InkCanvas 是非常有用的控件： 可以实现电子签名

<InkCanvas x:Name="inkCanvas1"

HorizontalAlignment="Left"

Height="199"

Margin="0,200,0,0"

VerticalAlignment="Top"

Width="284"

Background="#FFF1D0AF">

<InkCanvas.DefaultDrawingAttributes>

<DrawingAttributes Color="#FFF31B1B" Width="5"/>

</InkCanvas.DefaultDrawingAttributes>

</InkCanvas>

保存成图片文件：

DrawingVisual dv = new DrawingVisual();

using (DrawingContext dc = dv.RenderOpen())

{

VisualBrush vb = new VisualBrush(inkCanvas1);

dc.DrawRectangle( vb, null,

new Rect(new System.Windows.Point(),

new System.Windows.Size(

(int)inkCanvas1.Width,

(int)inkCanvas1.Height

)

)

);

}

RenderTargetBitmap rtb = new RenderTargetBitmap(

(int)inkCanvas1.Width,

(int)inkCanvas1.Height,

96d, 96d,

PixelFormats.Default

);

rtb.Render(dv);

using (FileStream file = new FileStream("lwh1.jpg", FileMode.Create, FileAccess.Write))

{

JpegBitmapEncoder encoder = new JpegBitmapEncoder();

encoder.Frames.Add(BitmapFrame.Create(rtb));

encoder.Save(file);

}

添加照片或者其他控件到画布上：

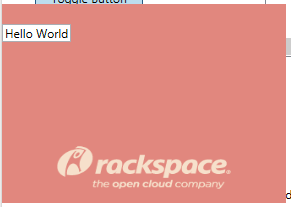


Image mm = new Image {

Width = (int)inkCanvas1.Width,

Height = (int)inkCanvas1.Height,

Source = new BitmapImage(

new Uri(@"http://s0.2mdn.net/viewad/4361691/ITconcerns\_336x280.gif", UriKind.RelativeOrAbsolute))

};

mm.Opacity = 0.35d;

mm.Stretch = Stretch.Fill;

inkCanvas1.Children.Add(mm);

TextBox cct = new TextBox();

cct.Visibility = System.Windows.Visibility.Visible;

cct.Text = "Hello World";

inkCanvas1.Children.Add(cct);

InkCanvas.SetTop(cct, 20);

InkCanvas.SetTop(cct, 20);



清除控件和手写画笔:

this.inkCanvas1.Children.Clear();

this.inkCanvas1.Strokes.Clear();

主要的几种模式：

this.inkCanvas1.EditingMode = InkCanvasEditingMode.EraseByStroke;

this.inkCanvas1.EditingMode = InkCanvasEditingMode.EraseByPoint;

this.inkCanvas1.EditingMode = InkCanvasEditingMode.Ink;

主要的几种模式：

this.inkCanvas1.EditingMode = InkCanvasEditingMode.EraseByStroke;

this.inkCanvas1.EditingMode = InkCanvasEditingMode.EraseByPoint;

this.inkCanvas1.EditingMode = InkCanvasEditingMode.Ink;

设置背景色

this.inkCanvas1.Background = new SolidColorBrush(Color.FromArgb(0, (byte)0, (byte)0, (byte)0));

new SolidColorBrush(Color.FromArgb(0xff, 0xff, 0x90));

以上是透明背景色

如果想保存透明效果， 图片格式选择 PNG

DrawingVisual dv = new DrawingVisual();

using (DrawingContext dc = dv.RenderOpen())

{

VisualBrush vb = new VisualBrush(inkCanvas1);

dc.DrawRectangle(vb, null, new Rect(new System.Windows.Point(), new System.Windows.Size((int)inkCanvas1.Width, (int)inkCanvas1.Height)));

}

RenderTargetBitmap rtb = new RenderTargetBitmap((int)inkCanvas1.Width, (int)inkCanvas1.Height, 96d, 96d, PixelFormats.Default);

rtb.Render(dv);

using (FileStream file = new FileStream("lwh1.png", FileMode.Create, FileAccess.Write))

{

PngBitmapEncoder encoder = new PngBitmapEncoder();

encoder.Frames.Add(BitmapFrame.Create(rtb));

encoder.Save(file);

}

更改画笔

 DrawingAttributes inkDA = new DrawingAttributes();

inkDA.Width = 5;

inkDA.Height = 5;

inkDA.FitToCurve = true;

inkDA.Color = Color.FromArgb(255, 255, 255, 255);

this.inkCanvas1.DefaultDrawingAttributes = inkDA;

如何获取电脑屏幕

public void GetScreenSnapshot()

{

Bitmap BitmapScreenShot;

Graphics GraphicsScreenShot;

BitmapScreenShot = new Bitmap(

System.Windows.Forms.Screen.PrimaryScreen.Bounds.Width,

System.Windows.Forms.Screen.PrimaryScreen.Bounds.Height,

System.Drawing.Imaging.PixelFormat.Format32bppArgb

);

GraphicsScreenShot = Graphics.FromImage(BitmapScreenShot);

GraphicsScreenShot.CopyFromScreen(

System.Windows.Forms.Screen.PrimaryScreen.Bounds.X, System.Windows.Forms.Screen.PrimaryScreen.Bounds.Y, 0, 0,

System.Windows.Forms.Screen.PrimaryScreen.Bounds.Size,

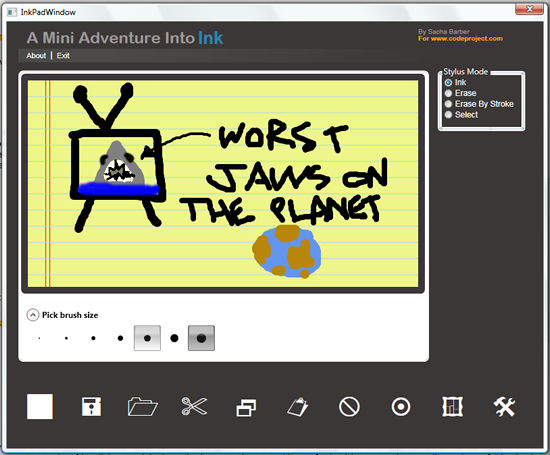
CopyPixelOperation.SourceCopy

);

BitmapScreenShot.Save("screen.png", System.Drawing.Imaging.ImageFormat.Png); BitmapScreenShot.Dispose();

}

其他文摘：

你可以在这里下载到它 <http://files.cnblogs.com/zhouyinhui/InkCanvas.rar>  
  
[](http://files.cnblogs.com/zhouyinhui/InkCanvas.rar)  
  
下面是InkCanvas的常见用法(假设我们的InkCanvas对象为inkCanv):

**保存**你可以选择将墨迹保存为墨迹文件或者保存为图片  
  
保存为墨迹文件(\*.isf):

http://www.cnblogs.com/Images/OutliningIndicators/None.gifthis.inkCanv.Strokes.Save(file);

其中file为要保存到的文件流(FileStream)  
  
保存为图片:  
我们可以先将墨迹绘制到图片上再将图片保存,其中要将Visual对象绘制到图片上可以使用RenderTargetBitmap类,并采用其Render()方法.要将图片保存可以使用BitmapEncoder类,并调用其Save()方法.

http://www.cnblogs.com/Images/OutliningIndicators/None.gif RenderTargetBitmap rtb = new RenderTargetBitmap(width,height, 0, 0, PixelFormats.Default);  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif rtb.Render(this.inkCanv);  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif BmpBitmapEncoder encoder = new BmpBitmapEncoder();  
http://www.cnblogs.com/Images/OutliningIndicators/None.gifencoder.Frames.Add(BitmapFrame.Create(rtb));  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif  encoder.Save(file);

**打开**  
  
要打开已保存的墨迹文件(\*.isf),可以使用下面的方法:

http://www.cnblogs.com/Images/OutliningIndicators/None.gifthis.inkCanv.Strokes = new StrokeCollection(file);

**选定**  
  
向画图板一样,你可以选定绘制的某些元素,选定有两种方式,一是用户使用鼠标进行选定,另一种是程序使用代码选择指定的墨迹.  
  
使用鼠标选定  
很简单地,你只需要切换InkCanvas的编辑模式就可以了.InkCanvas的EditingMode属性是InkCanvasEditingMode枚举类型,你可以将其值切换为InkCanvasEditingMode.Select以便进入选定模式.  
  
使用代码选定  
你可以调用InkCanvas的Select()方法

**复制、剪切、粘贴或删除选定的对象**  
  
这几个操作都比较简单,InkCanvas有对应的方法,分别是CopySelection(),CutSelection(),Past()与Remove().

**橡皮擦**要使用橡皮擦,与切换选定模式一样,同样是切换InkCanvas的EditingMode属性,擦除方式有两种,分别是InkCanvasEditingMode.EraseByPoint(按点擦除)与InkCanvasEditingMode.EraseByStroke(按墨迹笔画擦除,一次擦掉不间断的一笔).

**笔迹颜色、粗细...**  
  
要更改笔迹的相关属性,可以设置InkCanvas的DefaultDrawingAttributes属性,比如:

http://www.cnblogs.com/Images/OutliningIndicators/None.gif            DrawingAttributes inkDA = new DrawingAttributes();  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif            inkDA.Width = 5;  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif            inkDA.Height =5;  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif            inkDA.Color = Color.FormArgb(255,255,255,255);  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif            this.inkCanv.DefaultDrawingAttributes = inkDA;

InkCanvas还有很多有趣的功能,你可以参考SDK中的InkCanvas类以获得更多.

**基于InkCanvas实现的桌面涂鸦工具-[ WPF开发 ]**

时间:2011-04-21 03:16来源:博客园 作者:Ellic 点击:2206次

首先简单的介绍下InkCanvas，简单的来说，InkCanvas就是在WPF中 实现允许使用墨迹的布局控件。实际上，InkCanvas有着更多层面上的应用，它的主要目的是(通过鼠标或者和指示笔)捕捉笔迹。InkCanvas从 技术上说不是一个控件，因为它直接从FrameworkElement继承而来，但是它的行为和控件非常像(但不能用一个新的模板来改变它的

　　首先简单的介绍下InkCanvas，简单的来说，InkCanvas就是在WPF中实现允许使用墨迹的布局控件。实际 上，InkCanvas有着更多层面上的应用，它的主要目的是(通过鼠标或者和指示笔)捕捉笔迹。InkCanvas从技术上说不是一个控件，因为它直接 从FrameworkElement继承而来，但是它的行为和控件非常像(但不能用一个新的模板来改变它的样式)。

　　默认模式下，InkCanvas允许在它的表面上进行简单的书写和画图。当使用指示笔时，笔尖用来写、笔端用来擦。每一个笔画被捕捉为一个 System.Windows.Ink.Stroke对象，保存在InkCanvas的Strokes集合中。但是InkCanvas也支持在 Children集合(一个内容属性)中保留任意数量的UIElement元素。这样很容易通过墨水(ink)来注释任何东西。如下面的代码，我们可以很 容易的生成一个InkCanvas画板。

<Window x:Class="WPFTEST.InkCanvas"  
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
    Title="InkCanvasSample" Width="220" Height="87">  
        <InkCanvas>  
             <InkCanvas.DefaultDrawingAttributes>                
                 <DrawingAttributes Color="Red" />  
            </InkCanvas.DefaultDrawingAttributes>  
        <Image Source="E:\Cnblogs\wwf.png"/>  
        </InkCanvas>  
</Window>

　　InkCanvas 支持几种模式，它们能利用EditingMode属性被独立地应用到指示笔尖(或鼠标)，并通过EditingModeInverted属性来应用于指示 笔的末端(back end)。只读的ActiveEditingMode属性可以告诉你哪一个属性当前正在被使用。所有这3个属性都是 InkCanvasEditingMode类型的，它有以下几种值：

　　1、Ink(EditingMode的默认值) —— 通过鼠标或者指示笔来绘制笔画。

　　2、InkAndGesture —— 和Ink一样，但同样可以识别用户的手势。手势的列表(Up、Down、Circle、ScratchOut和Tap)保存在System.Windows.Ink.ApplicationGesture枚举类型中。

　　3、GestureOnly —— 只识别手势，不会绘制用户输入的笔画。

　　4、EraseByStroke (EditingModeInverted的默认值)—— 当笔画被触及时将笔画擦掉。

　　5、EraseByPoint —— 只擦掉直接碰及到的笔画部分(就像传统的铅笔橡皮)。

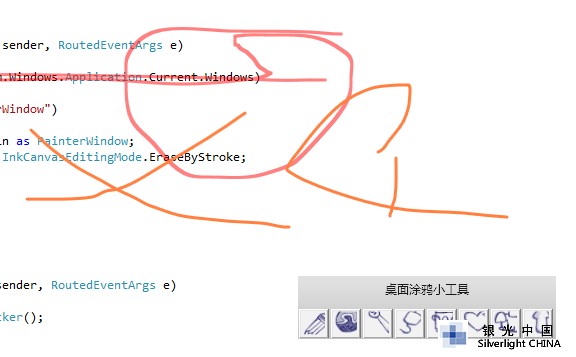
　　6、Select —— 当被触及时，选择笔画或者任何UIElement，使它们能被删除、移动或者在InkCanvas范围内被调整尺寸。

　　7、None —— 对于鼠标或者指示笔不做任何响应。

　　一些普通元素与墨水没有任何关系，如果在这些元素上使用Select模式将非常有趣，因为它自动会提供一个“穷人”的运行时设计界面用来排列控 件。InkCanvas还定义了15种事件，其中包括改变编辑模式、改变/移动/调整选择、收集或者擦除笔画，以及执行手势。当然，在应用程序中使用墨水 比在人脸上画胡子还是要复杂些!你经常要对一个笔画集合做手写识别，如果输入的是字符你就可以分析出它。WPF拥有内建的手势识别功能，但没有手写识别引 擎。

　　概述

　　程序截图如下:



　　大概的一个思路就是主要有两个操作窗口，一个是主窗口，可以进行新建涂鸦、调整画笔彩色、画笔形状等等，另外一个就是涂鸦的画板了。首先先截图，然后设置Inkcanvas的Background属性为该图片，然后就可以在上面进行画图了。主要要处理的细节问题在于：

　　截图的问题

　　取得窗口坐标的问题

　　窗口之间传值的问题

　　把这些细节处理好就很容易就可以实现了。

　　代码解释

　　首先是截图的问题。查MSDN，我找不到WPF里有现在的API可以调用，只能通过winForm来实现截图，代码如下：

01 public Bitmap GetScreenSnapshot()  
02 {  
03      System.Drawing.Rectangle rc = SystemInformation.WorkingArea;  
04      var bitmap = new Bitmap(rc.Width, rc.Height, System.Drawing.Imaging.PixelFormat.Format32bppArgb);  
05   
06      using (Graphics g = Graphics.FromImage(bitmap))  
07      {  
08          g.CopyFromScreen(rc.X, rc.Y, 0, 0, rc.Size, CopyPixelOperation.SourceCopy);  
09      }  
10   
11      return bitmap;  
12 }  
13   
14 public BitmapSource ToBitmapSource(Bitmap bmp)  
15 {  
16      BitmapSource returnSource;  
17   
18      try  
19      {  
20          returnSource = System.Windows.Interop.Imaging.CreateBitmapSourceFromHBitmap(bmp.GetHbitmap(), IntPtr.Zero, Int32Rect.Empty, BitmapSizeOptions.FromEmptyOptions());  
21      }  
22      catch  
23      {  
24          returnSource = null;  
25      }  
26      return returnSource;  
27   
28 }  
29 //截取整个屏幕作为画布，并开始画画  
30 private void pencil\_Click(object sender, RoutedEventArgs e)  
31 {  
32      //初始化画布  
33      PainterWindow pw = new PainterWindow();  
34      System.Drawing.Rectangle rc = SystemInformation.WorkingArea;  
35      pw.Width = rc.Width;  
36      pw.Height = rc.Height;  
37      pw.ink.Width = rc.Width;  
38      pw.ink.Height = rc.Height;  
39      //截图  
40        this.Hide();  
41      Bitmap bt = GetScreenSnapshot();  
42      BitmapSource bs = ToBitmapSource(bt);  
43   
44      System.Windows.Controls.Image img = new System.Windows.Controls.Image();  
45      img.Source = bs;  
46      pw.ink.Background = new ImageBrush(bs);  
47      pw.Show();  
48      this.Show();  
49   
50 }

　　看到上面的代码，通过System.Drawing.Rectangle rc = SystemInformation.WorkingArea;来获取当前系统工作窗口(不包括任务栏)的信息，包括分辨率的大小，各个点的坐标信息，获 取这些信息后，新建一个bitmap，通过winForm里的CopyFromScreen函数我们把屏幕绘画在新建的那个bitmap上，然后获得 bitmap的地址设置为BitmapSource，并设置ink的Background为ImageBrush(bs)即可。

　　下面是响应颜色选择的代码：

01 private void color\_Click(object sender, RoutedEventArgs e)  
02 {  
03     ColorDialog cd = new ColorDialog();  
04     //取得颜色  
05     if (cd.ShowDialog() == System.Windows.Forms.DialogResult.OK)  
06     {  
07         //要去了解这两种颜色的不同之处  
08         inkDA.Color = System.Windows.Media.Color.FromArgb(cd.Color.A, cd.Color.R, cd.Color.G, cd.Color.B);  
09         inkDA.Height = 3;  
10         inkDA.Width = 3;  
11         inkDA.FitToCurve = true;  
12         //修改颜色  
13         foreach (Window win in System.Windows.Application.Current.Windows)  
14         {  
15             if (win.Title == "PainterWindow")  
16             {  
17                 PainterWindow pw = win as PainterWindow;  
18                 pw.ink.DefaultDrawingAttributes = inkDA;  
19                 pw.Show();  
20             }  
21         }  
22   
23     }  
24   
25   
26 }

　　要注意的是System.Widows.Media.Color与InkCanvas定义的颜色的不同，这中间需要通过一个函数来转换。

　　其实想着能把它完善好的，像形状的自定义，界面的细节调整，等。最近有太多的事情要做，所以先搁置下来，待时间过后再回来看自己的代码，熟悉下。

　　相关资料

　　一些关于InkCancas的资料

[WPF中InkCanvas(墨水面板)用法](http://tech.ddvip.com/2008-11/122569914788441.html)

[分享WPF实现屏幕截图程序详解](http://tech.ddvip.com/2008-11/122569914788441.html)

[桌面涂鸦工具下载(需要.net支持，涂鸦过程按Esc键退出涂鸦界面。)](http://files.cnblogs.com/libenqing/DestopInk.rar)

本文来自Ellic的博客，原文地址:http://www.cnblogs.com/libenqing/archive/2011/04/21/inkcanvas-sample.html