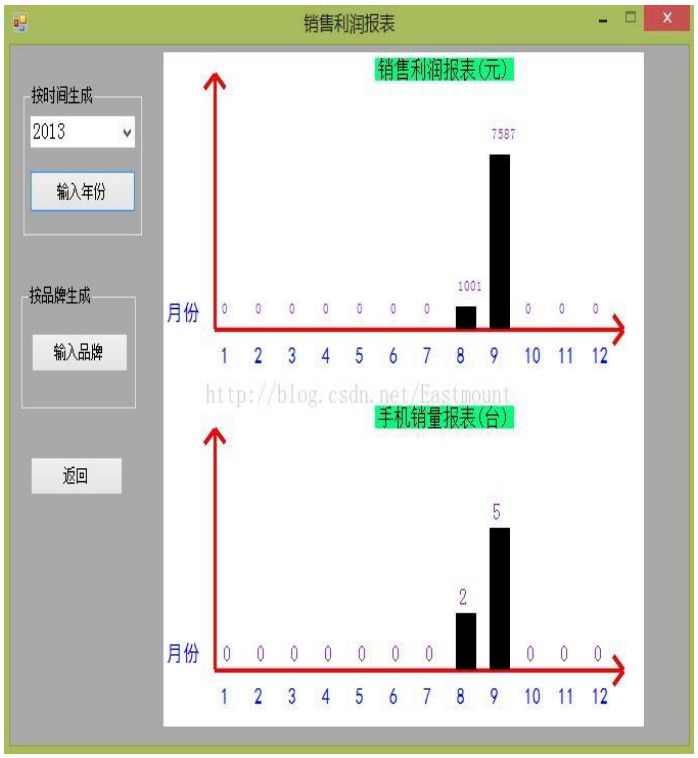
C# 数据库系统中使用GDI+绘制柱状图

在C#+SQL Server数据库做系统中,通常需要对数据库中的数据进行绘制图形报表方便经理查看,虽然有很多实用的水晶报表控件和图表控件实现该功能,但我还是想讲讲如何使用GDI绘制简单的柱状图.(推荐大家不要手画,尽量使用已有控件函数绘制)

一.前言

我们在使用C#+SQL Server做的简单应用系统是一个手机销售的系统,所以它有不同手机品牌和不同年份销售手机的利润和数量的柱状图,方便公司经理查看那个月和那种品牌手机销售更好,方便进货增加企业利润.其中它点击"输入年份",查看"2013"年的销售柱状图数据如下所示:(其实效果还行)



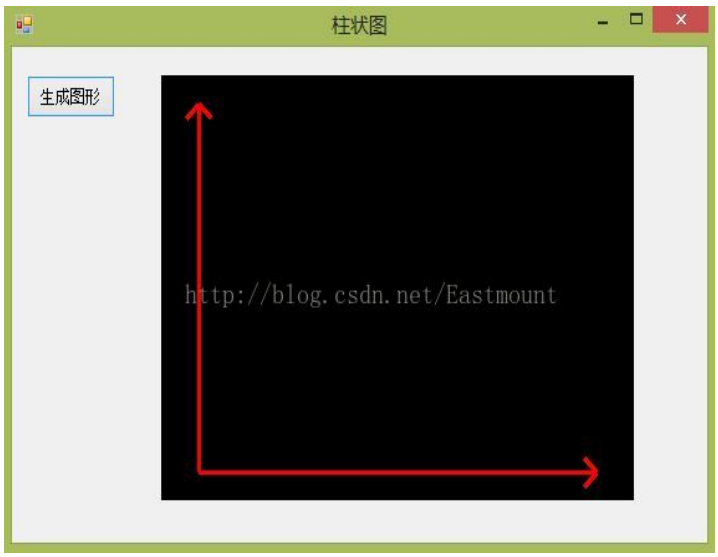
二.绘制坐标

下面就首先讲述如何使用GDI+绘制简单的柱状图,首先上面图中我定义为是静态柱状图,因为它X(横坐标)只有12个月份的数据,不会改变,而后面讲解的我定义的动态柱状图会根据手机品牌动态更新数据的,后面将介绍.
首先第一步需要绘制坐标,创建一个"C# Windows窗体应用程序",添加一个简单的button.然后双击button,进入button1_Click(object sender, EventArgs e)事件.添加如下代码.

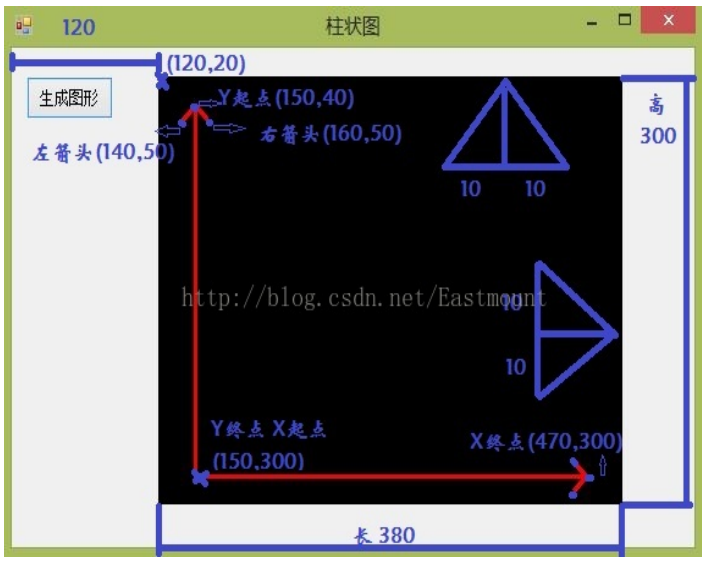
```
//定义变量
Graphics graphic;                                   //GDI+绘图图面
SolidBrush Bfill = new SolidBrush(Color.Black);     //定义单色画刷用于填充图形
Pen Rpen = new Pen(Color.Red, 3);                   //创建红色画笔
//该函数用于绘制坐标
private void CreateTable()
```

```
{
      graphic = this.CreateGraphics();
    Rectangle rect = new Rectangle(120, 20, 380, 300); //绘制黑色背景
    graphic.FillRectangle(Bfill, rect);                //填充这个矩形
    //Y坐标 (150,40)不变
    graphic.DrawLine(Rpen, new Point(150, 40), new Point(150, 300));   //Y坐标
    graphic.DrawLine(Rpen, new Point(150, 40), new Point(140, 50));    //Y坐标 左箭头
    graphic.DrawLine(Rpen, new Point(150, 40), new Point(160, 50));    //Y坐标 右箭头
    //X坐标 (470,300)不变
    graphic.DrawLine(Rpen, new Point(150, 300), new Point(470, 300));  //X坐标
    graphic.DrawLine(Rpen, new Point(460, 290), new Point(470, 300));  //X坐标 上箭头
    graphic.DrawLine(Rpen, new Point(460, 310), new Point(470, 300));  //X坐标 下箭头
}
```

在private void button1_Click(object sender, EventArgs e)事件中增加"CreateTable();"代码即可点按钮时击创建坐标,如图:



它具体的坐标如下图所示,其中Rectangle(120, 20, 380, 300)对应Rectangle(int x, int y, int width, int height)表示一个矩形对用起始位置和长宽;使用graphic.DrawRectangle是绘制一个空心的矩形,而graphic.FillRectangle(画刷,矩形)对矩形进行填充.使用graphic.DrawLine(Pen,Point1,Point2)表示画笔,起始坐标xy,终止坐标xy,是绘制直线,其中在绘制箭头需要注意它的相应坐标.其实在草稿纸上设计好在绘制,感觉还是很方便的.

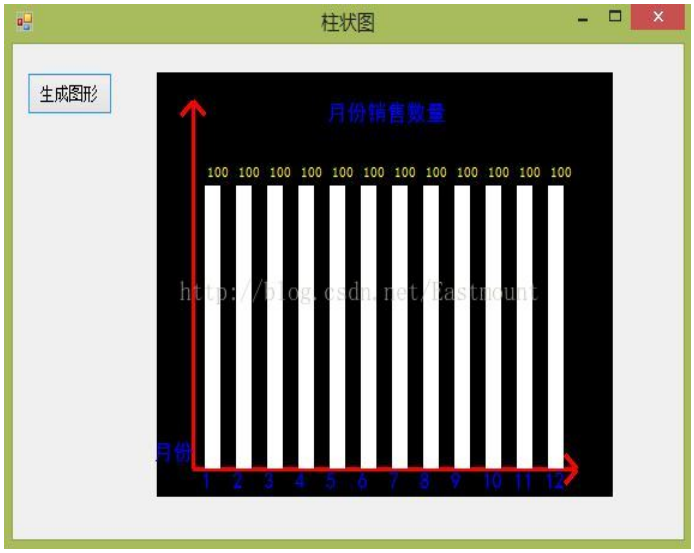


注意:在C#中load事件中是不能画图的,因为Windows的GDI必须在拥有屏幕资源的时候才能有效,FormLoad时窗体还没

有绘图资源,所以不能画出.在窗体创建完成后如果再调用FormLoad时间就可以绘制图形.所以我是在点击按钮时创建坐标.

三.静态的柱状图

下面是绘制静态的柱状图,它的运行效果如下图所示,绘制了12个等高的柱状图:

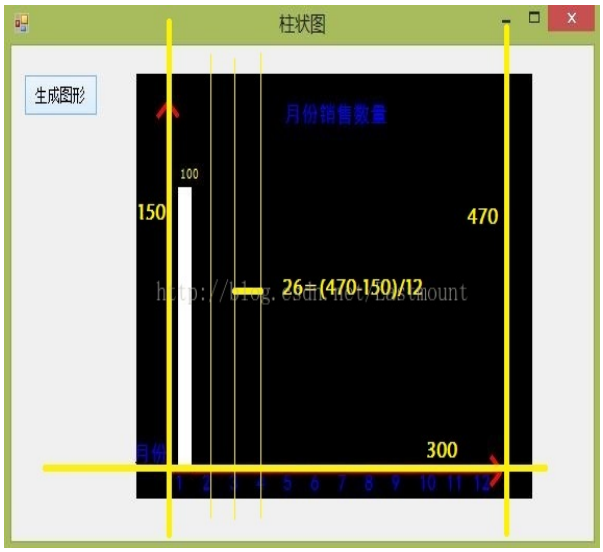


它的源代码如下,通过调用DrawTable()绘制静态的图形.为什么我定义它为静态的后面将介绍:

```
//定义变量
Brush Bbrush = Brushes.Blue;                          //创建蓝色画刷 文字
Font myFont = new Font("黑体", 12);                   //创建字体
Font tFont = new Font("宋体", 8);                     //创建字体 显示在直方图上的数量
SolidBrush Wfill = new SolidBrush(Color.White);       //定义单色画刷用于填充图形
//该函数用于绘制静态报表
private void DrawTable()
{
    //输出文字
    graphic.DrawString("月份", myFont, Bbrush, new RectangleF(115, 280, 40, 40));
    graphic.DrawString("月份销售数量", myFont, Bbrush, new RectangleF(260, 40, 200, 40));
    //绘制月份 12个
    for (int i = 1; i <= 12; i++)
    {
        graphic.DrawString(i.ToString(), myFont, Bbrush, 155 + (i-1) * 26, 300);
    }
    //定义绘制柱状图坐标|宽|高
    int x, y, width, heigh;
    x = 160;                    //X坐标定值=160 Y轴的x坐标为150
    width = 13;                 //width坐标=(470-150)/(12*2)=13
    heigh = 200;                //X轴y坐标=300 heigh=200
    y = 100;                    //Y坐标 y=300-200

    for (int i = 0; i < 12; i++)
    {
        //填充图形
        Rectangle rect = new Rectangle(x+i*26, y, width, heigh);
        graphic.FillRectangle(Wfill, rect);
        //显示数量
        graphic.DrawString("100", tFont, Brushes.Yellow, x + i * 26, y - 15);
    }
}
```

这里的有几个地方需要注意,其中26=(470-150)/12表示把X轴分成12个等分,其中每个等分宽为26,再在每个等分绘制一半13的白色矩形.如下图:

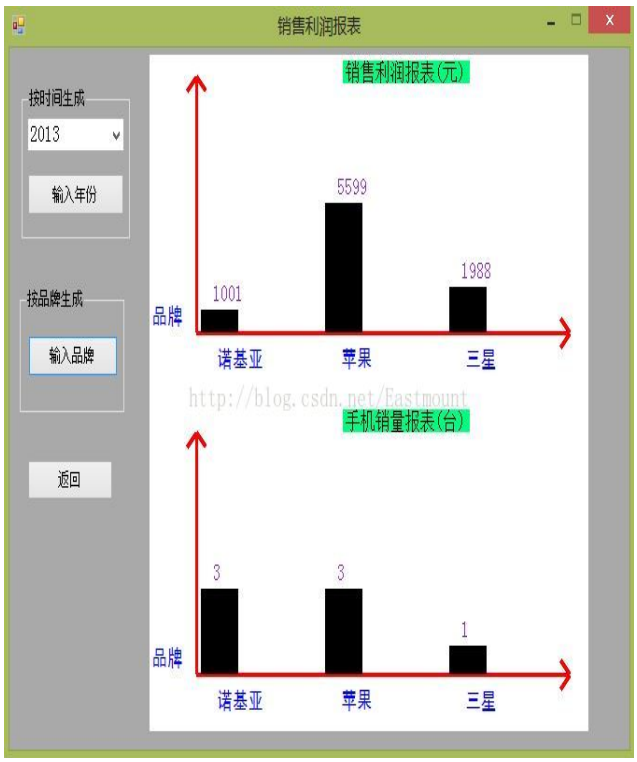


四.动态的柱状图

如果动态显示的柱状图,就是获取数据库中具体的数量,在进行绘制图形的基本过程为定义:int num[12]分别记录12个月份中的销售数量,在sumNum+=num[i];计算12个月总的销售数量,通过百分比计算具体的每个月的高度,这样的好处在于当其中一个月的销售数量很大时,会出现那个矩形很高,超出界面范围.

num[i]/sumNum=具体高度/总高度(其中总高度我们在上面设置为heigh=200)

同样,如果在做品牌销售业绩时,如下图所示:



这是销售3中品牌手机"诺基亚"、"三星"、"苹果"的情况,如果想增加一个新的品牌"HTC"时,这是就会显示4条柱状图,因此也需要动态的生成.我们已经求出了它的总宽度为(470-150)=320,再通过320/具体的品牌数量,即可平均分配每个品牌的数量,再求出具体的坐标即可.

分享文章:http://www.cnblogs.com/stg609/archive/2008/03/30/1129221.html

最后补充下获取数据库品牌的代码：

```
//获取手机品牌信息 (静态生成5个品牌)
private void SelectName()
{
    string sql = "select * from SellList";
    string consqlserver = "Data Source=.;Initial Catalog=TelephoneMS;Integrated Security=True;";
    SqlConnection con = new SqlConnection(consqlserver);     //定义SQL Server连接对象
    SqlDataAdapter da = new SqlDataAdapter(sql, con);        //数据库命令和数据库连接
    con.Open();
    DataSet ds = new DataSet();                              //声明一个DataSet对象
    da.Fill(ds);                                             //装入数据
    //清零
    for (int i = 0; i < 5; i++) {
        price[i] = 0;
        num[i] = 0;
    }
    try
    {
        //获取实际销售价格总和
        for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
        {
            if (ds.Tables[0].Rows[i]["phonename"].ToString() == "诺基亚")
            {
                //利润 = 实际销售价格 - 进货价格
                price[0] += float.Parse(ds.Tables[0].Rows[i]["sellmoney"].ToString()) -
                    float.Parse(ds.Tables[0].Rows[i]["price"].ToString());
                num[0]++;
            }
            else if (ds.Tables[0].Rows[i]["phonename"].ToString() == "iphone")
            {
                price[1] += float.Parse(ds.Tables[0].Rows[i]["sellmoney"].ToString()) -
                    float.Parse(ds.Tables[0].Rows[i]["price"].ToString());
                num[1]++;
            }
            else if (ds.Tables[0].Rows[i]["phonename"].ToString() == "三星")
            {
                price[2] += float.Parse(ds.Tables[0].Rows[i]["sellmoney"].ToString()) -
                    float.Parse(ds.Tables[0].Rows[i]["price"].ToString());
                num[2]++;
            }
            else if (ds.Tables[0].Rows[i]["phonename"].ToString() == "HTC")
            {
                price[3] += float.Parse(ds.Tables[0].Rows[i]["sellmoney"].ToString()) -
                    float.Parse(ds.Tables[0].Rows[i]["price"].ToString());
                num[3]++;
            }
            else if (ds.Tables[0].Rows[i]["phonename"].ToString() == "OPPO")
            {
                price[4] += float.Parse(ds.Tables[0].Rows[i]["sellmoney"].ToString()) -
                    float.Parse(ds.Tables[0].Rows[i]["price"].ToString());
                num[4]++;
            }
        }
    }
    catch (Exception msg)
    {
        MessageBox.Show(msg.Message);
    }
    finally
    {
```

```
            con.Close();
                                con.Dispose();
            da.Dispose();
        }
    }
```

总结:回想起来使用GDI+绘制柱状图,确实很蛋疼,也就没继续贴上代码了有具体的控件却不用,但我还是希望该文章对那些不知道在做使用C#做系统中如何生成柱状图报表的有帮助(尽量使用已有的控件).
如果知道如何使用Dundas Chart for .NET\ReportViewer\FormCrystal等制作报表的就当该文章简单帮助你回顾一些GDI+的知识吧!如果文章中有错误或不足的地方,见谅!
(BY:Eastmount 2013-9-14 夜1点http://blog.csdn.net/eastmount/)