【转】VB6 GDI+ 入门教程（1）——GDI+介绍

引言：鉴于网上关于 GDI+的教程都是.Net 的，基本上没有 VB6.0 的，而这方面又很多人有需要，所以我就写一个 Visual Basic 6 GDI+ 入门教程。
目标人群：所有能够较熟练使用 VB 的，对 GDI+感兴趣或有 GDI+编程需要的人。

1.What's GDI+

官方解释：GDI+是 Windows XP 中的一个子系统，它主要负责在显示屏幕和打印设备输出有关信息，它是一组通过 C++类实现的应用程序编程接口。顾名思义，GDI+是以前版本 GDI 的 继承者，出于兼容性考虑，Windows XP 仍然支持以前版本的 GDI，但是在开发新应用程序的时候，开发人员为了满足图形输出需要应该使用 GDI+，因为 GDI+对以前的 Windows 版本中 GDI 进行了优化，并添加了许多新的功能。
作为图形设备接口的 GDI+使得应用程序开发人员在输出屏幕和打印机信息的时候无需考虑具体显示设备的细节，他们只需调用 GDI+库输出的类的一些方法即 可完成图形操作，真正的绘图工作由这些方法交给特定的设备驱动程序来完成，GDI+使得图形硬件和应用程序相互隔离，从而使开发人员编写设备无关的应用程 序变得非常容易。

我的解释：GDI+其实就是一个绘图模块，用于在屏幕上输出各种需要的内容。

2.GDI+ DLL

GDI+的 Dll 在 Windows XP+中默认存在，如果 Windows XP 以下系统需要使用 GDI+，那么需要从微软网站上下载安装包。

3.使用 GDI+

GDI+在.net Framework 中默认集成，只要添加它的命名空间（System.Drawing.Drawing2D）就能够使用了；而 GDI+在其它上面就没有那 么容易了，例如 VB6 就需要添加 GDI+的 API。对于初学者，写一堆 API 可能比学 GDI+用时还要长，不过我整理好了 API 到了一个模块，使用时候呢 只要在 VB 里面加载一下就可以啦！

下载地址：
Http://filer.blogbus.com/1506269/resource_1506269_1297068731q.bas

【转】VB6 GDI+ 入门教程（2）——GDI+初始化

2012-11-24 00:11 2535 人阅读 评论(0) 收藏 举报
分类：
Gdi+(gdiPlus)（5） 图形图像（26） VB（29） 入门介绍（17）

现在先让我们了解下 GDI+的绘图机制。

1.初始化、关闭 GDI+

我们需要对 GDI+进行初始化，才能使用它的各种功能。如果没有初始化，那么 VB6 就会莫名其妙的崩溃。呵呵。

当然程序结束了我们还要关闭 GDI+释放内存。

2.Graphics

Graphics 是 GDI+基础。首先我们需要一个图形对象 graphics（可以看 作是画板），我们所有的东西都要画在这个上面。那么如何显示呢？不要急，我们可以通过 GDI+内置函数从一个对象的 DC（设备描述表）上创建 graphics。这样我们操作 graphics 的时候就会显示在对象上。当然我们还可以从对象的 hwnd 中创建；在.net 中也可以从 gdi+的图像(image)中创建（直接操作在图像上）。

3.绘图工具

有了画板，我们还要画笔、画刷才能画画 - -。画笔画刷呢，在 gdi+中就叫做 pen、brush。画笔 pen 只能画一个轮廓（画线），而画刷可以对一个东西进行填充（刷子）。这个就是一个基础 呵呵，很简单吧。

4.创建第一个 VB6 的 GDI+ 程序

首先，我们添加下 GDI+模块；然后我们需要对窗体（以后可以是其它容器）属性进行设置：AutoRedraw=True，开启自动重绘；*再把 ScaleMode 设置成 3(Pixel 像素)*（注：这么做是为了 UI 时候的统一，这一步是否执行对于 GDI+直接绘制的结果不会有影响），因为 GDI+基础单位就是像素（当然可以用别的单位）

好 现在双击窗体，写入下面代码：

[vb] view plain copy
print?

```
1. Option  Explicit
2. Dim  graphics  As  Long
3.
4. Private  Sub  Form_Load()
5.
6.         InitGDIPlus
7.         GdipCreateFromHDC  Me.hDC,  graphics
8.
9. End  Sub
10.
11.Private  Sub  Form_Unload(Cancel  As  Integer)
12.
13.        GdipDeleteGraphics  graphics  '释放 graphics 占用的内
   存
14.        TerminateGDIPlus
15.
16.End  Sub
```

OK，F5 运行。如果没有问题的话我们第一个最基础的 GDI+程序已经完成了。这个基本的程序创建了一个 graphics 对象，当然什么还没有画呢。
通过这个程序，我们就大致了解 VB6 中 GDI+如何初始化、关闭了。首先呢要启动 GDI+，然后要创建一个 graphics；关闭的时候也要做好扫地工作。

5.画线

线嘛，又不是填充，根据前面说的，我们需要一个 pen。那么如何创建 pen 呢？呵呵，下面的代码就能创建一个 pen（追加在 Form_Load 过程中的末尾）：

[vb] view plain copy
print?

```
1. Dim  pen  As  Long
2. GdipCreatePen1  &HFFFF0000,  1,  UnitPixel,  pen
```

这里已经新建了一个 pen。为什么是 GdipCreatePen1 而不是 GdipCreatePen2 什么的呢？你可以在代码里面输入“mgdip.”这样就列出了所有的 GDI+函数。通过对象浏览器可以得知 pen2 是根据 brush 来创建 pen 的，现在不用。

&HFFFF0000：这里就是一个 16 进制的 ARGB (Alpha,Red,Green,Blue——透明，红色，绿色，蓝色程度，255(&HFF)是完全，0(&H0)是完全不) 的数据。当然你可以输入 10 进制，只是 16 进制很方便，2 个 16 进制位就是一个字节，如&HFFFF0000 就代表一个透明度是 255（不透 明），颜色是红色的一种颜色。如果你知道一些绘图技巧就很容易用这个去写呵呵~。同时我们还能看到 gdi+过程是**传址**的，把 pen 传进去。为什么不用函数返回值传

出来呢？因为函数返回值要传出一个状态，执行结果的标志。一般如果成功了那么就返回的是 0（Ok）。

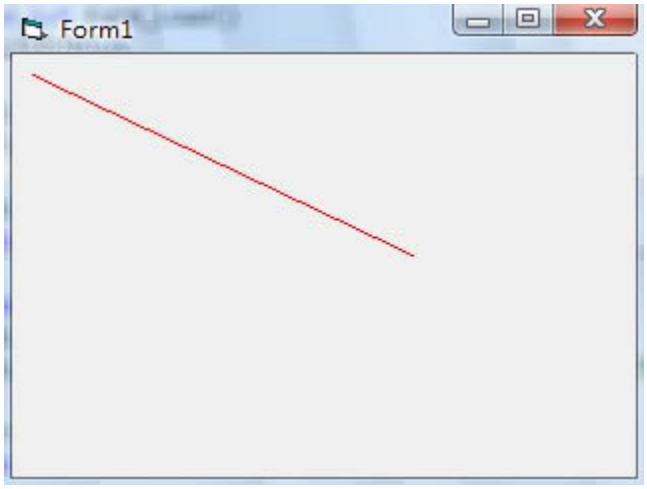
好，现在已经拿到笔了，接下来就是用这个笔去画线了。**通过查询 mGdip 的声明（请学会自主探索）**可 知有这么个 API：GdipDrawLine，它的 X1Y1,X2Y2 是 single 型，继续找又发现 GdipDrawLineI，它的坐标值都是 Long 型（我们一般用不到 single，因此我们一般用 GdipDrawLineI 就行了）。根据它的参数名字 乱猜都能猜出来哪个参数代表什么了，于是我随便写了一句：GdipDrawLineI graphics, pen, 10, 10, 200, 100。注意：你需要把 graphics 和 pen 传进去，否则怎么画呢？不告诉它画在哪里~~，后面 4 个参数分别对应：起始点 X、起始点 Y、终点 X、终点 Y 的坐标。当然扫地工作也要做好，删除 pen 的语句是 GdipDeletePen；参数很简单，传 pen 进去即可。

综合起来，于是我们有了第一段真正绘制的 GDI+ VB6 程序，虽然它只画了一条线：

[vb] view plain copy
print?

```
1. Option  Explicit
2.
3.
4. Dim  graphics  As  Long
5.
6. Dim  pen  As  Long
7.
8. Private  Sub  Form_Load()
9.
10.         InitGDIPlus
11.
12.         GdipCreateFromHDC  Me.hDC,  graphics
13.
14.         GdipCreatePen1  &HFFFF0000,  1,  UnitPixel,  pen

15.
16.         GdipDrawLineI  graphics,  pen,  10,  10,  200,  1
    00
17.
18. End  Sub
19.
20.
21.
22. Private  Sub  Form_Unload(Cancel  As  Integer)
23.
24.         GdipDeletePen  pen  '删除这个笔(pen)
25.
```

```
26.         GdipDeleteGraphics  graphics  '释放 graphics 占用
    的内存
27.
28.         TerminateGDIPlus
29.
30. End  Sub
31.
32. <span  style="font-size:14px;">  </span>
```



OK，F5 运行。红线没有出来？^_^……注意了 我们是在 Load 中绘制的。GDI+绘制与 VB 自己语句绘制一样。我们需要让他自动重绘（**窗体的 AutoRedraw=True**）或者放到 Paint 里面：）

注：如果你不是在 Load 事件 中绘制的东西，并且 Form 的 AutoRedraw 是 True，那么别忘记全部画完后 Refresh(例如 Me.Refresh)一下。因为 VB 的 AutoRedraw 的机制就是这样。但是开启 AutoRedraw 比较好，因为这样可以获得更快的绘制速度；全 部绘制完一次性刷新还能避免闪烁啥的问题。

【转】VB6 GDI+ 入门教程（3）——笔、刷子、矩形、椭圆绘制

好，我们已经学会如何画线了，那么后面的事情只要变通下都可以解决。不过变通前我还是得说几个基本的东西。

1.绘制，填充一个矩形

绘制一个整型长度的矩形，我们要用到 GdipDrawRectangleI 和 GdipFillRectangleI。前者用 pen 画一个轮廓边框，后者用 brush 刷出一个填充区域。当然接 下来就是如何创建刷子的问题了。GDI+中有多种刷子，有纯色刷子（创建：GdipCreateSolidFill），有渐变刷子（创 建：GdipCreateLineBrush），还有纹理刷子，贴图刷子，路径刷子等等……它们用于不同的方面。

(1)绘制一个矩形边框

首先，我们需要一个 pen。

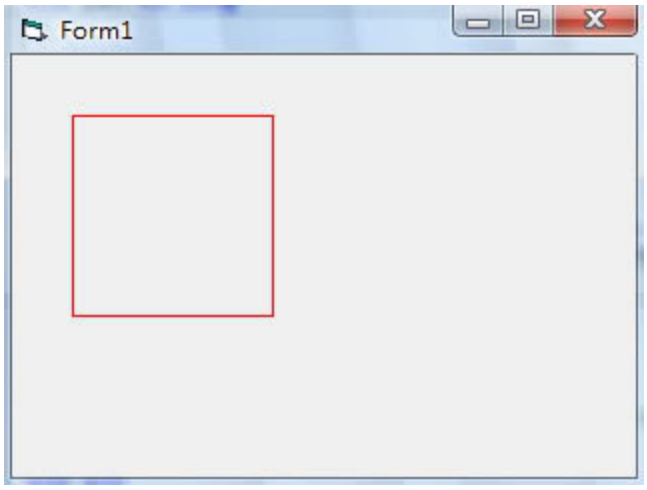
第一步，Dim！当然，我这样写了：Dim pen As Long；

第二步，创建一个红色的 pen（线的粗细是 1px）：GdipCreatePen1 &HFFFF0000, 1, UnitPixel, pen。

pen 创好了，接下来画矩形。这里我们用 GdipDrawRectangleI 来画矩形。画矩形跟画线可不一样，虽然指定坐标的都是 4 个参数，但是矩形里面四个参数分别是：X,Y,长,宽。OK，综合一下，代码如下：

[vb] view plain copy
print?

```
Option Explicit

Dim graphics As Long
Dim pen As Long

Private Sub Form_Load()
        InitGDIPlus
        GdipCreateFromHDC Me.hDC, graphics
        GdipCreatePen1 &HFFFF0000, 1, UnitPixel, pen
        GdipDrawRectangleI graphics, pen, 30, 30, 100, 100
End Sub

Private Sub Form_Unload(Cancel As Integer)
        GdipDeletePen pen
        GdipDeleteGraphics graphics '释放graphics占用的内存
          TerminateGDIPlus
End Sub

```



现在我们就绘制了一个100*100的红色矩形边框。很简单吧，变通就是这样的。

(2)创建纯色刷子

任何刷子包括其它的GDI+元素基本上都是一个思路：我们首先要Dim一个long型变量储存刷子/其它元素的地址，然后再调用GDI+相关函数去创建出指定刷子/其它元素。现在来创建一个蓝色，透明度为&HAA的刷子，那么代码就是这样：

[vb] view plain copy
print?

```
1. Dim  brush  As  Long
2. GdipCreateSolidFill  &HAA0000FF,  brush
```

刷子就这样拿到了。当然不要忘记扫地工作：GdipDeleteBrush brush。

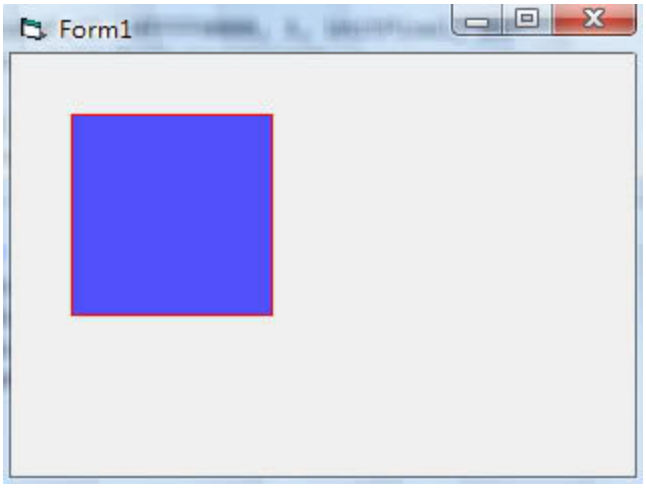
(3)用刷子填充一个矩形

很明显，函数里面找一下就会发现我们要的函 数：GdipFillRectangleI。它与DrawRectangleI 很类似，只不过把 pen 变成了 brush，因为现在要刷上去嘛- -。这一步也是很好变通的。把前面的一翻整理之后我们得到了一个绘制矩形并填充（不如说先填充再画边框，至于原因你可以自己颠倒一下顺序看下结果）的代码 （注意 填充色是有透明度的）：

[vb] view plain copy
print?

```
1. Option  Explicit
2.
3. Dim  graphics  As  Long
4. Dim  pen  As  Long,  brush  As  Long
5.
6. Private  Sub  Form_Load()
7.         InitGDIPlus
8.         GdipCreateFromHDC  Me.hDC,  graphics
9.         GdipCreatePen1  &HFFFF0000,  1,  UnitPixel,  pen
10.        GdipCreateSolidFill  &HAA0000FF,  brush

11.        GdipFillRectangleI  graphics,  brush,  30,  30,  100,
   100
12.        GdipDrawRectangleI  graphics,  pen,  30,  30,  100,  10
   0
13.End  Sub
14.
15.
16.
17.Private  Sub  Form_Unload(Cancel  As  Integer)
18.        GdipDeletePen  pen
19.        GdipDeleteBrush  brush
20.        GdipDeleteGraphics  graphics  '释放 graphics 占用的内
   存
21.        TerminateGDIPlus
```

22. End Sub



一样很简单吧！GDI+就是那么简单，只要懂了它的“工作机制”~！

(4)渐变刷子

渐变色很 Cool，纯 VB 代码却要很多，还好，GDI+有一个方便的渐变刷子函数——GdipCreateLineBrush。看参数，发现不一样：

[vb] view plain copy
print?

```
1. Function GdipCreateLineBrush(Point1 As POINTF, Point2 As POINTF, Color1 As Long, Color2 As Long, WrapMode As WrapMode, LineGradient As Long) As GpStatus
```

虽然复杂。。不过又很容易理解：Point1 是一个 PointF 结构，它储存了坐标的 X,Y，代表起始位置——这是渐变的起始、中止位置。一般情况下我们的起始中止位置是和绘制的图形一致的。Point2 一样，代表了终点。渐变就在这两个 点中“展开”。注意，现在是坐标点，而不是长宽值。Color1 自然就是起始颜色，Color2 则是第二颜色。WrapMode 就是填充方向，最后一个参 数自然就是传回建立好的 brush。

于是我们又开始写程序了，这次是创建一个蓝色->红色，纵向的刷子。绘制图形免去，**如果想看效果请自己添加 drawrectangle**（如果还要使用 point 画矩形请注意啦，rectangle 里面 后面参数是接受长宽，而刷子里面接受的是点……知道区别和解决方法了么？减呗！）。

[vb] view plain copy
print?

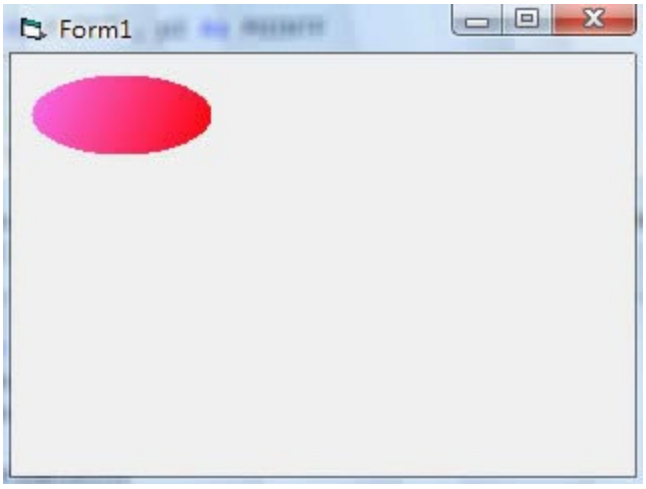
```
1. Dim  brush  As  Long
2. Dim  p1  As  POINTF,  p2  As  POINTF
3.
4. p1.X  =  0
5. p1.Y  =  0
6. p2.X  =  0
7. p2.Y  =  100
8. GdipCreateLineBrush  p1,  p2,  &HFF0000FF,  &HFFFF0000,  WrapMo
   deTileFlipy,  brush
```

2.绘制椭圆椭圆，想想也不会跟 rectangle 画法相差到哪里去。事实的确如此。下面就是一个绘制渐变椭圆的代码，解释免了吧，应该是很容易理解的。

[vb] view plain copy
print?

```
1. Option  Explicit
2. Dim  graphics  As  Long
3. Dim  brush  As  Long
4.
5. Private  Sub  Form_Load()
6.         InitGDIPlus
7.         GdipCreateFromHDC  Me.hDC,  graphics
8.
9.         Dim  p1  As  POINTF,  p2  As  POINTF
10.
11.        p1.X  =  10
12.        p1.Y  =  10
13.        p2.X  =  100
14.        p2.Y  =  50
15.
16.        GdipCreateLineBrush  p1,  p2,  &H8AFF00FF,  &HFFFF0000,
     WrapModeTileFlipXY,  brush
17.        GdipFillEllipseI  graphics,  brush,  p1.X,  p1.Y,  p2.X
     -  p1.X,  p2.Y  -  p1.X  '注意：类似的，绘制椭圆边框的语句是
   GdipDrawEllipseI
18.End  Sub
19.
20.Private  Sub  Form_Unload(Cancel  As  Integer)
```

```
21.         GdipDeleteBrush  brush
22.         GdipDeleteGraphics  graphics  '释放 graphics 占用的内
    存
23.         TerminateGDIPlus
24. End  Sub
```



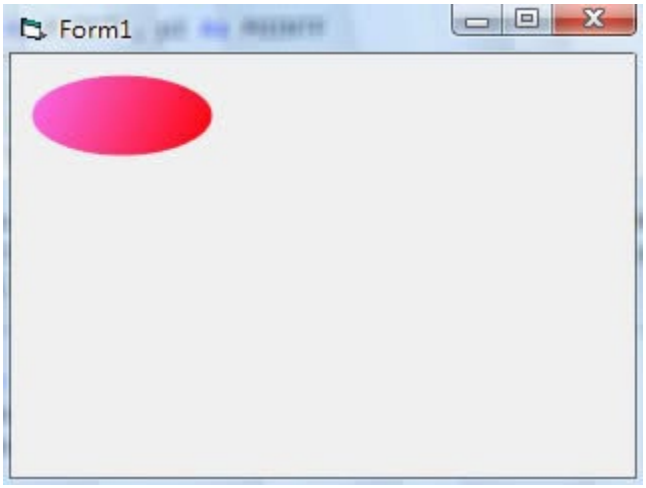
3.反锯齿功能

不知你有没有发现，画出来的椭圆是很不圆滑的（虽然用 VB 自己绘制也是如此）。如此强大的 GDI+怎么可能没有圆滑的功能呢？有！函数是 GdipSetSmoothingMode。我们需要把它加在绘制内容之前。一般我们把它加在创建好 graphics 之后（注意：它是作用于 graphics 的，因此请不要还没初始化 graphics 就设置 graphics 的光滑属性- -）。经过一翻调整，我们得出了一个设置圆滑的语句：

[vb] view plain copy
print?

```
1. GdipSetSmoothingMode  graphics,SmoothingModeAntiAlias
```

模式自然要设置成 AntiAlias，AntiAlias 顾名思义就是反锯齿(消除锯齿)。现在我们把这句话加到前面的椭圆程序中去，运行，如何，
椭圆很光滑吧！同样，前面任何一个程序都可以这样加使 gdi+画出来的东西看上去很平滑（为什么有人说是变模糊了呢= =、百思不得其解中...）。


Form1

【转】VB6 GDI+ 入门教程（4）——文字绘制

1.GDI+中文字的必须要素

首先，与其它软件一样，GDI+中的文字也有格式。画文字有多种画法，但是无论如何， 我们都需要创建一个 FontFamily，这其中包含了字体类型的信息，包括字体名称、字体对齐方式（需要设置）等等。一般的画法然后还要从这个 FontFamily 创建一个 Font，这个 Font 中包括字体样式（粗体、斜体）、字号等等，再后来我们调用一个函数把文字用这个 Font 显示出来~； 路径画法（可以显示边框画法）则不需要创建字体，直接调用函数，字体的样式包括在函数里面了。

可见，GDI+中文字是需要一个 FontFamily（一般是全局的），和一些 Font（各种不同样式）以及文字组成的。

2.GDI+绘制文字

GDI+绘制文字有几种，下面将分别示例。

(1)标准画法：GdipDrawString

这是一般的画文字的办法，这种画法支持 ClearTypeGridFit（还需要用语句再设置下），需要创建 Font。

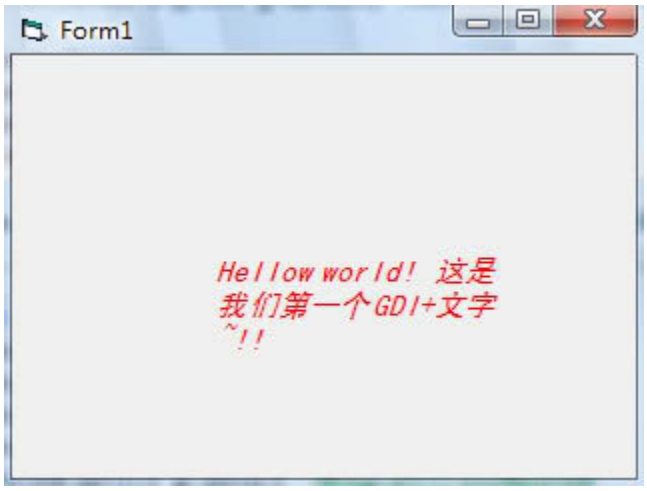
以下是主要绘图部分（窗体）：

[vb] view plain copy
print?

```
1. Option  Explicit
2.
3. Dim  graphics  As  Long,  Brush  As  Long
4. Dim  fontfam  As  Long,  strformat  As  Long,  curfont  As  Lon
   g,  rclayout  As  RECTF
```

```
5.
6. Private  Sub  Form_Load()
7.         InitGDIPlus
8.         GdipCreateFromHDC  Me.hDC,  graphics
9.
10.        GdipCreateFontFamilyFromName  StrPtr("黑体
   "),  0,  fontfam
11.        GdipCreateStringFormat  0,  0,  strformat
12.        GdipCreateSolidFill  &HFFFF0000,  Brush
13.        GdipSetStringFormatAlign  strformat,  StringAlignmentNe
   ar
14.        GdipCreateFont  fontfam,  15,  FontStyle.FontStyleItali
   c,  UnitPixel,  curfont
15.        GdipSetTextRenderingHint  graphics,  TextRenderingHintC
   learTypeGridFit
16.
17.        rclayout.Left  =  100
18.        rclayout.Top  =  100
19.        rclayout.Right  =  150
20.        rclayout.Bottom  =  150
21.
22.        GdipDrawString  graphics,  StrPtr("Hellow  world!  这是
   我们第一个 GDI+文字~!!"),  -
   1,  curfont,  rclayout,  strformat,  Brush
23.End  Sub
24.
25.
26.
27.Private  Sub  Form_Unload(Cancel  As  Integer)
28.
29.        GdipDeleteFontFamily  fontfam
30.
31.        GdipDeleteStringFormat  strformat
32.
33.        GdipDeleteFont  curfont
34.
35.        GdipDeleteBrush  Brush
36.
37.        GdipDeleteGraphics  graphics  '释放 graphics 占用的内
   存
38.
39.
40.
41.        TerminateGDIPlus
```

```
42.
43. End  Sub
```



可以看到这种画法思路是：

1. 创建 FontFamily（StrPtr:获取字符串指针，这样就能支持中文了！这就是不用 TLB 的原因……）

2. 创建 stringFormat（一般也可以不创），设置样式

3. 创建 Font。其中一定要注意单位问题。否则不要问我进去 14 输出的怎么不是 14px 大小文字……这里我们字体样式也巧妙了下，虽然声明中可以改写为 As FontStyle 但是不推荐。于是我们写就写 FontStyle.xxx 这样又可读性高，又不会出错。

4. 创建 Brush（显示文字咯）

5. 设置文字区域（RcLayout）

6. 绘制图形

7. 扫地工作

这样 完美地画出了字。

注意：rectf 中虽然是 right，bottom 但是实际上是 width height，不要被误导哟。！

(2)路径画法：GdipAddPathString

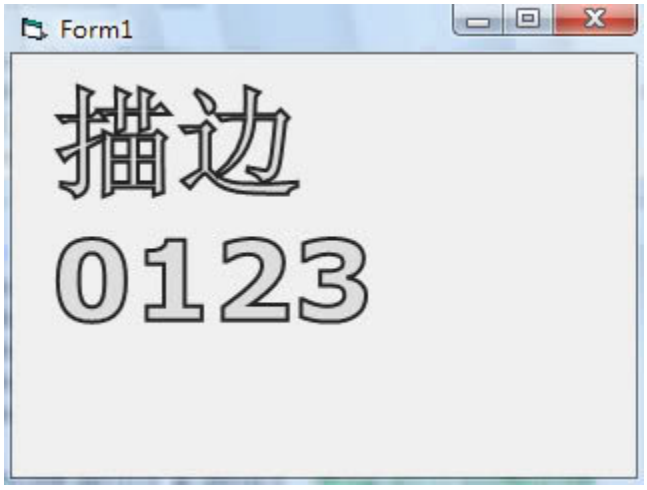
这种画法一般用于绘制旋转文字、描边的文字等等。虽然可以设置 graphics 的圆滑设置，但是它画出来的文字依然不怎么清晰（相对于第一种来说）

窗体中：

[vb] view plain copy
print?

```
Option Explicit

Dim graphics As Long, Brush As Long, Pen As Long
Dim fontFam As Long, strFormat As Long, strPath As Long, rclayout As RECTL

Private Sub Form_Load()

        InitGDIPlus
        GdipCreateFromHDC Me.hDC, graphics

        GdipSetSmoothingMode graphics, SmoothingModeAntiAlias

        GdipCreateFontFamilyFromName StrPtr("Verdana"), 0, fontFam
        GdipCreateStringFormat 0, 0, strFormat
        GdipSetStringFormatAlign strFormat, StringAlignmentNear
        GdipCreateSolidFill &HFFDEDEDE, Brush
        GdipCreatePen1 &HFF222222, 2, UnitPixel, Pen
        rclayout.Left = 10
        rclayout.Top = 10
        rclayout.Right = 200
        rclayout.Bottom = 150

        GdipCreatePath FillModeAlternate, strPath
        GdipAddPathStringI strPath, StrPtr("描边 0123"), -1, fontFam, FontStyle.FontStyleBold, 55, rclayout, strFormat
        GdipFillPath graphics, Brush, strPath
        GdipDrawPath graphics, Pen, strPath
End Sub
```

```
27.
28.
29.
30.Private  Sub  Form_Unload(Cancel  As  Integer)
31.         GdipDeleteFontFamily  fontFam
32.         GdipDeleteStringFormat  strFormat
33.         GdipDeletePath  strPath
34.         GdipDeleteBrush  Brush
35.         GdipDeletePen  Pen
36.         GdipDeleteGraphics  graphics  '释放 graphics 占用的内
    存
37.           TerminateGDIPlus
38.End  Sub
```



好 回来了 我们来比较一下这个画法有什么好处。

看出来了 它可以描边……恩 我不是在上面说了嘛 它还支持旋转、合并等等。

对了 我还说过“画出来不怎么清晰”，这里好像很好嘛！其实不然。如果你把描边去掉，单单 FillPath，并且把字号减小 比如 14，字体样式为普通，你就会发现不清晰了~!

它的过程是这样的：

1.首先前面部分和画普通文字一样 都需要创建 FontFamily 还有可选的创建字体对齐格式等等。

2.接下来路径画法不需要创建 Font，我们需要创建（初始化）一个路径，否则可是什么都没有哦~

3.然后我们需要把文字增加到 Path 中去。

4.我们要 FillPath 填充这个路径 或者是 DrawPath 描出这个路径。如果是实心文字自然就是 FillPath 咯

5.最后别忘了释放 Pen(如果有)和 Brush(如果有) 以及最后一个 Path。

(3)底层画法：GdipDrawDriverString

如名，底层画法。这种画法是最底层的绘制文字，底层到了……它不会自动转换字体（比如用 Verdana 绘制中文字体就不会显示出来） 由于不常使用，这里不贴画法了。

【转】VB6 GDI+ 入门教程（5）——基础绘图小结

2012-11-26 21:21 1660 人阅读 评论(0) 收藏 举报
分类:
Gdi+(gdiPlus) (5) VB (29) 图形图像 (26) 入门介绍 (17)

终于……我们的基础绘图部分可以先告一段落了。什么叫基础绘图？画线、画圈圈、画方块、画字……等等。我们来总结一下。

我们第一点就是总结 DrawXXXX 和 FillXXXX。

1.DrawXXXX：描边可以这么说 例如 DrawRectangle DrawPath。我们都需要一个 Pen（边框）来描绘它。

2.FillXXXX：填充。例如 FillRectangle 等等。我们需要的是 Brush。

我们第二点总结平滑（反锯齿）——什么时候用 GdipSetTextRenderingHint，什么时候用 GdipSetSmoothingMode。

这里我很简单的借用前面的结论告诉你：

1.凡是你要用 DrawXXXX 或者 FillXXXX 画出来的，你要让他平滑，你就要用 GdipSetSmoothingMode

2.其它的呢看它的类型，比如文字那么就是 GdipSetTextRenderingHint……（言下之意就是还有其它的东西哦）

我们第三点总结 Brush 和 Pen。

1.Pen 是一只笔（- -||）。用于 DrawXXXX 的。描边。你可以通过一个纯色创建 Pen（GdipCreatePen1），也可以通过一个 Brush 创建 Pen：GdipCreatePen2（比如说纹理 Pen，渐变 Pen 等等，不过貌似 GDI+有点 BUG）

2.Brush 呢是刷子。我们有贴图刷子，预置纹理刷子，纯色刷子，渐变刷子，路径刷子等等。

(1)贴图刷：我们会在下一章深入探讨

(2)纯色刷：我们已经用过了，很简单——给一个颜色，传回一个 Brush。

(3)渐变刷：我们也用过了，跟纯刷子差不多，给两个颜色就可以了，还有一个渐变方向~~，当然也是传回一个 Brush

(4)路径刷：这个刷子很高级 可以实现前面的(2)和(3)的刷子以及他们不能实现的内容——我们可以按照路径让他去渐变……还有很多其它功能。这个嘛 以后有空我也会说的 呵呵

我们第四点总结路径。

路径我们虽然只借用到了文字路径，但是如果你翻一下我提供的 API 大杂烩会发现 关于 Path 有很多有趣的东西。例如有添加直线路径，添加圆弧路径，添加曲线路径，路径合并，路径旋转等等……很强大吧。

路径，我们需要给他一个初始化好的 Path，然后按照各种需要给它参数；最后我们要把它画出来。

以后其它的路径东西我们有空会探讨。

最后再说下之前提过的一点：**如果你发现复制了我的代码 结果东西没出来，那么请确保你的窗体的 AutoRedraw=True。切记切记 不要忘记:)**

VB6 GDI+ 入门教程（6）——图片

VB 自己的绘图语句都需要用 LoadPicture 载入图片，同样，GDI+中也需要。

1.载入（初始化）图片资源

(1)来自文件：GdipLoadImageFromFile

我们先来看看这个最简单基本的载入图片来自文件：

[vb] view plain copy
print?

```
Dim img As Long, img_W As Long, img_H As Long

GdipLoadImageFromFile StrPtr("C:\TestImage.png"), img

'如果你希望得到长宽信息，可以使用下面的语句:

GdipGetImageWidth img, img_W

GdipGetImageHeight img, img_H

Msgbox "长为:" & CStr(img_W) & "px, 宽为:" & CStr(img_H) & "px."

'GdipDisposeImage img
```

载入图像之后别忘记释放 Image，否则会造成 MemoryLeak 内存泄漏(另外如果没有 Dispose 掉的话这个文件是被占用的)。

(2)来自资源文件：GdipLoadImageFromStream *(2010/2/9 修改)*

这个函数主要是用来从资源文件(RES)载入图像的，怎么载入呢？我们来看函数，函数 是从 Stream 载入，但是我们 VB6 没有集成 Stream 对象，从 RES 读取出来（LoadResData）也只是返回 Byte()。不过很好，OLE 提供了一个函数能够将 Byte()变为一个 IStream 对象——我们需要这个 API

[vb] view plain copy
print?

```
1. Declare Sub CreateStreamOnHGlobal Lib "ole32.dll" (ByVal
   hGlobal As Long, ByVal fDeleteOnRelease As Long, ByRef
   ppstm As Any)
```

但是，函数第一个参数需要的是一个**内存句柄**而不是**内存地址**， 这两个值有时一样有时不一样。当调用申请内存 GlobalAlloc 函数使用 GMEM_FIXED 参数时候它们相同，其它时候它们不同，一个数组的内存空 间是否是 GMEM_FIXED 申请的取决于数组的声明位置等各种因素。所以我们可不能这么冒险假定 hMem=pMem。那我们如何得到数据的内存句柄呢？ 新申请一块内存就得到句柄了，然后我们只需要复制数据即可。

[vb] view plain copy
print?

```
1. '声明部分如下
2.
3. Private Declare Function GlobalAlloc Lib "kernel32" (ByVa
   l wFlags As Long, ByVal dwBytes As Long) As Long
4.
5. Private Declare Function GlobalUnlock Lib "kernel32" (ByV
   al hMem As Long) As Long
6.
7. Private Declare Function GlobalLock Lib "kernel32" (ByVal
     hMem As Long) As Long
8.
9. Private Declare Function GlobalFree Lib "kernel32" (ByVal
     hMem As Long) As Long
10.
11. Private Declare Sub RtlMoveMemory Lib "kernel32" (ByRef
    Destination As Any, ByRef Source As Any, ByVal Length
    As Long)
12.
13. Const GMEM_MOVEABLE As Long = &H2
```

```
14.
15.
16.
17.'实现部分如下
18.
19.Dim  img  As  Long
20.
21.Dim  ResData()  As  Byte,  IStream  As  Object
22.
23.ResData  =  LoadResData(101,  "CUSTOM")    '获取数据
24.
25.
26.
27.'接下来需要一个内存句柄而不是内存地址
28.
29.Dim  hMemObj  As  Long,  pMem  As  Long
30.
31.hMemObj  =  GlobalAlloc(GMEM_MOVEABLE,  UBound(ResData)  +  1)
    '申请新内存获得句柄
32.
33.pMem  =  GlobalLock(hMemObj)'锁定内存块，返回内存块的指针
34.
35.If  pMem  =  0  Then    '分配内存失败
36.
37.        Msgbox  "Global  alloc  failed!"
38.
39.        Exit  Sub
40.
41.End  If
42.
43.RtlMoveMemory  ByVal  pMem,  ResData(0),  UBound(ResData)+1
   '复制源数据到新内存。
44.'注：最后一个参数，传递的是数据的字节数，由于数组是字节类型，所
   以数组有多少个元素，就有多少个字节
45.
46.GlobalUnlock  hMemObj'解锁
47.
48.CreateStreamOnHGlobal  ByVal  hMemObj,  False,  IStream    '根
   据新内存句柄创建 IStream
49.
50.GdipLoadImageFromStream  IStream,  img    '建立 Image
51.
52.Set  IStream  =  Nothing
53.
```

```
54.GlobalFree  hMemObj    '释放新内存
55.
56.'GdipDisposeImage  img      '图片不要忘记释放了
```

2.绘制图片

(1)GdipDrawImage(I)

这是 Gdi+绘图的一种基础画法，不需要进行长宽设置，不过我们平时不怎么用它。它按照图片的物理大小绘制，完全无视所有 Graphics 的 Scale 等缩放参数。这个函数支持 32 位透明通道绘制。*（技巧：有 I 的一般 坐标、长宽都是 Long 型 没有 I 的一般都是 Single 型）*

什么是物理大小？这个就要跟图像的分辨率(dpi)有关了。打开你的 Photoshop 或者是 Fireworks 或者是 AI 或者其他专业绘图软件，新建一个文档，你就会发现有分辨率选项，一般你看到的是 72 像素/英寸。但是，请注意，一般屏幕的分辨率是 96 像素/英寸。

96 这个值可以在系统的显示设置中看到。***在 Windows7 中的查看步骤是：****桌面右键->屏幕分辨率->放大或缩小文本和其他项目->(左侧)设置自定义文本大小(DPI)，在弹出对话框中有显示“每英寸 X 像素”。*

图片一般的分辨率与屏幕的分辨率不一致，这会有什么结果？一般不会有问题，**因为我们一般图像的绘制以 px 为单位，无论分辨率多高(结果是物理尺寸变小)，图像都是包含了同样数量的像素点**。可是现在这个函数是按照物理大小绘制的，这样 Dpi 的不同势必就会造成绘制出来的图像有“缩放”，一般呈现为比正常大小大。

分辨率如何调整？以后再说。

[vb] view plain copy
print?

```
1. Dim  img  As  Long
2. GdipLoadImageFromFile  StrPtr("C:\TestImage.png"),  img
3.
4. '此处请初始化 GDI+以及 graphics
5.
6. GdipDrawImage  graphics,  img,  0,  0
```

拓展阅读：

1. 72dpi 或 96dpi 的争论(cnBlogs)

2. Discussion of 72dpi & 96 dpi (English)

(2)GdipDrawImageRect(I) **推荐**

这是我们常用的画法，一般 Gdi+画图就用这个函数。我们可以对图片的大小进行平滑的拉伸缩放。

[vb] view plain copy
print?

```
1. Dim  img  As  Long
2.
3. GdipLoadImageFromFile  StrPtr("C:\TestImage.png"),  img
4. '此处请初始化 GDI+以及 graphics
5. GdipDrawImageRect  graphics,  img,  0,  0,  100,  200    '拉伸
   到 100*200
```

(3)GdipDrawImageRectRect(I)

还有个有点常用的函数就是这个了。通过它我们可以画一个图的一个部分，并且同样可以改变大小（好处：我们可以把所有的图片资源综合到一个图片中），另外它支持一个叫做 *ImageAttribs* 的东西，这是图片的滤镜，我们可以改变图片透明度和各种颜色参数(如二值化，灰度化等)。ImageAttrib(utes)会在之后的教程中有所涉及。

代码如下：

[vb] view plain copy
print?

```
1. Dim  img  As  Long
2.
3. GdipLoadImageFromFile  StrPtr("C:\TestImage.png"),  img
4. '此处请初始化 GDI+以及 graphics
5. GdipDrawImageRectRectI  graphics,  img,  20,  20,  10,  10,  0,
     0,  100,  200,  UnitPixel
```

注意咯：

第三~第六个参数是原来图片中要截取的部分；第七~第十呢则是画到哪里以及画出来多大的设置

第三~第六个参数是为绘制位置和绘制尺寸；第七~第十则是截取位置和截取尺寸。

(4)贴图刷

贴图刷子主要用来绘制平铺的内容。贴图刷子跟其它刷子一样，我们需要创建刷子，另外对于这个刷子我们需要先初始化图片:)

[vb] view plain copy
print?

```
1. Dim  img  As  Long,  textureBrush  As  Long
2.
3. GdipLoadImageFromFile  StrPtr("C:\TestImage.png"),  img
4. '此处请初始化 GDI+以及 graphics
5. GdipCreateTexture  img,  WrapModeTileFlipX,  textureBrush
6. GdipFillRectangle  graphics,  textureBrush,  0,  0,  100,  100
```

贴图刷子跟其它刷子有什么区别呢？普通的图片绘制（如 DrawImageRectI）支持的是拉伸，贴图刷则是平铺。另外贴图刷还要注意定位问题。因为**贴图刷纹理起始点是 Graphics 的 0,0**，而不是绘制内容的左上角坐标。

贴个图，直观明了。

那么贴图刷子中如何调整图片起始位置呢？我们可以平移图片——GdipTranslateTextureTransform。参数很简单 是平移量。（注意：这是个相对平移，也

就是这个平移是**参照之前量**的，而不是原图片；因此建议更改平移量要先 Reset 下：GdipResetTextureTransform）

【转】在 GDI+使用区域

2012-11-26 21:40 1344 人阅读 评论(0) 收藏 举报
分类：
VB（29） 图形图像（26） Gdi+(gdiPlus)（5） 入门介绍（17）

原文章地址：
Http://hi.baidu.com/belovedjuan/item/f5611eba57fafbf463388e7c

抠图原理：需要处理的图片都是具有单色背景的图片（否则，这种情况下就没有任何的意义）。假如图片的背景色是黑色，编程的主要工作就是把图中的非黑色部分抠出，形成一个区域。“抠”的原理就是：从图片的左上角（0，0）开始，按照从上到 下，从左到右的顺序形成长宽都为 1 的 N（N=图片长 X 图片宽）个矩形（实际上是一个像素），如果矩形中含有背景色黑色，将之除去，对剩下的矩形全都使用 CreateRectRgnIndirect 建立器若干 个区域，由于这些区域都不包含黑色（实际上代表着图片主要内容），所以，对这些区域进行合成就形成了主画面的区域，这就是“抠”出主画面区域的关键思路所 在。

[cpp] view plain copy
print?

```
1. void  CGDIPlusdemoView::OnRegionFromBitmap()
2. {
3. Graphics  graphsics(this->m_hWnd);
4. //绘制样本图片
5. Bitmap  bmp(AfxGetInstanceHandle(),(LPWSTR)MAKEINTRESOURCE(IDB_
   SCORE));
6. graphsics.DrawImage(&bmp,0,0);
7. //绘图平面右移，示意使用 GDI 生成区域
8. graphsics.TranslateTransform(bmp.GetWidth()+10,0);
9.
10.CDC*pDC=GetDC();
11.CDC  memDC;
12.CBitmap  bmpScore;
13.CBitmap*pOldMemBmp=NULL;
14.COLORREF  PixlColor,BackGroudColor;
15.CRect  cRect;
16.int  x,y;
17.GetClientRect(&cRect);
18.CPoint  TopLeft=cRect.TopLeft();
19.
20.CRgn  ScoreRGN,rgnTemp;
21.//载入图片
22.bmpScore.LoadBitmap(IDB_SCORE);
23.BITMAP  bmInfo;
24.bmpScore.GetObject(sizeof(bmInfo),&bmInfo);
25.CRect  picWnd=CRect(TopLeft,CSize(bmInfo.bmWidth,bmInfo.bmHeigh
   t));
26.memDC.CreateCompatibleDC(pDC);
27.pOldMemBmp=memDC.SelectObject(&bmpScore);
28.//指明背景色，图片的第一个像素值
29.BackGroudColor=memDC.GetPixel(0,0);
30.//创建一个与位图大小相等的区域
31.ScoreRGN.CreateRectRgn(0,0,picWnd.Width(),picWnd.Height());
32.//获取图片的每一个像素值
33.for  (x=0;x<=picWnd.Width();x++)
34.{
35.      for  (y=0;y<=picWnd.Height();y++)
36.      {
37.        PixlColor=memDC.GetPixel(x,y);
38.        if  (PixlColor==BackGroudColor)
39.        {
40.          //如果像素不包含蓝色，创建一个大小为 1 的区域
41.          rgnTemp.CreateRectRgn(x,y,x+1,y+1);
42.          //将这些区域相加，形成主画面的区域
```

```
43.             ScoreRGN.CombineRgn(&ScoreRGN,&rgnTemp,RGN_XOR);
44.             rgnTemp.DeleteObject();
45.           }
46.       }
47.}
48.//恢复设备环境
49.memDC.SelectObject(pOldMemBmp);
50.//从 CRGN 中创建区域
51.Region  region(ScoreRGN);
52.//填充区域
53.graphsics.FillRegion(&SolidBrush(Color::Blue),®ion);
54.}
```