 利用 TMetafile 绘图 …… 84
2.5.4 保存绘图结果 …… 85
2.6 TImage 对象 …… 92
2.6.1 装入图像 …… 93
2.6.2 在图像上绘图 …… 94
2.7 图形转换 …… 96
2.8 TPaintBox 对象 …… 101
第三章 TeeChart 的使用 …… 107
3.1 TeeChart 简介 …… 107
3.2 使用 TeeChart 向导 …… 107
3.2.1 编辑 Chart …… 110
3.2.2 编辑 Series …… 115
3.3 操纵 Series …… 119
3.3.1 Series 的类型 …… 119
3.3.2 处理 Series …… 136
3.3.3 动态控制 Series …… 147
3.4 处理 Chart …… 166
3.4.1 在 Chart 上绘图 …… 166
3.4.2 Chart 的缩放与滚动 …… 173
3.4.3 Chart 的打印 …… 179
第四章 Windows 图像高级编程 …… 182
4.1 Windows 图像设备接口 …… 182
4.1.1 设备上下文 …… 182
4.1.2 GDI 坐标系统和映像模式 …… 189
4.2 GDI 对象 …… 193
4.2.1 对象之间的关系 …… 193
4.2.2 位图对象 …… 194
4.2.3 区域对象 …… 195
4.3 处理颜色 …… 196
4.3.1 系统调色板 …… 196
4.3.2 逻辑调色板 …… 197
4.3.3 使用逻辑调色板 …… 197
4.4 使用字体 …… 201
4.4.1 字符集 …… 201

4.4.2 字体类型 …… 202
4.4.3 取得有关字体的信息 …… 203
4.4.4 创建字体 …… 203
4.5 位图资源 …… 205
4.5.1 建立位图资源 …… 205
4.5.2 从资源获得图像 …… 206
第五章 特殊效果的实现 …… 209
5.1 制作动画 …… 209
5.2 TAnimate 的使用 …… 212
5.3 Windows 消息处理与特殊效果 …… 213
5.3.1 拖动窗体 …… 215
5.3.2 窗体三维效果 …… 217
5.3.3 显示背景图像 …… 220
5.3.4 屏幕图像的捕获 …… 221
5.3.5 不规则形状窗体 …… 223
5.4 屏幕保护程序 …… 224
5.4.1 一个简单的屏幕保护程序 …… 224
5.4.2 OpenGL 屏幕保护程序 …… 237
第六章 用 OpenGL 作图 …… 245
6.1 OpenGL 概述 …… 245
6.2 OpenGL 的功能 …… 245
6.3 OpenGL 的作图流程 …… 246
6.3.1 OpenGL 的操作分类 …… 247
6.3.2 OpenGL 库函数的组成 …… 247
6.4 在 C++Builder 中利用 OpenGL 的一般过程 …… 248
6.5 一个简单的实例 …… 250
6.6 绘制几何体 …… 253
6.6.1 图形的颜色 …… 253
6.6.2 点、线、多边形的绘制 …… 254
6.6.3 二次曲面的绘制 …… 258
6.7 OpenGL 中图形的变换 …… 262
6.7.1 矩阵变换 …… 262
6.7.2 取景变换 …… 264
6.7.3 模型变换 …… 264
6.7.4 投影变换 …… 265
6.7.5 视见区变换 …… 267
6.8 光照和材料 …… 267
6.8.1 光照概念 …… 268
6.8.2 定义光源的特性 …… 268

6.8.3 定义材料属性 …… 270
6.8.4 纹理操作 …… 271
6.9 混合、反走样和雾化处理 …… 285
6.9.1 混合 …… 285
6.9.2 反走样 …… 285
6.9.3 雾化处理 …… 286
6.10 位图的处理和显示 …… 286
第七章 用 DirectDraw 作图 …… 290
7.1 DirectDraw 简介 …… 290
7.1.1 在用 DirectDraw 之前 …… 290
7.1.2 DirectDraw 是什么 …… 290
7.1.3 为什么要用 DirectDraw …… 291
7.2 基本概念和术语 …… 292
7.2.1 显示模式 …… 292
7.2.2 硬件加速 …… 292
7.2.3 调色板 …… 292
7.2.4 窗口应用程序和全屏应用程序 …… 293
7.3 DirectDraw 的对象 …… 293
7.3.1 DirectDraw 对象 …… 293
7.3.2 DirectDrawSurface 对象 …… 293
7.3.3 DirectDrawPalette 对象 …… 295
7.4 C++ Builder 中用 DirectDraw 作图的一般过程 …… 295
7.4.1 一个有关初始化的简单例子 …… 295
7.4.2 用 DirectDrawCreate 创建一个 DirectDraw 对象 …… 296
7.4.3 用 SetCooperativeLevel 转入排他模式 …… 297
7.4.4 用 SetDisplayMode 设置显示模式 …… 297
7.4.5 用 CreateSurface 创建一个主平面 …… 297
7.4.6 用 GetAttachSurface 获取屏后平面的指针 …… 298
7.5 位图的显示 …… 299
7.6 调色板的显示 …… 299
7.7 一个简单的动画例子 …… 300
7.8 调试 …… 307
第八章 多媒体和游戏编程 …… 308
8.1 多媒体的一些概念 …… 308
8.1.1 音频、MIDI 音乐和数字化音乐 …… 309
8.1.2 数字视频 …… 309
8.2 MCI 控制播放原理 …… 309
8.2.1 MCI 接口 …… 309
8.2.2 播放时间位置控制 …… 311

8.2.3 MCI 设备 …… 311
8.3 媒体播放机元件 TMediaPlayer 及其运用 …… 312
8.3.1 TMediaPlayer 元件的主要属性 …… 312
8.3.2 TMediaPlayer 元件的事件和方法 …… 317
8.3.3 C++ Builder 中用 MCI 控制播放及其时间格式 …… 320
8.3.4 MCI 控制播放原理 …… 320
8.4 程序背景音乐—— MIDI 的播放 …… 320
8.5 媒体播放器的制作 …… 324
8.6 游戏编程 …… 331
8.7 游戏引擎的演示 …… 338

第一章 VCL 对图像处理的支持

为了支持用户对图像处理技术的应用和开发，C++ Builder 在 VCL 中封装了基本的 Windows 图形设备接口，以便用户直接使用，利用这些封装完备的接口，用户可以方便快捷地进行图像处理。

1.1 图形对象概述

随着计算机技术的迅猛发展和应用范围日益广泛，高质量的计算机图形已经成为 Windows 应用程序的重要组成部分。为此，Microsoft Windows 为开发它们准备了很完备的 API 类库，但是使用过 Win32 API 的用户肯定能体会到使用 API 进行 Windows 图形编程的繁琐。而 C++ Builder 在 VCL 中为用户封装了基本 Windows 图形设备接口（GDI），开发者可以直接调用，这就使得 Windows 图形编程的工作得到极大的简化。本章将介绍 C++ Builder VCL 中基本图形组件的使用。本章内容可以应用于开发基本 Windows 程序，同时学习本章有助于掌握后面的 DirectDraw 和游戏编程等内容。

VCL 主要包括了表 1－1 中几种图像处理组件，详细信息可以查看 C++ Builder 的 Graphics.HPP 文件。

表 1－1　　VCL 封装的图像处理组件

组　件	说　明
TCanvas	基本图像处理组件，用于在窗体或者其它可作图容器组件中绘制各种几何图形形状
TBrush	用于控制封闭形状，例如矩形、椭圆的填充颜色或者贴图图案
TPen	用于画轮廓线
TPicture	图片组件，用于控制位图文件、图元文件、图标文件的显示
TMetaFileCanvas	用于绘制图元文件
TMetaFile	Windows 图元文件组件
TBitmap	Windows 位图文件组件
TIcon	Windows 图表文件组件
TGraphicsObject	TBrush, TFont 和 Tpen 的基类
TGraphic	TMetaFile, Tbitmap 和 Ticon 的基类

本章着重介绍 TCanvas 组件的使用和在屏幕上绘制各种几何形体，同时还将介绍 TBrush、TPen 等基本图形组件。

1.2 功能强大的 TCanvas

1.2.1 TCanvas 介绍

Canvas 即画布，TCanvas 对象即是用于在画布上输出图形，它是 VCL 封装的基本图像处理组件。TCanvas 对象封装了大部分 Windows 的图像处理功能，在 C++ Builder 里可以方便地利用 TCanvas 处理图像。

Canvas 已经嵌入到 Tform、TImage 等许多组件的核心，作为它们的属性。同时，Canvas 作为一个类也是许多对象和属性的聚集。这就给 Canvas 的使用带来了方便，这也说明 Canvas 在 Windows 应用程序中使用非常频繁。因此，在程序中可以以如下的形式调用 Canvas 方法和属性：

```
Form1->Canvas->TextOut(1, 1, "Hello from the canvas");
```

上面的代码可在窗体左上角输出“Hello from the canvas”，一般情况下可以直接写为：

```
Canvas->TextOut(1, 1, "Hello from the canvas");
```

1.2.2 TCanvas 的属性

TCanvas 强大的功能正是体现在其多种多样的属性和方法上。TCanvas 的属性包含了画笔、画刷、颜色、填充模式等绘图操作的各个必要组成部分，通过在程序中改变其属性值，用户可以方便地在需要的地方绘制出复杂的图案，下面就详细介绍 TCanvas 的属性。

1. Font 属性

声明为：

```
__property TFont* Font = {read=FFont, write=SetFont};
```

Font 属性可以指定画布上所有文字的字体，包括字体的类型、颜色、大小和风格等。

2. Brush 属性

声明为：

```
__property TBrush* Brush = {read=FBrush, write=SetBrush};
```

Brush 即是刷子，用来填充画布空间里图形的前景颜色或背景颜色。Brush 是 TCanvas 类的重要属性，将在后面详细讲解。

3. Pen 属性

声明为：

```
__property TPen* Pen = {read=FPen, write=SetPen};
```

TPen 对象用来设定画布上使用的画笔。包括画笔的颜色、粗细、风格和画笔的种类。

它同样也是 TCanvas 的一个重要属性，将在后面具体介绍。

4. PenPos 属性

声明为：

```
__property Windows::TPoint PenPos = {read=GetPenPos, write=SetPenPos};
```

PenPos 属性用来返回当前画笔在画布上所处的位置。一般不能用更改 PenPos 属性的方法来设定当前画笔的位置，如果要设定画笔的位置，需要用 MoveTo 方法

5. Pixels 属性

声明为：

```
property TColor Pixels[int X][int Y] = {read=GetPixel, write=SetPixel};
```

Pixels 属性用来设定或者返回当前画布上像素的颜色。X 和 Y 是像素的坐标，可以用这个属性进行点的绘制。

6. CanvasOrientation 属性

声明为：

```
__property TCanvasOrientation CanvasOrientation = {read=GetCanvasOrientation, nodefault};
```

这是一个只读属性，用于返回 Canvas 的方向，当(0,0)位于客户区左上角时为从左至右（left-to-right），其值是 coLeftToRight；当(0,0)位于客户区右上角时为从右至左（right-to-left），其值是 coRightToLeft。

7. ClipRect 属性

声明为：

```
__property Windows::TRect ClipRect = {read=GetClipRect};
```

ClipRect 属性直接返回画布的矩形区域，超过这个区域的图形将自动被剪裁。

8. CopyMode 属性

声明为：

```
__property int CopyMode = {read=FCopyMode, write=FCopyMode, default=13369376};
```

CopyMode 属性用来设置从其他画布上复制图像的方式，默认值是 cmSrcCopy，表示复制的画布将覆盖画布上原有的像素。CopyMode 可供选择的方式有多种，如表 1－2 所示。

表 1－2　　CopyMode 可供选择的方式

粘贴模式	模式功能
cmBlack	以黑色输出
cmDsInvert	把目标图像反转
cmMergeCopy	把目标文件与源图像文件进行布尔“与”操作
cmMergePaint	把源文件反转再与目标图像进行“或”操作
cmNotSrcCopy	把源文件反转复制

续表

粘贴模式	模式功能
cmNotSrcErase	先对源图像和目标图像进行“或”操作，再把结果反转
cmPatCopy	把源图像进行 XOR 操作后复制到目标设备
cmPatInvert	把目标图像与图案进行 XOR 操作
cmPatPaint	把源点位图反转再和图案进行“或”操作
cmSrcAnd	把目标像素与源点位图进行“与”操作
cmSrcCopy	复制并覆盖目标点位图
cmSrcErase	首先把目标点位图反转，然后再与源点位图进行“与”操作
cmSrcInvert	把源图像与目标图像的像素进行 XOR 操作
cmSrcPaint	把源图像与目标图像的像素进行“或”操作
cmWhiteness	使所有的输出以白色输出

9. Handle 属性

声明为：

```
__property HDC Handle = {read=GetHandle, write=SetHandle, nodefault};
```

Handle 属性返回画布的句柄，在用户调用 Windows API 时可能会用到这个句柄。只有在用户需要调用 Windows API 来实现 TCanvas 所不能实现或者不支持的功能时，才可直接调用 API。

10. LockCount 属性

声明为：

```
__property int LockCount = {read=FLockCount, nodefault};
```

LockCount 属性用来返回画布被锁定的次数，锁定画布以后，可以避免与其他的线程冲突，每调用一次 Lock，LockCount 属性就相应增加一次，对应地，如果调用一个 UnLock，则 LockCount 属性就相应减少一次，如果 LockCount 属性返回的值为 0，则表示锁定已经解除，其他线程可以访问该画布。

11. TextFlags 属性

声明为：

```
__property int TextFlags = {read=FTextFlags, write=FTextFlags, nodefault};
```

TextFlags 属性的设置表示了画布上文字的显示方式，该选择性属性有多种方式可供选择，具体如表 1－3 所示。

表 1－3　　TextFlags 可供选择的方式

TextFlags 设置	功　能
ETO_RTLREADING	文字或字符从左向右排列显示
ETO_IGNORELANGUAGE	有信息提供时，该标志的返回值用于检查微软标志的更新
ETO_CLIPPED	当文字在一个特定的矩形内显示时，调用 TextRect 方法，该标志会自动增加，如果用户在输出文字时调用 TextOut 方法，由于图像边界是由 TextExtend 属性决定的，因此对该标志没有影响

续表

TextFlags 设置	功　能
ETO_OPAQUE	文字输出在一个背景为不透明的区域内，但是背景不能透过文字所在的矩形显示出来
ETO_GLYPH_INDEX	文字字符是一组定位编码的排列组合，是通过计算机图形设备接口来解析的。该选项即字形索引只对 TrueType 的字体有效，如果需要对其他的字体有效则必须使计算机图形设备接口能对文字字符直接进行处理
ETO_NUMERICSLOCAL	该标志返回检查微软文件的更新，它是微软一种非正式的标志
ETO_NUMERICSLATIN	功用同上

1.2.3　TCanvas 的方法

如果说 TCanvas 的属性对绘图的各项属性进行了控制，那么绘图的实施则需要调用其方法来实现。TCanvas 提供了绘制矩形、圆形等基本几何图形的绘制方法，而且同时也可以通过调用 TCanvas 的方法实现对图形图像的控制，比如画布的锁定和刷新等等。下面就具体介绍 TCanvas 的各种方法的使用。

1. Arc 方法

声明为：

```
void __fastcall Arc(int X1, int Y1, int X2, int Y2, int X3, int Y3, int X4, int Y4)
```

此方法用于在画布上画弧形，弧被包含在由(X1、Y1)和(X2、Y2)组成的矩形区域内。弧的起点在(X3、Y3)，终点在(X4、Y4)。

2. BrushCopy 方法

声明为：

```
void __fastcall BrushCopy(const Windows::TRect &Dest, TBitmap* Bitmap, const Windows::TRect
    &Source, TColor Color);
```

此方法用于从一个位图复制部分图像到另外一个画布上，并且以目标画布上刷子的颜色替代位图上指定的颜色，其中 Dest 参数指定源图像的区域，Color 参数指定位图中要被替换的颜色。

3. Chord 方法

声明为：

```
void __fastcall Chord(int X1, int Y1, int X2, int Y2, int X3, int Y3, int X4, int Y4)
```

此方法用于在画布上画弦，即是由椭圆和直线相交所成的区域，椭圆的位置由矩形（X1、Y1）和（X2、Y2）决定，弦的起点在（X3、Y3），终点在（X4、Y4）。

4. CopyRect 方法

声明为：

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

调用该方法可以从一个画布上复制部分图像到另外一个画布上，Dest 参数指定目标画布的区域，Canvas 参数指定源画布，Source 参数指定源画布上要复制的矩形区域。

5. Draw 方法

声明为：

```
void __fastcall Draw(int X, int Y, TGraphic* Graphic)
```

调用 Draw 方法可用于在画布上指定位置输出由 Graphic 参数指定的图像，可以是位图，也可以是图标或者图元文件。

6. DrawFocusRect 方法

声明为：

```
void __fastcall DrawFocusRect(const Windows::TRect &Rect);
```

此方法可以画一个矩形边框，用于有输入焦点的矩形，第二次调用时该矩形框将消失。

7. Ellipse 方法

声明为：

```
void __fastcall Ellipse(int X1, int Y1, int X2, int Y2)
```

此方法用于画椭圆，椭圆的形状由（X1、Y1）和（X2、Y2）决定，内部区域由指定的刷子填充。

8. FillRect 方法

声明为：

```
void __fastcall FillRect(const Windows::TRect &Rect);
```

调用 FillRect 方法可以用当前设置的刷子填充 Rect 指定的矩形区域。

9. FloodFill 方法

声明为：

```
void __fastcall FloodFill(int X, int Y, TColor Color, TFillStyle FillStyle);
```

调用该方法可以用当前刷子填充屏幕上的区域，填充从（X、Y）开始向四周扩展，FillStyle 参数用于指定填充的风格，设为 fsBorder 表示填充一直继续除非遇到 Color 参数指定的颜色边界；设为 fsSurface 则表示填充 Color 参数指定的颜色边界。例如：

```
Canvas->FloodFill(ClientWidth/2, ClientHeight/2, clBlack, fsBorder);
```

上例可以从客户区中心点开始填充，直到遇到黑色为止。

10. FrameRect 方法

声明为：

```
void __fastcall FrameRect(const Windows::TRect &Rect);
```

调用该方法可以用当前刷子画一个矩形边框，不对其内部进行填充。

11. LineTo 方法

声明为：

```
void __fastcall LineTo(int X, int Y)
```

调用该方法可以用当前画笔画一条直线。

12. Lock 方法

声明为：

```
void __fastcall Lock(void);
```

调用该方法可对当前画布进行锁定，不允许其他线程访问该画布，一般不用

13. MoveTo 方法

声明为：

```
void __fastcall MoveTo(int X, int Y)
```

MoveTo 方法可直接移动画笔到指定位置。

14. Pie 方法

声明为：

```
void __fastcall Pie(int X1, int Y1, int X2, int Y2, int X3, int Y3, int X4, int Y4)
```

调用 Pie 方法用于画扇形（椭圆的一部分），椭圆由（X1，Y1）和（X2、Y2）决定，扇形由（X3、Y3）开始，终于（X4、Y4），用当前设定的刷子填充。

15. Polygon 方法

声明为：

```
void __fastcall Polygon(const Windows::TPoint * Points, const int Points_Size)
```

调用该方法可用于多个点的多边形绘画，将自动封闭并用当前设定的刷子填充此多边形区域。

16. Polyline 方法

声明为：

```
void __fastcall Polyline(const Windows::TPoint * Points, const int Points_Size);
```

调用该方法可连接多个点画折线，不封闭。

17. Rectangle 方法

声明为：

```
void __fastcall Rectangle(int X1, int Y1, int X2, int Y2);
```

该方法用于矩形的绘画，并用当前设置的刷子填充。

18. Refresh 方法

声明为：

```
void __fastcall Refresh(void);
```

调用该方法可以刷新 TPen、Tbrush 和 TFont 对象的变更。

19. RoundRect 方法

声明为：

```
void __fastcall RoundRect(int X1, int Y1, int X2, int Y2, int X3, int Y3)
```

调用该方法可以绘制带圆角的矩形，矩形由（X1、Y1）和（X2、Y2）定位，（X3、Y3）用来设定圆角椭圆的宽度和高度。

20. StretchDraw 方法

声明为：

```
void __fastcall StretchDraw(const Windows::TRect &Rect, TGraphic* Graphic)
```

该方法参数中 Graphic 参数将指定一个图形，调用该方法后所选定的图形将在 Rect 参数指定的区域内输出，图形将根据指定矩形大小进行自动放大或缩小。

21. TextExtent 方法

声明为：

```
TSize __fastcall TextExtent(const AnsiString Text);
```

调用该方法将返回所指定字符串的宽度和高度。

22. TextHeight 方法

声明为：

```
int __fastcall TextHeight(const AnsiString Text);
```

调用该方法将返回所指定字符串的高度。

23. TextOut 方法

声明为：

```
void __fastcall TextOut(int X, int Y, const AnsiString Text);
```

调用该方法可用指定的字体输出字符串。

24. TextRect 方法

声明为：