PbIdea 功能系列之 PowerBuilder 高级图像处理功能

我们知道，PowerBuilder 是 CS 桌面应用的开发利器，具方便、快捷、高效的特点。然而，PowerBuilder 的特长在数据库方面，对于图像处理这块，功能几乎是一片空白，只有几个简单的图形形状，更谈不上图片处理与加工，甚至有些格式都不支持。

本文着重于介绍利用 PbIdea 扩展库，增强 PB 图片处理功能。

一、uo_image 对象

uo_image 对象着重于图片显示、转换、缩放等处理的一些基础功能，着重实现图片的工具性功能。

1、显示图片。uo_image 本身是一个可视控件，直接拖放到窗口上，再使用打开图片功能即可以显示图片。支持的图片格式有：bmp,jpg,png,gif。例如：

uo_1.open(“test.bmp”)

参数可以是文件名，也可以是 blob 数据，数据可以是从数据库中取出，也可以是从网上下载等等:

Selectblob into :pic from … 然后 uo_1.open(pic)

httpclient.request(……) 然后 uo_1.open(httpclient.response.data)

均可以直接显示图片，不需要存为图片文件再显示。

对于一些下载通过 base64 编码的图片，可以直接打开显示,例如：

Ls_pic = “……”

uo_1.open(ls_pic)

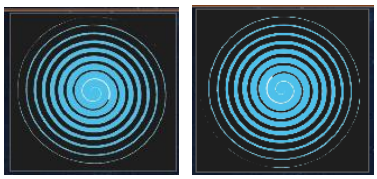
是不是感觉比 PB 自带的 Picture 控件强大了不少？

2、显示功能增强。我们显示图片文件到窗口的功能会了，支持的图片格式多，内容多。同时也支持一些增强功能。

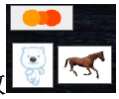
增强一：uo_1. SetAlpha(180) 以 180/255 的透明度显示。0 为全透明，255 为不透明。

增强二：使指定颜色变成透明。uo_3.SetMaskColor(RGB(255,255,255))，即让图片的白色背景变成透明颜色。

增强三：旋转图片角度。Uo_1. Rotate(15) ,即旋转 15 度。



增强四：基于上列一和三，实现渐变或者旋转动画显示效果。

增强五：支持动画 gif 播放 

增强六：uo_1. Flip()，图片上下翻转

增强七：uo_1.Paint(…) ，将本控件的图片内容绘制到其他控件上去，本控件可以隐藏。可以由多个 uo_image 把图片绘制到同一个控件上，形成图片叠加效果，在使用带有透明效果 png 图片时，这个功能比较有用。比如下列这张图就是一个透明的蝴蝶被叠加到窗口背景上无缝对接。

3、图片处理功能。除了作为显示控件，uo_image 也可以直接做一些图片处理。在这些处理功能中，上面列出的大部分增强功能也都可以使用。同时还有额外功能。

uo_image img

img = create uo_image

img.open(…) //打开图片

img. GetSize(w,h) //获取图片宽度和高度

img. ReplaceColor(…)//实现颜色替换，有域值设置，比如张纸的背景有点脏点模糊，可以替换为纯白背景，或者其他颜色背景。如果存为 png 图片，也可以把背景替换为透明色，类似于 PS 的扣图。

Img.zoom(…) //图片缩放，可以指定大小尺寸进行缩放，也可以指定缩放比例进行缩放。

Img. Save(…) //另存为图片，格式由给定的图片名称的文件扩展名决定。

Open，然后再指定文件名 Save，通常可以作为图片格式转换的通用方式。

4、二维码。二维码是现在比较常见的功能。Uo_image 可以生成并显示二维码，也可以将生成的二维码存为图片文件。

uo_1.OpenQrCode(" 大 自 在 781770213@qq.com", 1, 3,RGB(0,0,0), RGB(255,255,255), "..\demores\images\logo.png")

uo_1.save(…) 可另存为图片文件。

具体函数定义是： function boolean OpenBarCode(readonly string codeText,string barType, int nHeight,double scale,readonly string options) system library "PbIdea.dll" alias for "imageOpenBarCode"

5、条码。支持多种条码。其中也包括二维码。（文档 https://www.zint.org.uk/manual）

//条码类型定义

CONSTANT int BARCODE_CODE11 = 1

CONSTANT int BARCODE_C25MATRIX = 2

CONSTANT int BARCODE_C25INTER = 3

CONSTANT int BARCODE_C25IATA = 4

CONSTANT int BARCODE_C25LOGIC = 6

CONSTANT int BARCODE_C25IND = 7

CONSTANT int BARCODE_CODE39 = 8

CONSTANT int BARCODE_EXCODE39 = 9

CONSTANT int BARCODE_EANX = 13

CONSTANT int BARCODE_EAN128 = 16

CONSTANT int BARCODE_CODABAR = 18

CONSTANT int BARCODE_CODE128 = 20

CONSTANT int BARCODE_DPLEIT = 21

CONSTANT int BARCODE_DPIDENT = 22

CONSTANT int BARCODE_CODE16K = 23

CONSTANT int BARCODE_CODE49 = 24

CONSTANT int BARCODE_CODE93 = 25

CONSTANT int BARCODE_FLAT = 28

CONSTANT int BARCODE_RSS14 = 29

CONSTANT int BARCODE_RSS_LTD = 30

CONSTANT int BARCODE_RSS_EXP = 31

CONSTANT int BARCODE_TELEPEN = 32

CONSTANT int BARCODE_UPCA = 34

CONSTANT int BARCODE_UPCE = 37

CONSTANT int BARCODE_POSTNET = 40

CONSTANT int BARCODE_MSI_PLESSEY = 47

CONSTANT int BARCODE_FIM = 49

CONSTANT int BARCODE_LOGMARS = 50

CONSTANT int BARCODE_PHARMA = 51

CONSTANT int BARCODE_PZN = 52

CONSTANT int BARCODE_PHARMA_TWO = 53

CONSTANT int BARCODE_PDF417 = 55

CONSTANT int BARCODE_PDF417TRUNC = 56

CONSTANT int BARCODE_MAXICODE = 57

CONSTANT int BARCODE_QRCODE = 58

CONSTANT int BARCODE_CODE128B = 60

CONSTANT int BARCODE_AUSPOST = 63

CONSTANT int BARCODE_AUSREPLY = 66

CONSTANT int BARCODE_AUSROUTE = 67

CONSTANT int BARCODE_AUSREDIRECT = 68

CONSTANT int BARCODE_ISBNX = 69

CONSTANT int BARCODE_RM4SCC = 70

CONSTANT int BARCODE_DATAMATRIX = 71

CONSTANT int BARCODE_EAN14 = 72

CONSTANT int BARCODE_CODABLOCKF = 74

CONSTANT int BARCODE_NVE18 = 75

CONSTANT int BARCODE_JAPANPOST = 76

CONSTANT int BARCODE_KOREAPOST = 77

CONSTANT int BARCODE_RSS14STACK = 79

CONSTANT int BARCODE_RSS14STACK_OMNI = 80

CONSTANT int BARCODE_RSS_EXPSTACK = 81

CONSTANT int BARCODE_PLANET = 82

CONSTANT int BARCODE_MICROPDF417 = 84

CONSTANT int BARCODE_ONECODE = 85

CONSTANT int BARCODE_PLESSEY = 86

CONSTANT int BARCODE_TELEPEN_NUM = 87

CONSTANT int BARCODE_ITF14 = 89

CONSTANT int BARCODE_KIX = 90

CONSTANT int BARCODE_AZTEC = 92

CONSTANT int BARCODE_DAFT = 93

CONSTANT int BARCODE_MICROQR = 97

CONSTANT int BARCODE_HIBC_128 = 98

CONSTANT int BARCODE_HIBC_39 = 99

CONSTANT int BARCODE_HIBC_DM = 102

CONSTANT int BARCODE_HIBC_QR = 104

CONSTANT int BARCODE_HIBC_PDF = 106

CONSTANT int BARCODE_HIBC_MICPDF = 108

CONSTANT int BARCODE_HIBC_BLOCKF = 110

CONSTANT int BARCODE_HIBC_AZTEC = 112

CONSTANT int BARCODE_AZRUNE = 128

CONSTANT int BARCODE_CODE32 = 129

CONSTANT int BARCODE_EANX_CC = 130

CONSTANT int BARCODE_EAN128_CC = 131

CONSTANT int BARCODE_RSS14_CC = 132

CONSTANT int BARCODE_RSS_LTD_CC = 133

CONSTANT int BARCODE_RSS_EXP_CC = 134

CONSTANT int BARCODE_UPCA_CC = 135

CONSTANT int BARCODE_UPCE_CC = 136

CONSTANT int BARCODE_RSS14STACK_CC = 137

CONSTANT int BARCODE_RSS14_OMNI_CC = 138

CONSTANT int BARCODE_RSS_EXPSTACK_CC = 139

CONSTANT int BARCODE_CHANNEL = 140

CONSTANT int BARCODE_CODEONE = 141

CONSTANT int BARCODE_GRIDMATRIX = 142

/*

条码参数说明：

bind 选项，添加绑定条

hidetext 选项，不显示文字

show_hrt 选项，删除人类可读文本

bold 选项，使用粗体文本

box 选项，在符号周围加一个方框

cymk 选项，在 EPS 符号中使用 cmyk 颜色

dotty 选项，多蒂用点代替方块来表示矩阵符号

dmre 选项，允许数据矩阵矩形扩展

esc 选项，处理输入数据中的转义字符

gs1 选项，将输入视为与 gs1 兼容的数据

small 选项，在图像中使用小文本

square 选项，平方力数据矩阵符号为平方

type = n ,条码类型数(默认为 20 (=Code128))

bg = ffffff,指定背景色，十六进制方式，6 个字符长， ,默认白色

fg = 000000,指定前景色，十六进制方式，6 个字符长， ,默认黑色

align = left / right ， 文本居左，居右，默认居中

border = n, 边框的宽度

cols = n, 设置符号中的数据列数

rows = n, 设置符号中的数据行数

dotsize = n ,点在 dotty 模式下的数量设置半径

eci = n, 设置原始数据的 eci 模式

filetype = PNG/EPS/SVG/PNG/EPS/GIF/TXT，设置输出文件类型

height = n ,设置高度

mode = n,设置编码模式(Maxicode/组合)

output = out.png ,文件发送输出到文件。(缺省为 out.png)

primary = STRING Set 结构化主消息(Maxicode/Composite)

secure = NUMBER ,Set error correct level

scale = n,缩放比例,默认为 1

rotate = 0/90/180/270,旋转角度,(PNG/BMP/PCX)

vers = n,设置符号版本(二维码/Han Xin)

whitesp = n,维数倍数下空格的数量设置宽度

*/

function boolean OpenBarCode(readonly string codeText,string barType, int nHeight,double scale,readonly string options) system library "PbIdea.dll" alias for "imageOpenBarCode"

uo_barcode.OpenBarCode("12345678901234567890123456789012345678901234567890", “code128”, ai_height,1.0," hidetext ") 通过成不显示文本内容的 128 码

uo_barcode.save(…) 可另存为图片文件。

```
uo_json jsBar
jsBar = create uo_json
jsBar.set("width",180)
jsBar.set("height",24)
jsBar.set("text","12345678901234567890123456789012345678901234567890")
jsBar.set("fontsize",48)
jsBar.set("type",20) //或"code128"
//jsBar.set("type",58) //或"qrcode"
//jsBar.set("vers",5)
//jsBar.set("secure",3)
jsBar.set("show_hrt",1)
jsBar.set("scale",1.0)
img.OpenBarCode(jsBar)
destroy jsBar
```

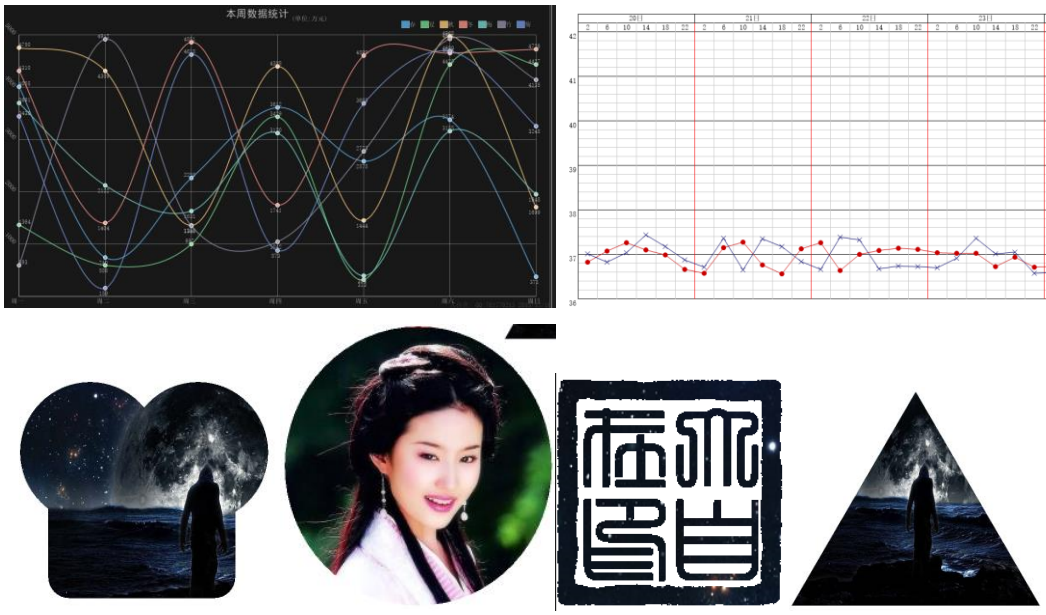
二、uo_painter 对象

在开始本章节前，我们先闭上眼睛做一个瞑想：假如你是一位画师，你要准备画画了，你得有一些基本元素，首先你得有一大张纸（或者是空白墙壁），上面有底色，或者是有现成的底案图纹，然后你往上面安置你想到的各种图片元素（各种类型的图片），或者各种图形元素（圆、矩、文字等），绘制结束后，你可以把成果保存起来，成为一幅画或图片文件。具体到 PC 里面，你会想到，这不就是 windows 操作系统自带的

“画图”工具吗？我们打开画图工具，可以拉伸图片大小、填底色、粘贴图片、放文字、放各种图形元素，最后我们还可以另存为各种格式的图片文件。

没错，uo_painter 就是 windows 操作系统自带的“画图”工具一样神奇工具，由于是用程序实现这些功能，可以通过计算机强大计算功能实现更加复杂的功能。接下来，我们按照上面暝想的步骤来实现一些基本功能。本章介绍所函数，都是 uo_painter 的成员函数，需要先对 uo_painter 实例化才能使用，例如：

```
uo_painter p;
p = create uo_painter;
p. SetCanvas(this,”ue_paint”);
destroy p;
```



（一）先了解图形的基本元素

1、**画板。**画板可以是窗口，可以是 HDC，可以是一张图，可以是内存的一块区域。

（1）对于窗口，绝大多数很好理解 ，就是把内容画到一个窗口上面，我们可以看得到显示效果。

（2）HDC 是什么呢？这个东西有点难理解，微软文档里就这个名词，也没有很好的办法去翻译这个缩写。我们可以把它看作是一个指向硬件的绘图句柄，它可以根据窗口句柄创建一个指向屏幕或桌面的绘图句柄，它也可以是根据打印机创建的一个指向打印机的绘图句柄，也可以是指向绘图仪或其他什么硬件的绘图句柄。我们常用的就 2 个：屏幕和打印机。同一句绘图语句，当 HDC 指向屏的时候，会显示显示到窗口，指向指印机时，操作系统会把这些内容输出到打印机上。

（3）一张图，可以是内存的一块区域。图片在内存里加载的时候，其实它就是一块连续的内存区域。也就是我们可以创建一张空白图，或者使用一张现有的图，只不过这些图不显示 ，只是加载在内存里，然后在这个图上进行绘图。就象画家在纸上作画一样。

2、**图片。**我们知道图片有各种格式，常见的有 bmp、jpg(jpeg)、png、gif 等等。各种图片格式，只是图片保存时使用，当一张图片加载到内存里后，实际上都是 bmp 点阵图，使用一块内存区域保存。uo_painter

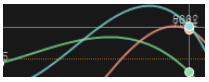
里使用 CacheImage/CacheImageBase64 函数，对图片加载，缓存到内存里面，缓存后返回一个唯一 ID 号代表被缓存的这张图，后面需要用到这张图时，以这人 ID 作为图片参数传入即可。用完后可以使用 RemoveImage 函数把这张缓存的图片从内存里释放掉。那么为什么我们要缓存图片到内存里呢，完全可以需要时直接传递文件名，从硬盘读取的呀？是的，这想法也没错。但我们为了追求高效率，就得使用缓存方式。比如在一个窗口的存在周期里，使用一张图上作为绘制背景，每次刷新时如果都要从硬盘读文件，这将是一个非常低效率的事情，甚至导致背景绘制停滞或者闪烁，而缓存到内存，就非常高效。

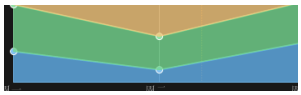
3、背景、前景色与透明度。这些颜色为大家所熟知。除了背景色与前景色，还有一个透明度。当使用 png 图片的时候，自然这张图片就有一部分是透明，一部分不透明，当把它叠加在其他图片上时，透明部分就不会遮住底图相应部分。此外，还有整图透明度形成上图与下图你中有我、我中有你的那种朦胧感。绘图函数中经常会需要指定背景色、前景色和透明度。

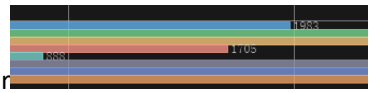
4、形状元素。

（1）点，是线的基础。多点可以连成线。函数：AddPoint

（2）直线，多直线可以连成固定形状。函数：DrawLines 

（3）弧线，DrawCurve 和 DrawBezier。 

（4）多边形，函数：DrawPolygon 

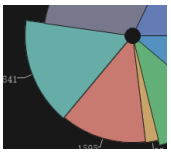
（5）矩形条状，函数 DrawBar 

（6）圆角矩形，函数 DrawRoundRect 

（7）菱形，函数 DrawRhombus

（8）箭头，函数 DrawArrow

（9）椭圆，函数：DrawEllipse

（10）扇形，其实也是圆形，函数 DrawPie 

5、文字。绘制文本，可以有前景色、背景色、字体名称、字号等各种属性，可以计算指定字体属性前提下的相应文本所占位置大小等等。函数 DrawText、CalcTextSize、InstallFontPlus

（二）准备一张“白纸”，为所有的绘图做准备。

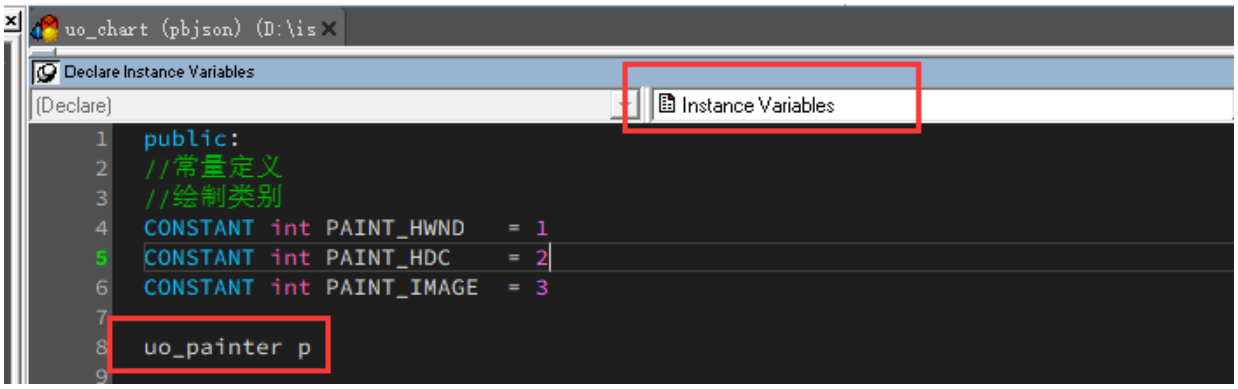
1、在窗口或控件上绘制。在窗口或控件上绘制内容，并不象我们想像的那么容易，下面我简称 UI 绘制。UI 绘制由操作系统负责调度，什么时候需要绘制，什么时候不需要绘制，完全由操作系统管理、控制，所

以，我们只能响应操作系统的绘制请求，做出正确绘制，才是有效的，否则绘制无效，看不到结果。

UI 有背景擦除时，操作系统会发送 WM_ERASEBKGND 消息，有刷新绘制需求时，会发送 WM_PAINT 消息。作为控件绘图，我们默认忽略 WM_ERASEBKGND 消息，只响应 WM_PAINT 消息就可以了。具体到 PB 里面，有个 pbm_paint 消息对应于 WM_PAINT。但是，这个消息与其他消息不同，这个消息只能响应一次，PB 内部因绘制需要已响应过一次，我们再去自定义事件响应这个消息是无效的。因此，uo_painter 提供了 SetCanvas 函数响应绘制消息。

步骤一、为窗口或可视自定义控件声明一个实例变量

uo_painter p;



步骤二、open 事件，或者的 Constructor 事件里对控件实例化，并且将当前 UI 对象变成可响应消息画板：

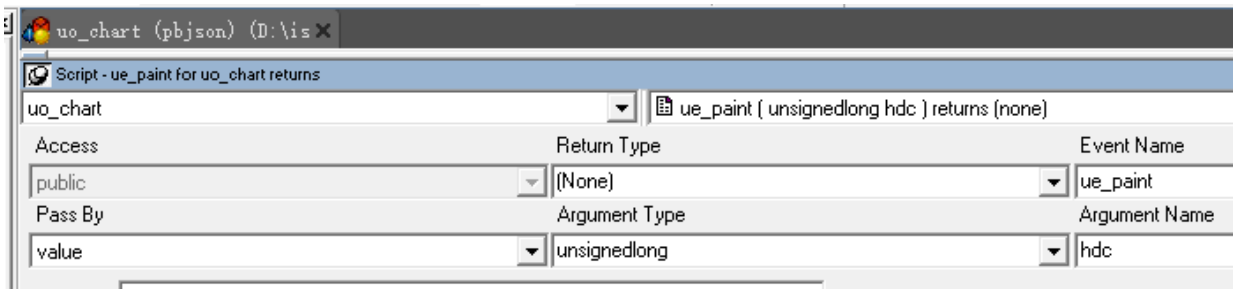
p = create uo_painter;

p. SetCanvas(this,"ue_paint");

步骤三、close 事件，或者的 Destructor 事件里销毁实例，释放资源

destroy p;

步骤四、然后再给这个窗口或控件添加一个事件，命名为 ue_paint，给它一个 ulong 类型的参数，命名为 hdc。



经过以上 4 个步骤，当前窗口或控件就具备在 ue_paint 事件里响应绘制动作，并且显示绘制效果即可。我们可以直接使用 hdc 参数：

p.BeginPaint(hdc,RGB(255,255,255))　//以白色背景开始绘制

…… //接下来可以完成自己的绘制任务

由于 Uo_painter 采用了双缓存绘图机制，这时候绘制的内容实际上是在内存里创建的一个虚拟屏幕上，我们要将内容显示到实际 hdc 上，最后还必要须：

p. EndPaint() //将绘制内容刷新到物理屏幕上，形成可视效果

2、在现在图片上绘制。前面我们说过缓存存图，在这个基础上进行图片绘制。可以在一张图片上添加

水印、文字等。

```
Long li_image
Li_image = p.CacheImage(“back.bmp”) //加载一张图开缓存
p. BeginPaintImage(li_image) //以上面加载的图为底图，开始绘制
…… //接下来可以完成自己的绘制任务
p. SaveMemoryImage (“new.bmp”) //将内容另存为图，释放资源
```

或者

```
Blob pic
Pic = p.GetMemoryImage(“jpg”) //以 JPG 格式返回 BLOB 数据，自己另行使用。
p.RemoveImage(li_image) //原背景不用了要释放掉
```

3、开启空白图片进行绘制。可以开启一张底色为指定颜色的空白图片，然后在这个基础上绘制，就和系统的画图工具一样的功能。

```
p. BeginPaintImage(400,600,RGB(255,255,255)) //开启一张宽 400 高 600 背景色为白色的空白图
…… //接下来可以完成自己的绘制任务
```

或者

```
Blob pic
Pic = p.GetMemoryImage(“jpg”) //以 JPG 格式返回 BLOB 数据，自己另行使用。
```

（三）绘图命令

具体绘图命令，就不一一介绍了，可以直接打开 uo_painter 对象，切换到 Local External Functions 看函数声明，都有具体中文注释。

有一点请注意，图片处理同时支持文件操作和内存 BLOB 操作，不是必须要存为文件或者从文件加载。甚至直接支持打开图片 base64 编码，内部自动进行解码还原为图片数据。

（四）一些常用功能示例操作

1、图片格式转换

```
uo_painter p; p = create uo_painter
long image
image = p.CacheImage(“test.bmp”) //加载原图
p.SaveImage(image,”test.png”) //另存为 png 格式
p.SaveImage(image,”test.jpg”) //另存为 jpg 格式
p.RemoveImage(image)
destroy p
```

2、图片缩放

```
uo_painter p; p = create uo_painter
long image1,image2
image1 = p. CacheImageThumb (“test.bmp”,200,0.5) //加载图片，透明度为 200，宽和高缩小 50%
image2 = p. CacheImageThumb (“test.bmp”,200,1.5) //加载图片，透明度为 200，宽和高放大 150%
p.SaveImage(image1,”test.png”)
p.SaveImage(image,2”test.jpg”)
p.RemoveImage(image1)
p.RemoveImage(image2)
destroy p
```

3、图片拼接

将 a.bmp 和 b.bmp 按左右方式进行拼接，生成新图

```
uo_painter p; p = create uo_painter
long a,b
int width,height,wa,ha,wb,hb
a = p. CacheImage ("a.bmp") //加载图片
b = p. CacheImage ("b.bmp") //加载图片
p.GetImageSize(a,wa,ha) //获取 A 图片大小
p.GetImageSize(b,wb,hb) //获取图片大小
width = wa + wb //计算合并后的宽度
height = ha
if ha < hb then height = hb //以最大高度为新图片的高度
p.BeginPaintImage(width,height,RGB(255,255,255)) //开启一张新图
p.DrawImage(0,(height - ha)/2,wa,ha,a,0) //把 A 图片绘制到指定位置
p.DrawImage(wa,(height - hb)/2,wb,hb,b,0) //把 B 图片绘制到指定位置
p.SaveMemoryImage ("c.png") //合并后的图片另存为 c.png
p.RemoveImage(a)
p.RemoveImage(b)
destroy p
```

合并后效果如下：

a.bmp


b.bmp

c.png

4、图片拼接 2

Uo_painter 也提供了现在拼接函数

uo_painter p; p = create uo_painter

p. MergeImageInit(8, 0, true, 2, 1, RGB(255,255,255), “back.jpg”, 180, false,”c.png”)//边距为 8，边距有效，2 行，2 列，背景为白色，背景图片 back.jpg 透明度为 180，不进行缩放，拼完后存为 c.png

p. MergeImageAdd(1, 1, 1, 1, “a.bmp”,255,0)　//添加 b.bmp 放到第 1 行第 1 列，只占 1 行 1 列范围

p. MergeImageAdd(2, 1, 1, 1, “hd.png”,255,0) //添加 b.bmp 放到第 2 行第 1 列，只占 1 行 1 列范围

p. MergeImage()//完成合并

destroy p

效果如下图

a.bmp

c.png


hd.png

5、图片上写字和加水印

uo_painter p; p = create uo_painter

long image,water,dzz

int width,height

image = p.cacheImage("tmp/f.jpg")

water = p.cacheImage("tmp/hd.png")

dzz　　= p.cacheImage("tmp/dzz.png")

p.GetImageSize(image,width,height)

p.BeginPaintImage(image)

p.DrawImage(200,236,64,64,water,0)

p.DrawImage(8,height - 84,84,84,dzz,0)

p.DrawText(0,8,width,32,"漂亮美女人人爱","黑体",28,RGB(250,250,250),1)

p.DrawText(0,height　-　38,width,32,string(datetime(today(),now()),"yyyy-mm-dd　hh:mm_ss")," 黑　体

",28,RGB(255,0,0),2)

p.SaveMemoryImage ("tmp/new.png")

p.removeImage(image)

destroy p

效果图如下：

f.jpg

new.png

6、更多其他功能，请加参考 PbIdea DEMO 之 uo_painter 部分。

大自在 于 2022 年元旦假日

QQ:781770213 QQ 群:624409252