第16卷 第12期
2006年12月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol.16 No.12
Dec. 2006

在C#中用GDI+实现图形动态显示

闫宇晗,常 鑫
(北京跟踪与通信技术研究所,北京 100094)

摘 要:GDI+,即图形设备接口,它作为一个类库将数据转换为与图形设备兼容的形式。Microsoft Visual C#是一种简单、现代、类型安全和面向对象的语言,它为程序员提供了一个可以开发运行在Windows和其他平台上的几乎所有程序的环境。在C#中利用GDI+可方便快捷地实现应用程序与图形设备的交互。为适应复杂工程需求,更加灵活多态地显示图形,文中论述了在C#中用GDI+绘制图形的基本方法,以及如何实现图形动态显示。

关键词:C#;GDI+;动态显示
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2006)12-0117-02

Implementation of Dynamic Graphical Display Using GDI+ in C#

YAN Yu-han, CHANG Xin
(Beijing Institute of Tracking and Telecommunication Technology, Beijing 100094, China)

Abstract: GDI+ is graphics device interface+. It is a class libraries. It transforms data to a form which is compatible with graphics device. Microsoft Visual C# is object oriented language which is simple, modern and secure. It provides us with environment which is able to exploit program that run at Windows and other flat. Therefore, with the use of GDI+, it will achieve the alternation conveniently and quickly between application and graphics device in C#. This paper will introduce approaches to implementation of dynamic graphical in C# so as to meet with the requirements of complex engineering and more flexible display.

Key words: C#; GDI+; dynamic display

0 引 言

GDI+(Graphics Device Interface+)作为公共语言运行库,是Windows图形设计界面(GDI)的高级实现,它已完全替代GDI,拥有较好的性能和易于使用的特点。利用它可以创建图形、绘制文本以及将图形图像作为对象操作,并且可在Windows窗体和控件上呈现图形图像。虽然针对Web窗体无法直接使用,但可以通过Web服务器"图像"(Image)控件显示图形图像。目前,GDI+是在Windows窗体应用程序中以编程方式呈现图形的惟一方法[1]。

Microsoft Visual C#从C和C++演变而来,它是一种简单、现代、类型安全和面向对象的语言,还具有强大的图形、图像功能,能够快捷、方便地开发图形设计、图像处理及多媒体技术的Windows应用程序[2~4]。所以,两者结合(即以Visual C#为平台,采用新的功能更强的GDI+在屏幕上绘制图形、文字或显示图像)的方式已越来越广泛地受到程序员的青睐。但是,为了满足各种复杂工程的需要,仅作单纯的显示是不够的,文中针对这一点,简单介绍C#中用GDI+实现图形动态显示的方法。

收稿日期:2006-02-22
作者简介:闫宇晗(1979-),女,河南开封人,助理工程师,研究方向为计算机软件开发与数据库。

1 GDI+简介

1.1 GDI+的概念

GDI+作为GDI(Windows早期版本提供的图形设备接口)的后续版本,是一种应用程序编程接口(API),通过一套部署为托管代码的类来展现,这套类被称为GDI+的"托管类接口"。此接口允许程序员编写与打印机、监视器或文件等图形设备进行交互的Windows和Web图形应用程序,并显示在屏幕或由打印机输出。其实现途经是通过调用GDI+类提供的方法,此方法又反过来调用相应特定设备的驱动程序,进而实现图形在屏幕或其它特定设备上的显示。GDI+将应用程序与图形硬件隔离,正是这种隔离允许开发人员创建与设备无关的应用程序。

GDI+提供的服务分为3个大类:二位矢量图形、图像处理和版式。

1.2 GDI+的服务

(1)二维矢量图形。矢量图形包括坐标系统中的系列点指定的绘图基元(例如直线、曲线和图形)。它的程序设计是指绘制可以由坐标系中的一组点确定的形状,即:图元。在GDI+中,一个类对象或结构体表示一个图元,每个类或结构体都提供了可用于获取和设置图元属性的成员(如:Point结构体)。另外,设计二维矢量图形的方法分为常规设计和高级设计。

(2)图像处理。图像处理是指对图像的查看和操作。

在 GDI+ 中,其基本功能是在 Image 类中定义的,它提供了加载、创建和保存图像的成员,其基类 Bitmap 和 Metafile 还定义了显示、操作和保存位图和图元文件的功能。例如 Metafile 类,可用于记录、显示和保存图元文件;MetafileHeader 和 MetaHeader 类,用于检查图元文件头中存储的数据。

(3)版式。版式是指文本和设计的外观,关系到使用何种字体、字号和样式来显示文本。GDI+ 为这种复杂任务提供了大量的支持。例如:创建和使用字体的类,子像素消除锯齿等[1,5]。

2 GDI+ 画图方法

在 Windows 应用程序中,一个窗体就是一个绘制表面。为了在窗体上绘制出合适的图形,首先,需要创建 Graphics 对象,即 GDI+ 绘图表面,它是用于创建图形图像的对象。然后才可以使用 GDI+ 绘制线条和形状、呈现文本或显示与操作图像。最后,把画好的图形呈现在窗体上。下面就介绍具体实现方法。

2.1 创建 Graphics 对象

创建 Graphics 对象。可通过多种方式获得一个与窗体关联的 Graphics 对象,这里介绍其中常用的 3 种。

(1)使用窗体的 Paint 事件。

接收对图形对象的引用,该对象为窗体或控件 Paint 事件中 PaintEventArgs 的一部分。在为控件创建绘制代码时,通常会使用此方法来获取 Graphics 对象。

(2)使用 CreatGraphics 方法。

调用某控件或窗体的 CreateGraphics 方法来获取对 Graphics 对象的引用,该对象表示该控件或窗体的绘图表面。在已存在的窗体或控件上绘图,则可使用此方法。

(3)从 Image 对象创建。

从继承自图像的任何对象创建 Graphics 对象。此方法在需要更改已存在的图像时十分有用。

2.2 绘制图像

可以使用 GDI+ 在应用程序中呈现以文件形式存在的图像。实现此操作的方法是:创建某 Image 类(如 Bitmap)的一个新对象和一个 Graphics 对象(它表示要使用的绘图表面),然后调用 Graphics 对象的 DrawImage 方法把 Image 类的新对象填充入 Graphics 对象中,用画笔等其它工具在 Graphics 对象中绘制图形,运行时使用 GDI+ 呈现它们。具体来讲,有以下两种常用方法:

(1)创建一个对象,该对象表示要显示的图像。该对象必须是从 Image(如 Bitmap 或 MetaFile)继承的类的成员。例如:

```
Bitmap myBitmap = new Bitmap();
g = Graphics.FromImage(myBitmap);
```

(2)创建一个 Graphics 对象,该对象表示要使用的绘图表面。例如:

```
Graphics g = this.CreateGraphics();
```

2.3 呈现图像

当图形绘制完毕,需要呈现在窗体上时就用到 Form 类提供的 CreatGraphics,此方法返回一个 Graphics 对象。另外,在软件需要一个按钮或菜单的 click 事件处理程序中进行绘制操作时,也可使用此方法。代码如下:

```
Graphics g = this.CreateGraphics();
g.DrawImage(Bitmap, 0, 0, Windows_Width, Windows_Height);
```

除了以上介绍的方法外,还可以使用 Graphics 类提供的 FromImage,FromHwnd,FromHdc 等静态方法,分别从图像、窗口句柄和设备上下文的窗口句柄创建 Graphics 对象。总之,在窗体上画图是建立在创建 Graphics 对象的基础之上的,任何操作都要通过 Graphics 实现。

3 动态显示图形的方法

在各种试验或者工程建设中,经常需要模拟显示某些状态,那么,如何实现画面的动态显示呢? 简单来讲,按照显示频率的快慢提取坐标值,显示在界面上即可。下面就以动态显示某公交线路走向为例具体讨论其实现的方法。

步骤如下:

1)定义两个 Graphics 绘图面(以下简称 g1,g2)。

2)以北京市地图为图像源,定义两个位图 Bitmap1,Bitmap2。

```
Bitmap1 = new Bitmap(BeiJingMap.bmp);
Bitmap2 = new Bitmap(BeiJingMap.bmp);
```

把 Bitmap1,Bitmap2 分别填充入 g1,g2 中,以便接下来进行绘制操作。

```
g1 = Graphics.FromImage(Bitmap1);
g2 = Graphics.FromImage(Bitmap2);
```

3)利用 C# 中的计时器控件,以 t 秒为时间间隔循环取出行车线路的坐标值(X0,Y0),(X1,Y1),使用 g1.DrawLine 函数连线两点。

到此就可大致实现线路的动态显示。但是,这种简单的显示在工作实践中往往显得不够丰富。如果需要显示两个或两个以上的动态点又该如何呢? 比如,在运动的轨迹前端加上汽车图标。这就需要在以上画图的基础上再添加以下步骤:

4)把已经画有轨迹的 Bitmap1 利用 DrawImage 函数填充入 g2 绘图面。即:

```
g2.DrawImage(Bitmap1, 0, 0, Bitmap1.Width, Bitmap1.Height);
```

5)在 g2 绘图面上绘制图标。即:

```
g2.DrawIcon(iconBus, X0, Y0);
```

6)创建一个用于呈现在窗口上的 Graphics(以下简称 g)。

```
Graphics g = this.CreateGraphics();
```

把绘制好的图像 Bitmap2(对 g2 的操作相当于改变位

(下转第 232 页)

书包含主动节点的名称)。服务器端存放了合法主动节点表及其权限等级,同时也存放主动节点黑名单表。

若主动节点在黑名单内,则直接丢弃该包不做任何处理。若不在黑名单内,到主动节点权限表中查询该主动节点的权限,若有,取出权限;若没有,按默认权限处理。

检查完主动节点的权限后,再检查主动节点所请求的可载入库。对服务器上不存在的可载入库,直接返回一个没有此可载入库的信息;对存在的可载入库,首先查询可载入库的权限等级表。

在服务器上存放一张可载入库的等级表:

只有大于或等于可载入库的权限等级的主动节点才有权请求此可载入库。

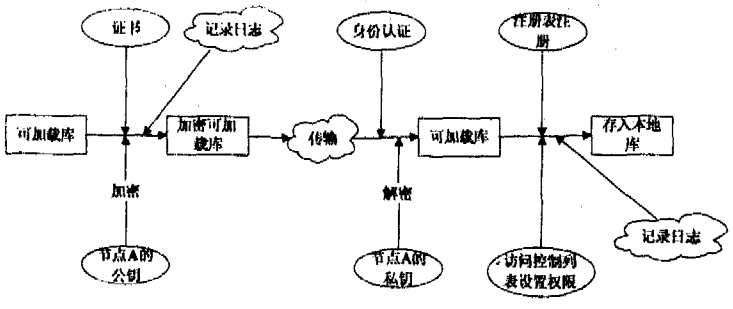

图 4 动态载入库传输过程

对于主动节点请求的可载入库,若主动节点的权限不够,则返回一个无法请求下载的信息。若权限允许,则服务器端将此可载入库加密后传送给该主动节点。

同样,主动节点也根据服务器返回的数字证书来验证服务器的身份,原理同上。

4 结 论

人们在 SENCOMM 平台上应用此安全模型后,得到了如下的安全性:

* 预防非法的或没有权限的主动节点请求可载入库;
* 预防假冒的可加载服务器传播恶意可载入库;
* 可载入库在传输过程中不被修改;
* 可载入库加密传输,即使被截获也不能被破解;
* 使有相应权限的主动节点只能请求相应权限的可载入库。

从以上结论中可以看出,在主动网络中,这种安全传输模型可以较好地解决信息传输双方的身份认证和信息安全传输的问题。当然,此安全模型所获得的安全性在一定程度上牺牲了传输效率,在以后的工作中将在安全性和效率性方面不断完善。

参考文献:

[1] Tennenhouse D L, Smith J M, Sincoskie W D. A survey of active network research[J]. Communications Magazine, IEEE, 1997, 35(1): 80-86.

[2] Jackson A W, Sterbenz J P G, Condell M N, et al. SENCOMM Architecture [DB/OL]. 2001-04. http://www.ir.bbn.com/projects/sencomm/sencomm-index.html.

[3] Calvert K L. Architectural Framework for Active Networks Version 1.0[S]. DARPA Active Network architecture document, 1999.

[4] 徐迎晓. Java 安全性编程实例[M]. 北京:清华大学出版社, 2003: 112-216.

[5] Rivest R. RFC 1321. The MD5 Message-Digest Algorithm [S]. [s.l.]: MIT Laboratory for Computer Science and RSA DATA Security, Inc, 1992.

[6] Condell M N, Hain R R. SENCOMM Programmer's API [DB/OL]. 2001-04. http://www.ir.bbn.com/projects/sencomm/sencomm-index.html.

[7] Hain R R, Comdell M N. Writing a Probe in the SENCOMM Environment [DB/OL]. 2001-04. http://www.ir.bbn.com/projects/sencomm/sencomm-index.html.

(上接第 118 页)

图 Bitmap2)通过 g 呈现在窗体上(x_Width, y_Height 表示窗体的长宽)。

g.DrawImage(Bitmap2, 0, 0, x_Width, y_Height);

总的来讲,实现本例的思想是:创建两个绘图面,在一个绘图面(g1)上绘制轨迹后再"贴在"另一个绘图面(g2)上,并在 g2 上绘制图标,当进入下一个循环时新的轨迹图又再次贴在 g2 上,然后再在 g2 绘制图标,这样虽然每一个循环都是先画轨迹再画图标,但是显示在窗体上始终只有运动的轨迹和随轨迹运动的图标。到此即可较为丰富地实现轨迹动态显示。

4 结束语

在信息处理显示技术的发展过程中,模拟仿真已经成为工程试验必不可少的一部分。以 C# 这种较为前沿程序设计语言为编程平台,已经得到了较为广泛的应用。而 GDI+是在 Windows 窗体应用程序中以编程方式呈现图形的惟一方法。所以,文中简单介绍了运用这两种方式实现仿真显示的技巧,希望能对以后的工作实践有所启示。

参考文献:

[1] Chand M. GDI+图形程序设计[M]. 北京:电子工业出版社, 2005.

[2] 丁 腾. C 编程从入门到精通[M]. 北京:北京希望电子出版社, 2002.

[3] Robinson S, Allen K S. C# 高级编程[M]. 北京:清华大学出版社, 2002.

[4] Schildt H. C# 完全手册[M]. 北京:电子工业出版社, 2002.

[5] 李兰友, 韩广峰, 裘旭光. C# 图形程序设计实例[M]. 北京:国防工业出版社, 2003.