Seismic.NET 教程

V1.1

2014 年 2 月 27 日

简介

本教程通过一些例子一步一步地让你学会如何用 Seismic.NET 开发一个地震剖面程序，所有例子在 GeoToolkit.NET 2.3[1]和 Visual Studio 2010[2]开发环境下通过。

源程序文件包分为 step01 到 step15 分别对应于 15 章的例子。

Resources 是程序中用到的一些图标资源。

CGM_Office 是一个用于查看 CGM 文件的共享软件。

jz.segy 和 chunbo1.segy 是 2 个用于测试的 SEGY 文件。

如有问题请联系：申龙斌，SLOFSLB@QQ.COM

修订记录：

2014 年 1 月 19 日，完成初稿 V1.0

2014 年 2 月 27 日，V2.0，在第 2 章对 cgPlot 加了少量说明。

[1] GeoToolkit.NET 是 INT 公司的软件产品

[2] Visual Studio 是微软公司的软件开发工具

1 例子一：迷你剖面显示程序

1.1 程序功能

用最少的语句实现一个地震剖面显示程序，里面没有任何事件的处理和交互。

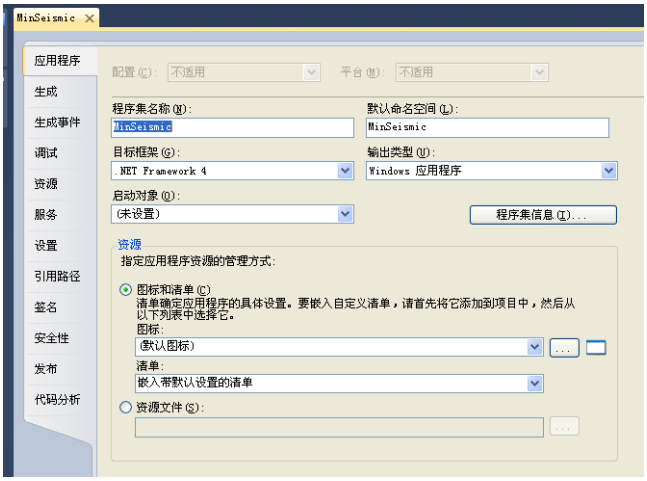
1.2 步骤

1）安装 GeoToolkit.NET 2.3 版本

默认安装文件的路径中 C:\Program Files\int\Net\2.3.2987.0 下

2）在 VS2010 中新建一个项目

项目类型为“Windows 窗体应用程序”，名称可以是 step1。项目属性的目标框架用.NET Framework 2.0，3.0，3.5 或 4.0（推荐 4.0），不能用 client profile。



3）添加 GeoToolkit.NET 的引用文件

当前例子需要 4 个 DLL，这些文件可以在 C:\Program Files\int\Net\2.3.2987.0\Dlls 下找到：

Carnac.NET.dll：基础绘图工具包

Carnac.Windows.NET.dll：在 Windows 下的绘图包

Seismic.NET.dll：地震剖面组件包

Seismic.Windows.NET.dll 地震剖面在 Windows 环境的组件包。

4）用到一些名字空间

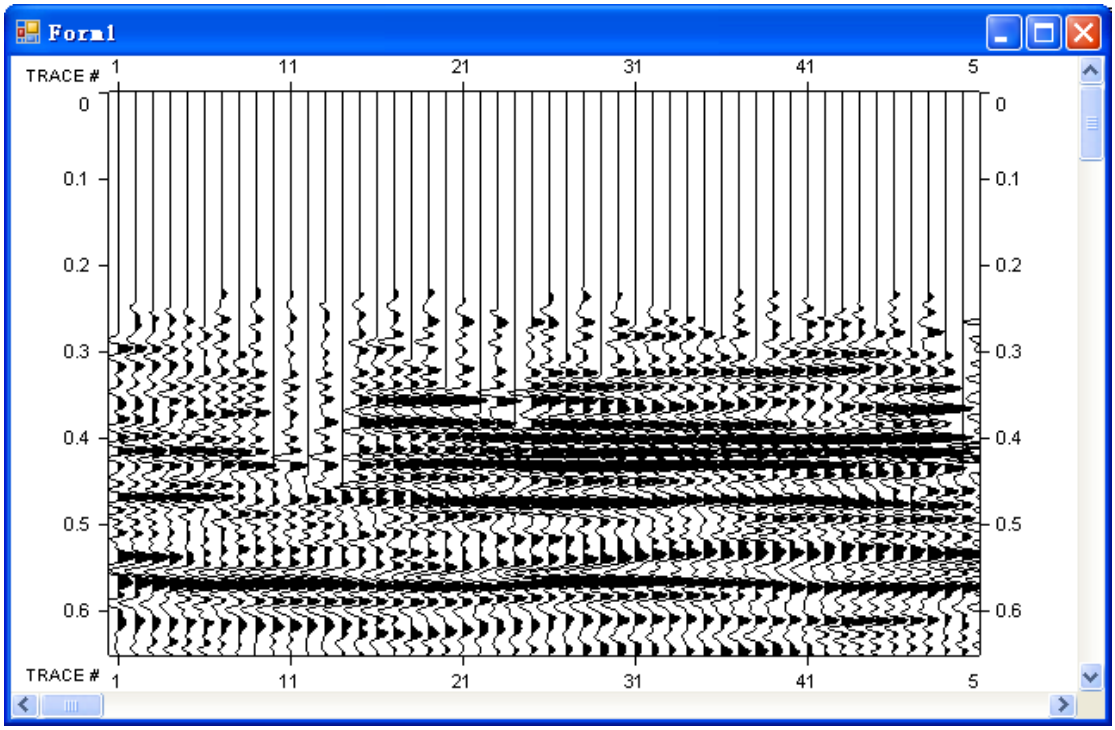
```
using Interactive.Carnac2d.Plot.Common.Shaped.Gui;
using Interactive.Geo.Seismic.View;
using Interactive.Geo.Seismic.Pipeline;
```

```
using Interactive.Geo.Seismic.Segy;
using Interactive.Geo.Seismic.Axes;
```

5）在 InitializeComponent()语句后面添加如下代码：

```
// 用一行语句把 reader, pipeline, view 和 plot 都创建出来
cgSeismicPlot plot = new cgSeismicPlot(
          new cgSeismicView(
              new cgSeismicPipeline(
                  new cgSegyReader("..\\..\\..\\jz.segy")
              )
          ),
          cgTraceAxisPosition.Both,   // 上下都显示道号轴
          cgSampleAxisPosition.Both  // 左右都显示时间轴
);

// 创建一个 PlotPanel 用来容纳 plot，这个 plot 本身是无窗口的
cgScrollablePlotPanel plotControl = new cgScotPanel(plot);
plotControl.Dock = DockStyle.Fill;
plotControl.PrimaryView.Background.SetFill(Color.White);

// 把 Seismic.NET 的控件放在 Windows 的控件中
this.Controls.Add(plotControl);
```

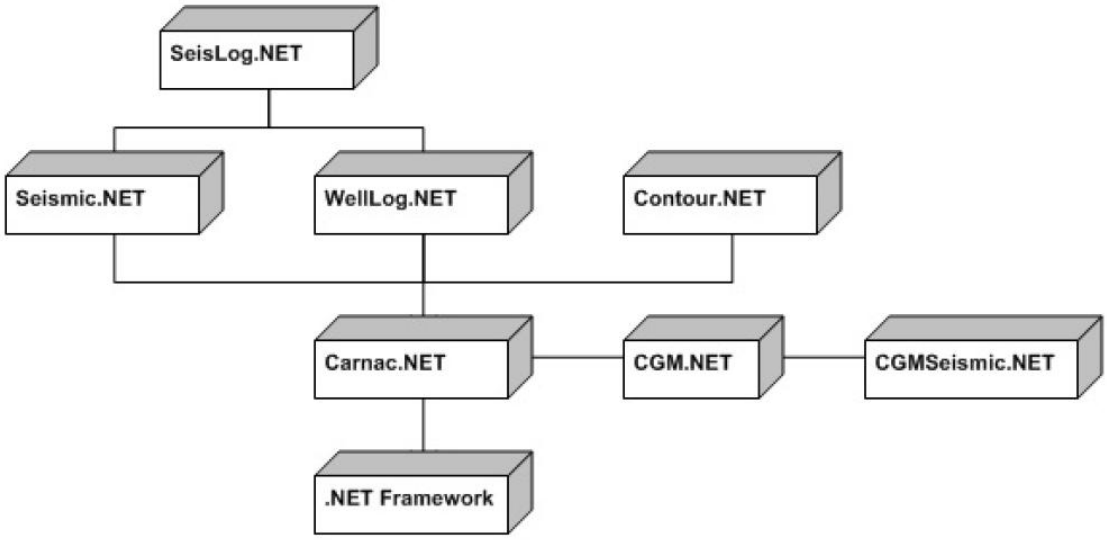
6）编译运行



1.3 重点讲解

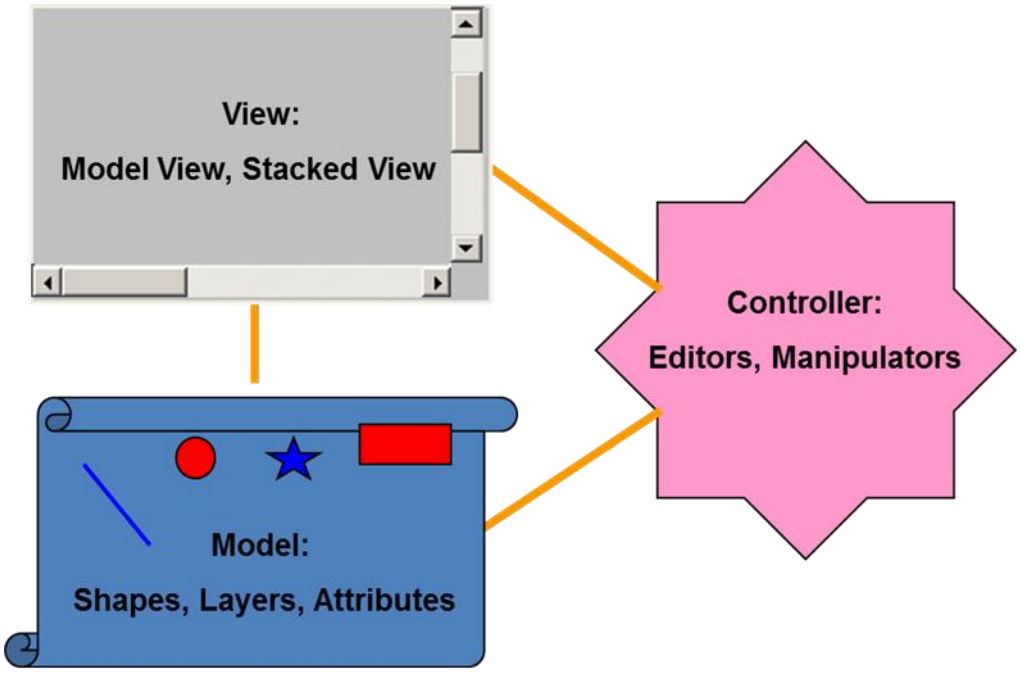
1.3.1 Seismic.NET 与 Carnac.NET 的关系

Carnac.NET 是基础的绘图包，Seismic.NET 依赖 Carnac.NET，如下图所示。



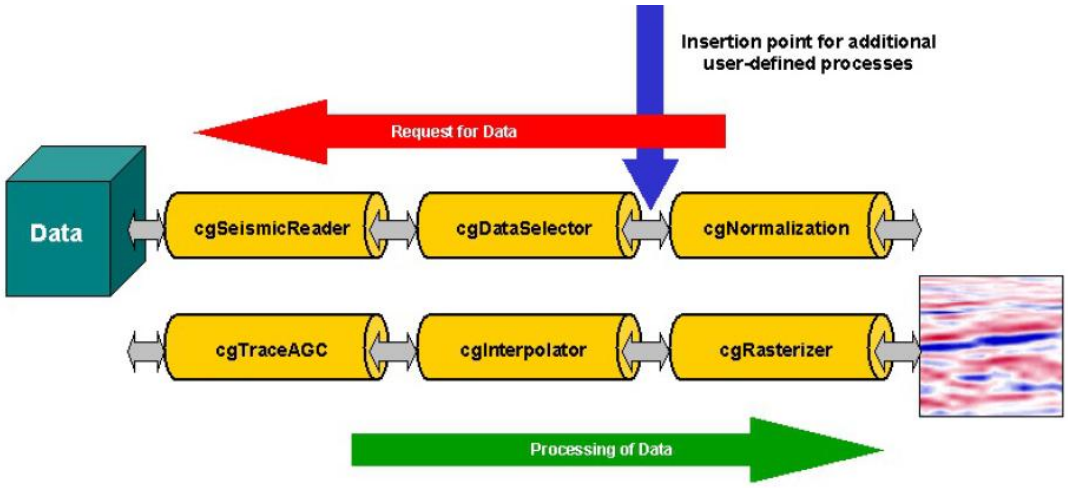
1.3.2 reader, pipeline, view 和 plot

在 Seismic.NET 程序中经常用到 4 个类，它们是 cgSegyReader、cgSeismicPipeline、cgSeismicView 和 cgSeismicPlot，它们是 MVC 编程模型的体现，Carnac 框架是按 MVC 思想设计的。



cgSegyReader 负责从 SEG-Y 文件中读取数据。

cgSeismicPipeline 是一个比较特殊的类，由于地震数据处理作业会有许多的处理过程，这些地震道可以像管道一样连起来，当一个处理流程完成时，数据会流入下一个处理流程，这样像一条流水线一样，处理效率会非常高。从数据到剖面图像的过程中实际上也经过了许多步骤。



cgSeismicView 就是指的 MVC 中的 V(view)。

cgSeismicPlot 是一个没有窗口的绘图区，也就是说这段代码也可以在服务器端运行，可以将这个 plot 输出为一个位图，也就是说可以在 WEB 服务器中生成剖面图像传送给客户端浏览器。

1.3.3 cgScrollablePlotPanel

由于 plot 是无窗口的，所以要将它装在一个窗口控件中才能显示出来，在 Seismic.NET 中 cgScrollablePlotPanel 是对 Windows.NET 中 Panel 的封装，这个 Panel 实际上就是一个 UserControl。

```
cgScrollablePlotPanel plotControl = new cgScrollablePlotPanel(plot);
```

1.3.4 最后记得要把这个控件加入到主窗体中

```
this.Controls.Add(plotControl);
```

当然也可以将这个 plotControl 加入到一个 Windows 的任何一个 Panel 中。

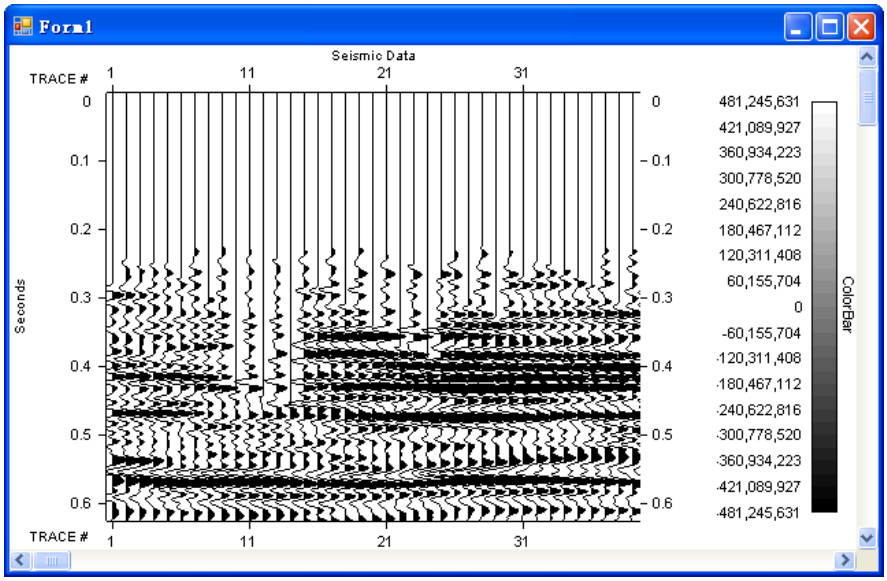
1.4 实验

试着修改 new cgSeismicPlot()中的各种参数设置，可以得到不同的显示效果。

```
public cgSeismicPlot (
    cgAbstractSeismicView view,
    cgTraceAxisPosition traceAxisPosition,  // 在哪些位置显示道号
    cgSampleAxisPosition sampleAxisPosition, // 在哪些位置显示时间或深度(如果是深度剖面)
    cgLabelVisibility labelVisibility,  // 坐标轴的名称，例如纵轴单位是 seconds
    cgAnnotationPosition titlePosition, // 这个 plot 还可以设置 title 的位置
    cgAnnotationPosition colorBarLocation // 颜色棒的位置
)
```

例如：

```
cgSeismicPlot plot = new cgSeismicPlot(
    new cgSeismicView(new cgSeismicPipeline(new cgSegyReader("..\\..\\..\\jz.segy"))),
    cgTraceAxisPosition.Both,   // 上下都显示道号轴
    cgSampleAxisPosition.Both,  // 左右都显示时间轴
    cgLabelVisibility.Left,
    cgAnnotationPosition.Top,
    cgAnnotationPosition.Right
);
```



2 例子二：响应右键点击事件

2.1 程序目标

可以弹出右键菜单，设置剖面的显示参数。

2.2 步骤

1）私有变量 plot

由于程序中经常要引用 plot，所以把它放在一个私有变量中。

```
    private cgSeismicPlot plot;
```

2）一个重要的 using

```
using Common = Interactive.Carnac2d.Plot.Common;
```

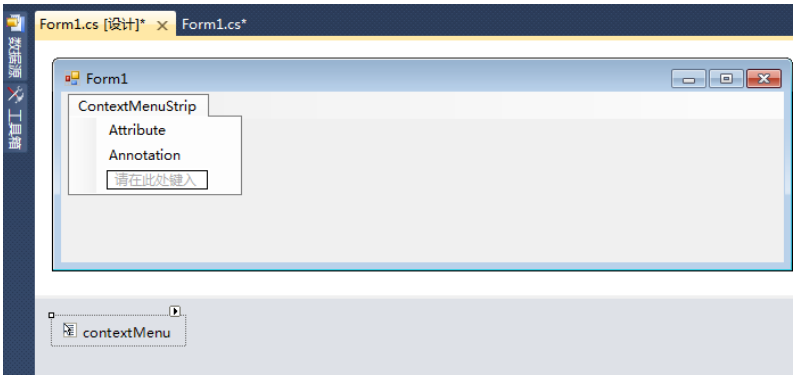
之所以这样，是因为在名字空间 Interactive.Carnac2d.Plot 中，还有一个 cgPlot，不过这个 cgPlot 是一个接口，如果你同时引用了 2 个名字空间，则会使 cgPlot 产生冲突。

3)在 Controls.Add(plotControl)代码行之后加入代码

```
Common.cgPlot centerPlot = plot.GetAnnotation( cgGenericPlotLayout.CENTER ) as
Common.cgPlot;
Common.cgWinFormTool tool = new Common.cgWinFormTool();
centerPlot.Tool = tool;
tool.MouseDown += new MouseEventHandler(OnCenterPlot_MouseDown);
```

4）加上右键菜单

在主窗口的设计视图中加上一个右键菜单，命名为 contextMenu，里面加上 2 个菜单项，Attribute 和 Annotation。

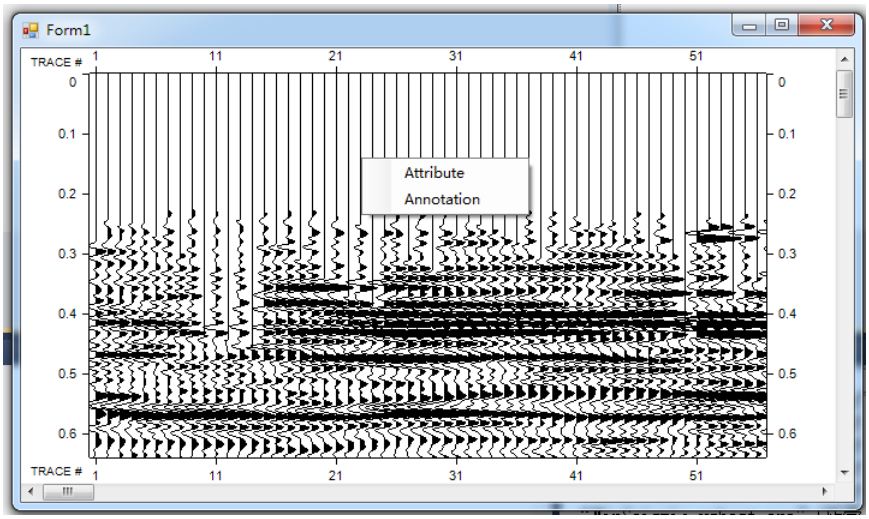


5）加上事件响应 OnCenterPlot_MouseDown()方法

```
private void OnCenterPlot_MouseDown(object sender, MouseEventArgs  e)
{
    if (e.Button == MouseButtons.Right)
    {
        Common.cgPlot centerPlot = plot.GetAnnotation(cgGenericPlotLayout.CENTER) as
Common.cgPlot;
        contextMenu.Show(this, centerPlot.PointToDevice(new cgPoint(e.X, e.Y)));
    }
}
```

6）运行程序

此时，可以响应鼠标右键，弹出菜单。



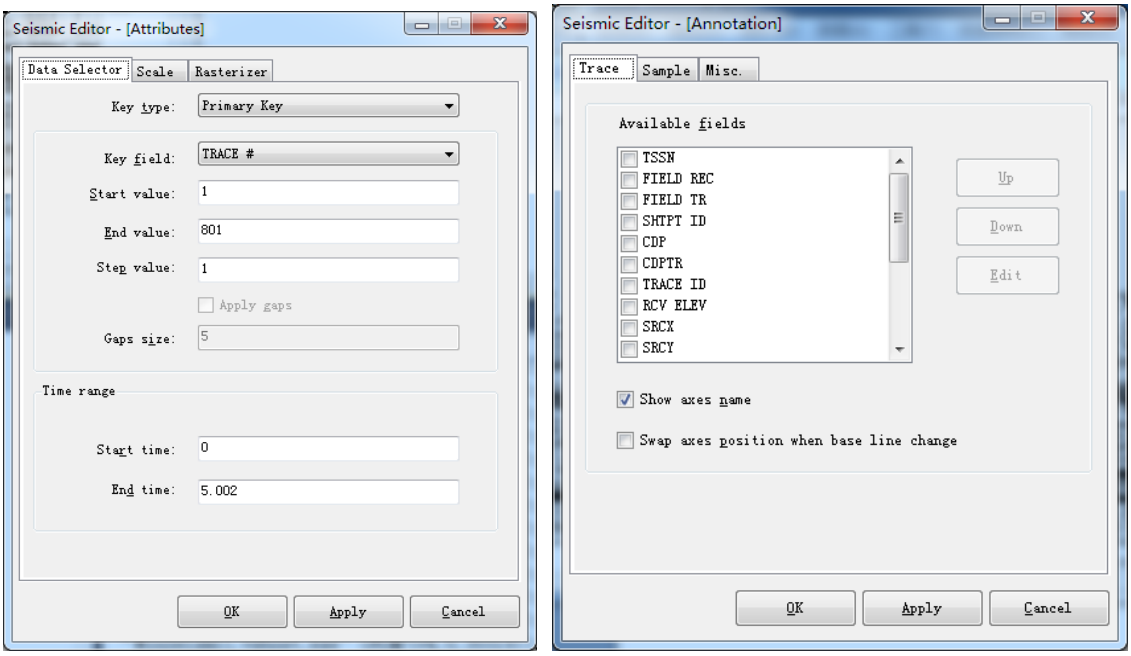
7）响应菜单项单击事件

```
cgAttributesEditorBuilder builder = new cgAttributesEditorBuilder(plot);
Form form = builder.CreateForm(plot.SeismicView.Pipeline);
form.ShowDialog();
```

再运行后，在点击右键菜单中的 Attributes 菜单项，出现 Attributes 编辑器，调整各项参数，看看剖面显示的变化情况。

同理可以构建出 Annotation 编辑器。

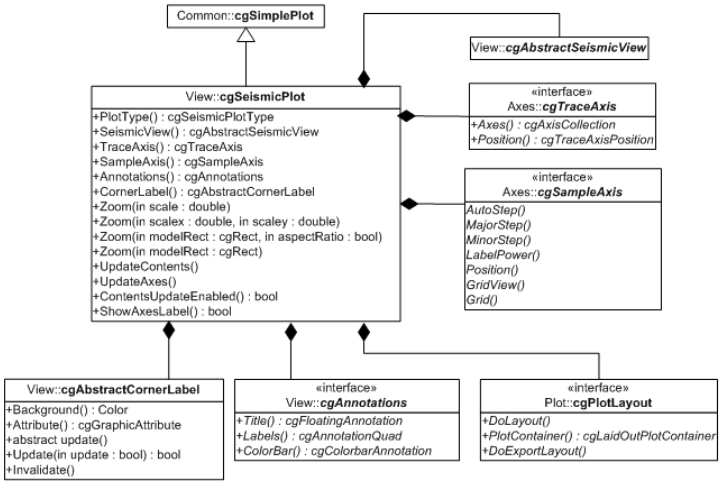
```
cgAnnotationEditorBuilder builder = new cgAnnotationEditorBuilder(plot);
Form form = builder.CreateForm(null);
form.ShowDialog();
```



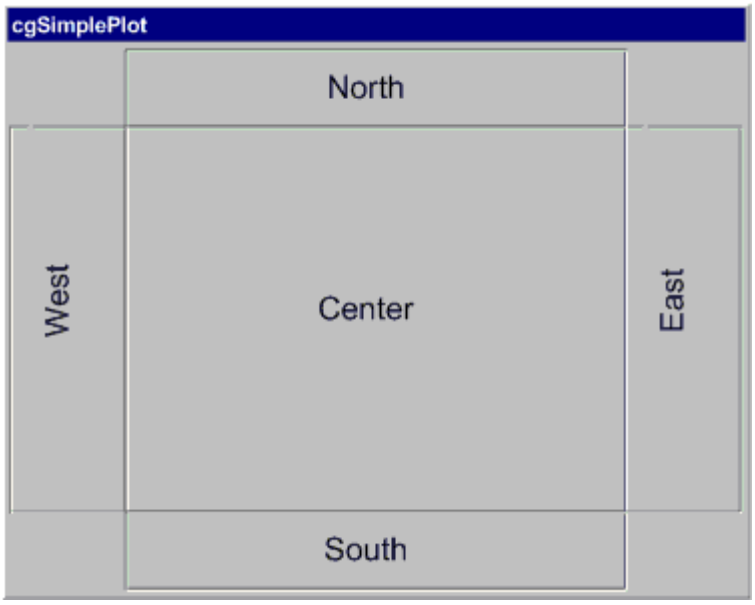
2.3 重点讲解

2.3.1 plot 的布局

Seismic Plot 已经封装了剖面显示中常用的可视化组件，它包含了 Seismic View、坐标轴以及一些标注。



特别注意的是它继承自 Common::cgSimplePlot 类，cgSimplePlot 包含东、西、南、北、中 5 个子 plot。



而 cgSeismic Plot 功能更强大，有 9 个子 plot，四周通常用于显示坐标轴、颜色棒或一些标注，中间那个 plot 用于显示地震剖面，大多数交互动作都发生在这里，所以经常用到，可以用下面这条语句取得 center plot。

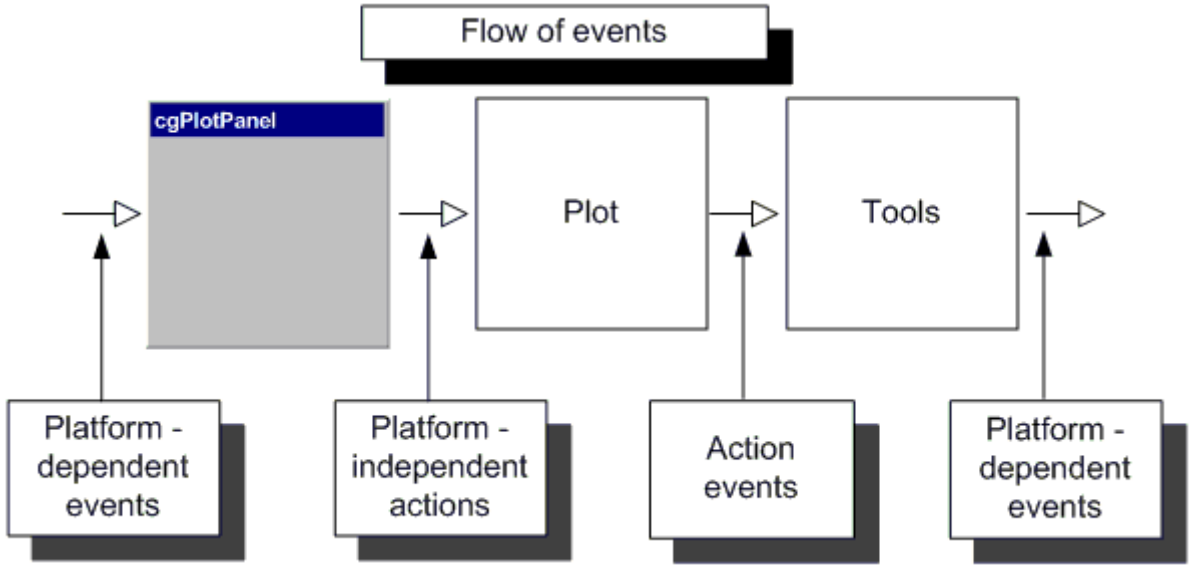
```
Common.cgPlot centerPlot = plot.GetAnnotation( cgGenericPlotLayout.CENTER ) as
Common.cgPlot;
```

NorthWest	North	NorthEast
West	Center	East
SouthWest	South	SouthEast

2.3.2 cgWinFormTool

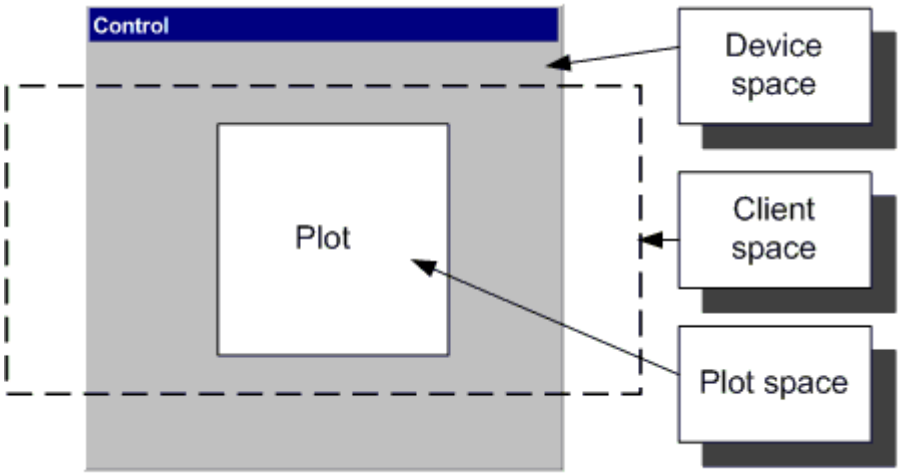
Carnac.NET 在设计的时候要支持多个.NET 框架（Microsoft 和 Mono）以及多个 GUI 库(WinForms, Gtk#)，为了不直接依赖于 System.Windows.Forms，所以又加了一层 Tools，用于

支持鼠标和键盘事件的响应。



2.3.3 几种坐标

Carnac 中有 3 种坐标：Device space, Client space 和 Plot Space。



Device space 的坐标对应于 Windows 上控件的坐标，左上角是(0,0)，坐标单位是像素。

```
cgPoint pt  = plot.PointToDevice( new cgPoint( e.X, e,Y ) );
```

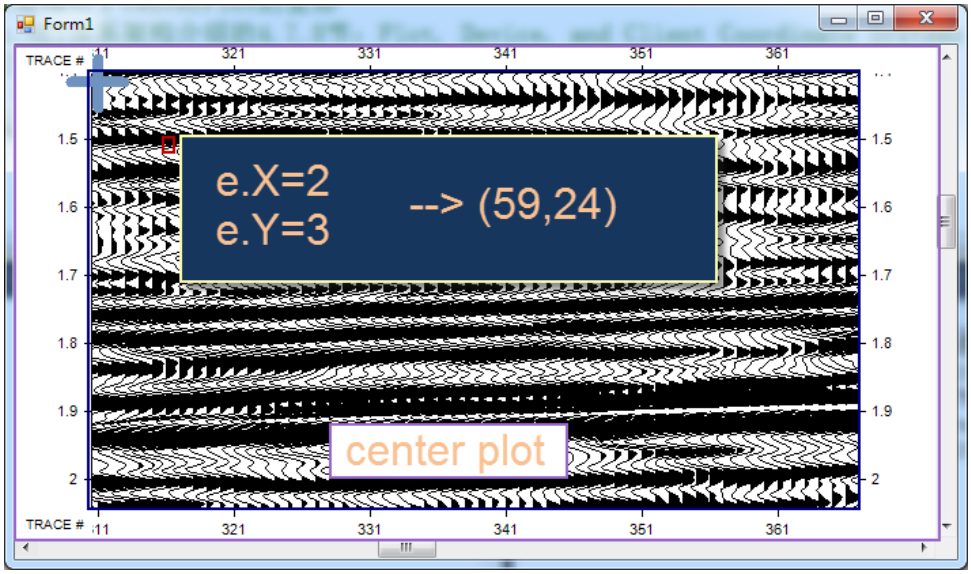
Client space 可以称为模型坐标。在剖面程序里的模型横坐标是道号（从 0 开始），纵坐标是时间值（单位是秒），这样在 311 道，1.4 秒处的 client 坐标，就是(2,3)。

```
cgPoint plotPoint = new cgPoint(2, 3);
cgPoint pt  = _plot.PointToClient( plotPoint );
```

Plot space 是一块可视区域的坐标。

```
cgPoint clientPoint = new cgPoint(310, 1.4);
Point pt2 = (Point)centerPlot.PointToScreen(clientPoint);
```

以剖面程序为例，事件上得到的坐标(e.X, e.Y)是在 center plot 上得到的，下图上十字光标位置上的坐标为(2,3)，调用 centerPlot.PointToDevice 方法后，得到(59, 24)，这是因为左边和上边的刻度和坐标轴占据了一部分空间，Windows 中显示右键菜单的位置需要这个坐标。



2.3.4 Seismic Editor

Seismic.NET 中提供了 2 个常用的属性编辑器(cgAttributesEditor 和 cgAnnotationEditor)，可以调整剖面的显示参数。里面的选项卡也可以定制，详细说明参见 Seismic Architecture 4.8 节。

2.4 试验

在 Attributes Editor 和 Annotation Editor 里调整各项参数，观察剖面显示发生的变化。

3 例子三：缩放操作

3.1 程序目标

可以放大 zoom in，缩小 zoom out，也可以拉框放大。

3.2 实现步骤

1）在主窗口左侧添加一个工具栏，里面加上 3 个按钮，分别对应于 zoomin，zoomout 和 zoom。

2）加上一个操纵器

```
private cgDragBoxManipulator zoomMan;
```

3）加上 2 个事件响应

```
tool.MouseMove += new System.Windows.Forms.MouseEventHandler(OnCenterPlot_MouseMove);
tool.MouseUp += new System.Windows.Forms.MouseEventHandler(OnCenterPlot_MouseUp);
```

4）初始化操纵器 manipulator

```
private static cgDragBoxManipulator CreateZoomManipulatorInPlot( cgSeismicPlot plot)
{
    cgAttributePalette palette = new cgManipulatorAttributePalette();
    cgAttribute palAttr = palette.GetAttribute("RubberBand");
    palAttr.SetLineColor(Color.Red);
    palAttr.SetLineWidth(3);
    palAttr.SetFill(Color.FromArgb(50, Color.Red));  // 拉框矩形的透明度为 50
    cgDragBoxManipulator zoomMan = new cgDragRectangleManipulator(palette);

    // 分别建立一个 model 和 view model，叠在整个 SeismicView 之上
    cgSimpleModel rubberModel = new cgSimpleModel();
    cgSimpleModelView rubberView = new cgSimpleModelView(rubberModel);
    plot.SeismicView.Views.Add("RubberView", rubberView);

    // 要激活 manipulator
    zoomMan.Activate(
                new cgSimpleViewAdapter(rubberView, rubberModel),
                new cgSimpleViewAdapter(plot.SeismicView));

    return zoomMan;
}
```

5）响应 zoom in 和 zoom out

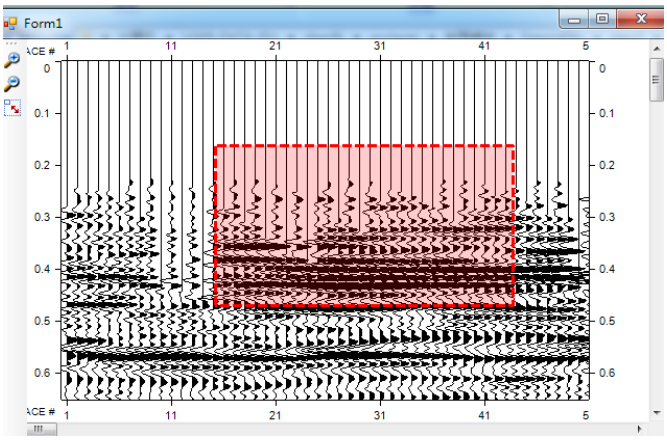
```
plot.Zoom(2.0);  // 放大
plot.Zoom(0.5);  // 缩小
```

6）分别修改鼠标按下、移动和松开的事件响应

```
// 修改 MouseDown 事件响应
if (zoomMan.IsActive())
    zoomMan.Begin(centerPlot.PointToDevice(new cgPoint(e.X, e.Y)));

// 修改 MouseMove 事件响应
if (e.Button == MouseButtons.Left && zoomMan.IsActive() && zoomMan.IsStarted())
{
    zoomMan.Move(centerPlot.PointToDevice(new cgPoint(e.X, e.Y)));
}

// 修改 MouseUp 事件响应
if (e.Button == MouseButtons.Left && zoomMan.IsActive())
{
cgRect modelRect = zoomMan.GetRect();
    if(modelRect.width > 1 && modelRect.height > 0.01)
        plot.Zoom(modelRect);
    zoomMan.End();
}
```

7）运行程序

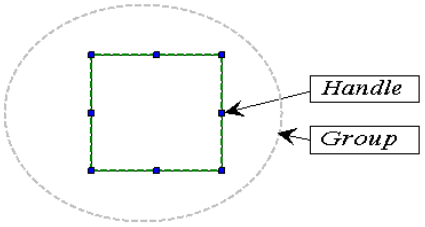
试试放大、缩小以及拉框放大操作。

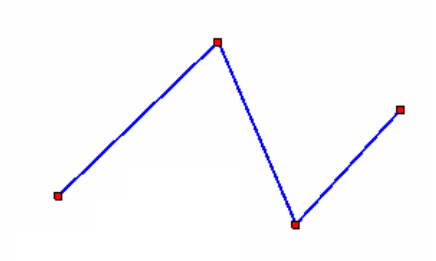


3.3 重点说明

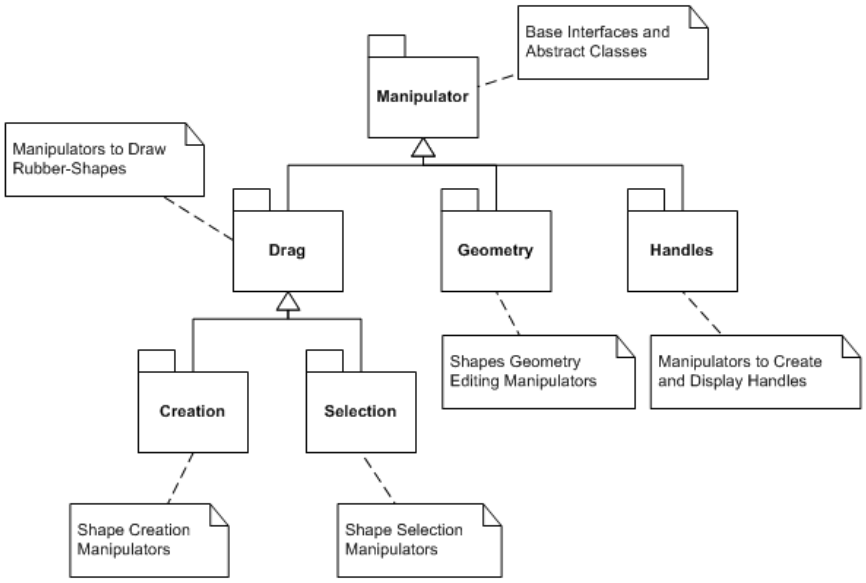
3.3.1 操纵器 manipulator

当拉框放大操作时，程序会显示出一个橡皮条式的矩形框，指示将要放大的区域，当操作结束时，这个橡皮条就会消失，这种辅助性的交互手段，在 Carnac 中称为 manipulator。同样，在选择和编辑一个图形对象时，这种操纵器更常见。





下图是 Carnac 中常用的操纵器，放在不同的类包中。



3.3.2 操纵器的初始化

cgManipulatorAttributePalette 这里有预置好的几种调色板，RubberBand 就是其中之一，可以得到橡皮条的效果，你只需简单修改一下颜色、线宽等，就可以用于你的程序了。

cgDragRectangleManipulator 可以拖出一个矩形框。这个矩形框也要有 Model 和 View，才能显示出来。还要叠在地震剖面的 View 之上。

最后要调用 Activate 激活这个操纵器。

3.3.3 操纵器工作的顺序

鼠标按下后，调用 Begin 画出矩形框的起始点。

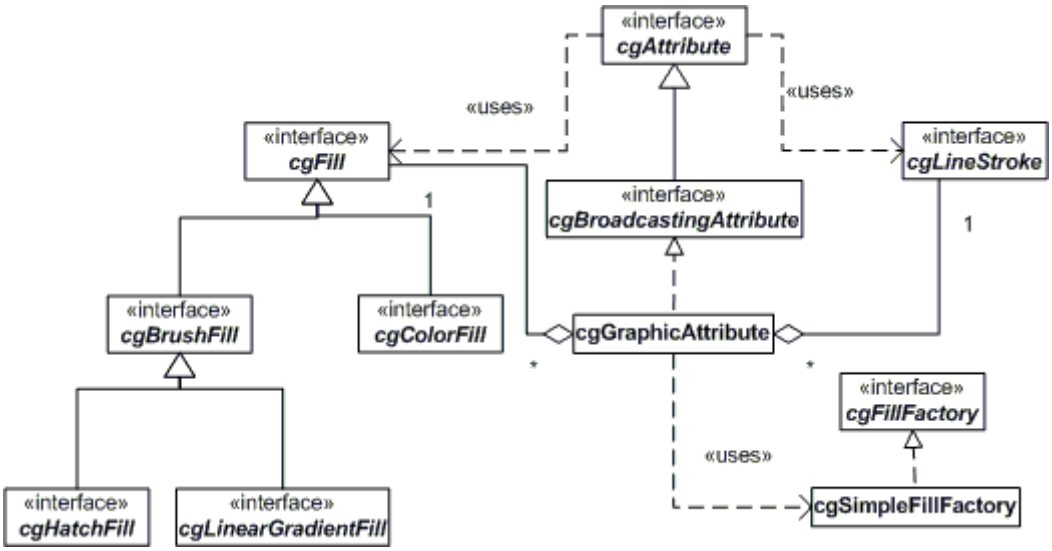
鼠标按住不松并移动鼠标时，画出一个橡皮条式的矩形框，调用 Move 方法。

鼠标松开后，调用 End，矩形框消失，用 GetRect()可以得到矩形框的大小，执行放大操作。

3.4 补充 Carnac 基础知识

3.4.1 cgAttribute

在 Carnac 中图形的几何形状（比如线的坐标）与显示属性（比如线宽、线型、颜色等）是分离的，这些属性分为三类：线型、填充属性和字体属性，类图如下。



可以看出 cgAttribute 是一个接口，用 cgGraphicAttribute 可以实例化一个对象，然后设置线型、颜色等。

```
cgAttribute attr  = new cgGraphicAttribute();
attr.SetLineColor( Color.Red );
attr.SetLineStyle( cgLineStyle.SOLID );
attr.SetFill( Color.Green );
```

如果不想填充一个区域，有三种办法：

```
attr.SetFill( (cgFill)null );
```

或者

```
attr.SetFill( Color.Empty );
```

或者

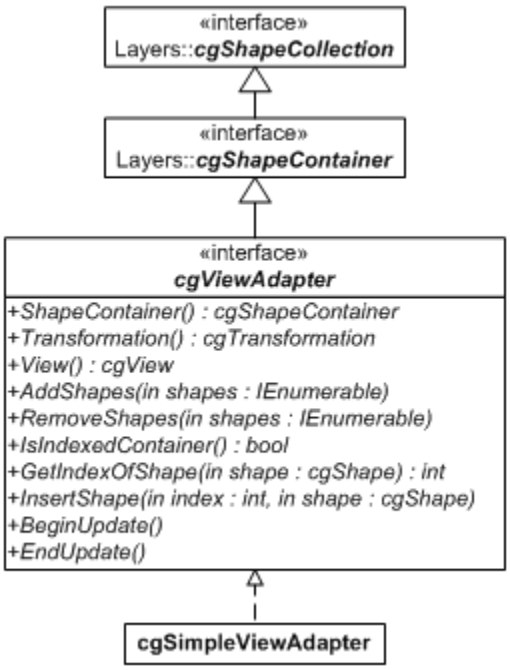
```
attr.SetFill( Color.Transparent );
```

设置了显示属性后，可以用 SetAttribute 把这种显示属性赋给图形对象。

```
cgAttribute attr  = new cgGraphicAttribute();
attr.SetLineColor(cgColor.Green );
cgRectangle rectangle = new cgRectangle( new cgRect( 0,0, 100, 100 ) );
rectangle.SetAttribute( attr );
```

3.4.2 View Adapter

这个概念一开始较难理解，主要用于 manipulator 等对象的处理。manipulator 通过 adapter 来操作数据 model 和图形 view，为了解耦 manipulator 和 view，也就是 manipulator 不必知道 view 的细节，引入了 adapter 的设计。



一个 view adapter 主要包含 2 种属性：

View：操纵器 manipulator 要修改这个 view

ShapeContainer：图形元素都在这里保存着，编辑时会改变它。需要注意的是，这个属性可以为 null（例如对于 panning 拖动这个操纵器来说，不需要修改任何的 shape）。

这个类可以在编辑图形对象时详细了解，可以参见 Carnac Architecture 4.3.2 小节。

3.5 试验

Zoom 方法如果只给一个参数，则放大时将保持纵横比。下面的语句可以不保持纵横比。

```
plot.Zoom(modelRect, false);  // 放大时不保持纵横比
```

4 例子四：剖面的坐标变换

4.1 程序功能

初始剖面显示时用标准剖面的显示比例，每厘米 8 道，每 100 毫秒 1 厘米。

4.2 主要步骤

1）设置 view 的坐标变换，在 plot 初始化之后，加上一条语句即可。

```
plot.SeismicView.Transformation = new cgSeismicTransformation(
                  plot.SeismicView.Pipeline.SeismicReader.MetaData,
                  8, cgTraceUnits.TracesPerCm, //每厘米 8 道
                  10, cgSampleUnits.CmPerSecond,  //每秒 10 厘米，即每 100 毫秒 1 厘米
                  new cgGDIPlusDriver());
```

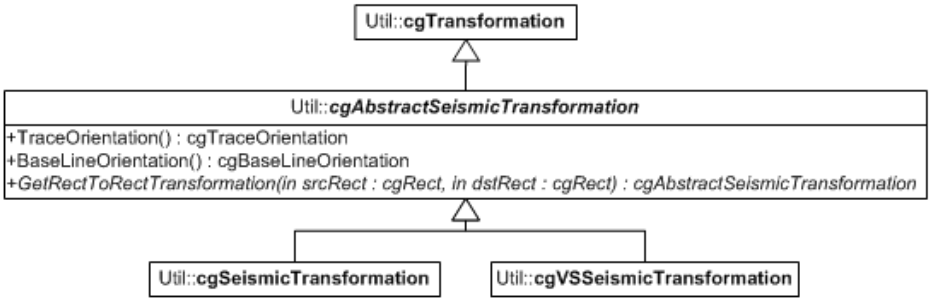
2）加一个重置剖面比例的按钮，添加上事件，再调用上面的一段代码即可。

4.3 重点讲解

4.3.1 剖面中的显示比例设置

剖面中显示比例的设置都在 Transformation 这个特性中设置。

Carnac 本身提供了坐标变换用于在模型坐标与设备坐标之间转换，但 Seismic.NET 里的坐标变换更符合企业标准，即每厘米几道，每厘米几秒。



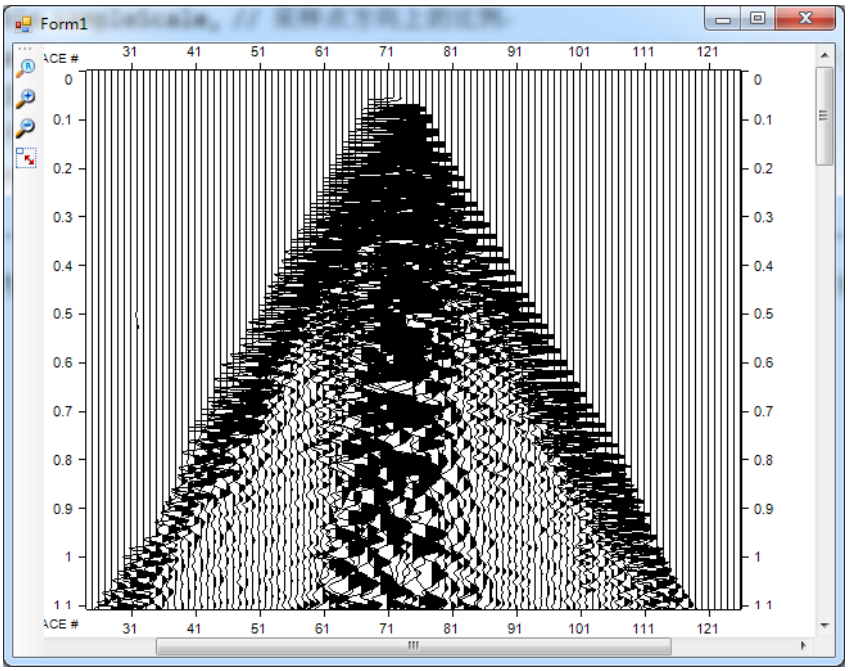
由于常见的剖面通常都是水平方向显示道号，垂直方向上显示时间，所以 plot.SeismicView.Transformation.TraceOrientation 都是从左到右绘制的，plot.SeismicView.Transformation.BaseLineOrientation 是从上到下绘制的。

4.3.2 构造函数

常用的构造函数是：

```
public cgSeismicTransformation (
    cgSeismicMetaData seismicData,  // 地震剖面的元数据
    double traceScale, // 道方向的比例
    cgTraceUnits traceUnits, // 道方向上的单位，国内常用每厘米多少道，即 TracesPerCm
    double sampleScale, // 采样点方向上的比例
    cgSampleUnits sampleUnits,  // 采样点方向上的单位，国内常用的是每秒多少厘米 CmPerSecond
    // 或者每厘米多少米 MetersPerCm (对于深度剖面)
    // 另外两种 FeetPerInch，InchesPerSecond 不用
    cgDeviceResolution resolution //设备分辨率
)
```

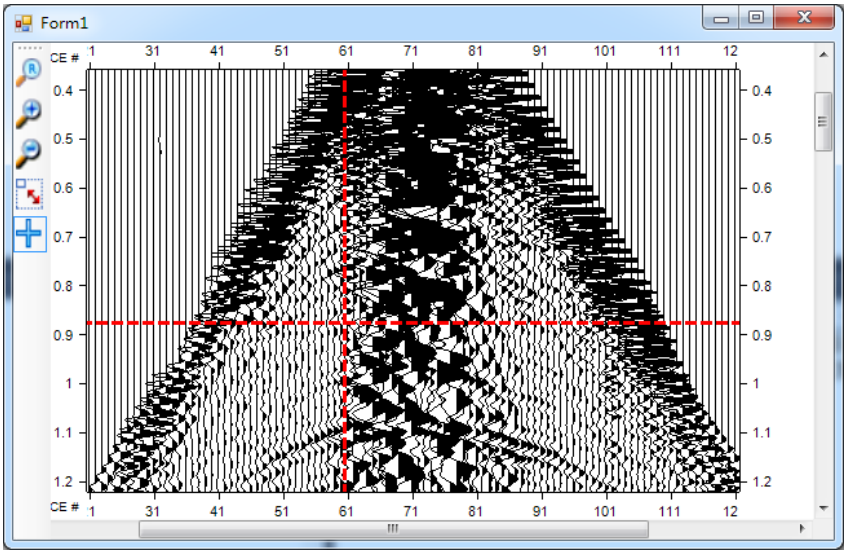
国内常规剖面的标准是每厘米 8 道，每秒 10 厘米（即每 100 毫秒 1 厘米）。



5 例子五：十字光标

5.1 程序功能

因为剖面中显示的内容较多，有时找不到光标的位置，打开十字光标功能，可以清楚地定位光标的位置。



5.2 主要步骤

1）新建一个类 CrossHairView

2）加入两个成员变量

```
private cgSimpleViewAdapter cursorViewAdapter;
private cgCrossHairCursorManipulator cursorMani;
```

3）主要内容在构造函数中

```
public CrossHairView(cgSeismicView view)
{
    Model = new cgSimpleModel();
    Model.SetBoundingBox(new cgRect(view.Model.GetBoundingBox()));
    cursorViewAdapter = new cgSimpleViewAdapter(this, Model as cgShapeContainer);
    cgManipulatorAttributePalette palette = new cgManipulatorAttributePalette();
    cgAttribute attr = palette.GetAttribute("CrossHair");
    attr.SetLineColor(Color.Red);
    attr.SetLineWidth(3);
    cursorMani = new cgCrossHairCursorManipulator(palette);
    view.Views.Add("CursorView", this);
```

```
}
```

4）主窗体的代码里新建一个十字光标对象，并加上事件

```
crossHairView = new CrossHairView(view);
tool.MouseEnter += new System.EventHandler(this.OnCenterPlot_MouseEnter);
tool.MouseLeave += new System.EventHandler(this.OnCenterPlot_MouseLeave);
```

5）鼠标进入事件 OnCenterPlot_MouseEnter

```
crossHairView.CrossHair.Activate(
        new cgSimpleViewAdapter(plot.SeismicView),
        crossHairView.ViewAdapter);
crossHairView.CrossHair.Begin(new cgPoint(0, 0));
```

5）鼠标离开事件 OnCenterPlot_MouseLeave

```
if (crossHairView.CrossHair.IsActive())
{
    crossHairView.CrossHair.Deactivate();
}
```

6）在鼠标移动的代码中增加一段

```
if (crossHairView.CrossHair.IsActive() && crossHairView.CrossHair.IsStarted())
{
    crossHairView.CrossHair.Move(pt);
}
```

5.3 要点说明

5.3.1 十字光标操纵器 cgCrossHairCursorManipulator

在 Carnac 中已经提供了显示十字光标的类，需要一段初始化的代码就可以方便地工作。

首先要建立一个 Model，由于十字光标要贯穿整个剖面显示区，所以把 BoundingBox 设置为地震剖面显示区的大小。

```
    Model = new cgSimpleModel();
    Model.SetBoundingBox(new cgRect(view.Model.GetBoundingBox()));
```

然后建一个 View Adapter，可以从 palette 中取"CrossHair"这个属性设置，根据自己的需要修改颜色、线宽、线型等即可。

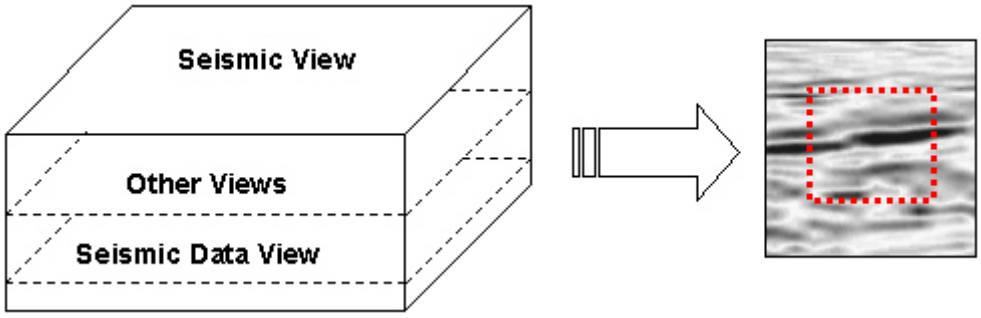
```
cursorViewAdapter = new cgSimpleViewAdapter(this, Model as cgShapeContainer);
cgManipulatorAttributePalette palette = new cgManipulatorAttributePalette();
cgAttribute attr = palette.GetAttribute("CrossHair");
attr.SetLineColor(Color.Red);
attr.SetLineWidth(3);
```

最后创建操纵器，并叠在地震剖面的 View 之上。

```
cursorMani = new cgCrossHairCursorManipulator(palette);
view.Views.Add("CursorView", this);
```

5.3.2 Seismic View

Seismic View 是一种 Stacked View（这是 Carnac 中的一个术语），它里面自己管理着一个 **Seismic Data View**（是一种 simple model view），你还可以在它的上面或下面放一些其它 View，比如显示层位，也可以临时放上操纵器，像拉框放大的橡皮条、十字光标等。**Seismic Data View 内部有缓存** cache 机制来改善剖面显示的性能。



假如你编写了一个层位视图（HorizonView），你可以给它命名并叠在剖面视图上。

```
HorizonView hView = new HorizonView( view );
view.Views.Add( "HorizonView", hView );
```

如果你想访问隐藏在里面的 Seismic Data View，可以用下面的代码：

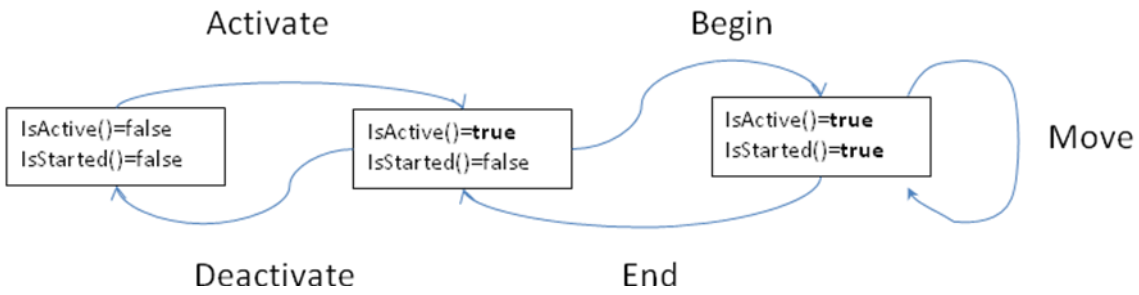
```
cgSimpleModelView dataView = view.Views[cgSeismicView.DATA_VIEW] as cgSimpleModelView;
```

5.3.3 进入和离开

由于我们想让鼠标进入剖面区域时就显示十字，鼠标出去后不留痕迹，所以要在鼠标进入和离开进行事件处理。操纵器的工作顺序是 Activate, Begin, Move, End, Deactivate。实际上你可以在工具栏加上一个按钮来控制显示或隐藏十字光标，分别调用 Activate 和 Deactivate，在进入和离开里调用 Begin 和 End。在后面的例子里会把十字光标封装得更好用

一些，外面几乎不需要了解里面操纵器的任何细节。

当调用了 Activate 之后，IsActive()会返回 true，当调用了 Begin()之后， IsStarted()会为 true，调用 End()后，IsStarted()为 false，调用 Deactivate()后，IsActive()为 false。



简单解释一下这一行语句：

```
crossHairView.CrossHair.Begin(new cgPoint(0, 0));
```

这一行不太严谨，实际上应该在光标进入的地点显示十字光标，但鼠标通常进入后马上会调用 Move 方法，所以鼠标马上会显示在正确的位置上，肉眼看不出有什么不同。

5.4 扩展说明

CursorMani 不仅可以显示十字光标，实际上它还可以只显示垂直线，或者只显示水平线，再以后的例子中可以用下面这样的代码来得到一个选择地震道的交互操作。

```
cursorMani.Type = cgCrossHairCursorManipulator.EnType.VERT;
```

6 例子六：封装十字光标

6.1 程序功能

将十字光标的功能封装起来，外界基本不需要了解这个十字光标内部的实现。

6.2 主要步骤

1）将 Seismic View, Adapter 和 Manipulator 都封装在类里作为私有变量

在上一个例子中，外面主窗体的代码还要访问这几个类，容易使用错，所以最好的办法就是将这些内容全部封装起来

2）把 Activate()、Begin()、End()、Deactivate()都封装为无参数的方法

外界代码只需要在相应的事件里调用即可，内部自动维护 adapter 和 manipulator 的状态，Move()方法还需要一个参数。

```
public void Activate()
{
    if (!cursorMani.IsActive())
        cursorMani.Activate(new cgSimpleViewAdapter(seismicView), cursorViewAdapter);
}

public void Begin()
{
    if (cursorMani.IsActive())
        cursorMani.Begin(new cgPoint(0, 0));
}

public void Move(cgPoint pt)
{
    if (cursorMani.IsActive() && cursorMani.IsStarted())
        cursorMani.Move(pt);
}

public void End()
{
    cursorMani.End();
}

public void Deactivate()
{
```

```
    if (cursorMani.IsActive())
        cursorMani.Deactivate();
}
```

3）处理工具栏上的十字光标按下事件

```
if (toolStripCrossHair.Checked)
    crossHairView.Activate();
else
    crossHairView.Deactivate();
```

4）鼠标进入和离开的事件

```
//鼠标进入时
crossHairView.Begin();
//鼠标离开时
crossHairView.End();
```

6.3 要点说明

可以看到主窗体调用 crossHairView 的代码变得非常一致，也很简洁，这样不容易发生错误，也很容易重用了。

你可以试试在鼠标离开的时候，不调用 End()会发生什么情况。

7 例子七：漫游拖动剖面

7.1 程序功能

出现一个手状图标，拖动剖面。

7.2 主要步骤

1）Carnac 中提供了 Panning 的操纵器

```
panningManipulator = new cgPanningManipulator();
```

2）响应工具栏上的事件，激活操纵器，并改变光标形状

点击漫游图标后，激活操纵器。

```
panningManipulator.Activate(new cgSimpleViewAdapter(plot.PrimaryView));
```

光标改为手状。主要思路是通过 center Plot，找到它的 cgWinFormTool，然后改变 tool 的 Cursor 属性。

```
Common.cgPlot centerPlot = plot.GetAnnotation(cgGenericPlotLayout.CENTER) as
Common.cgPlot;
Common.cgWinFormTool tool = (Common.cgWinFormTool)(centerPlot.Tool);
tool.Cursor = Cursors.Hand;
```

3）处理 OnCenterPlot_MouseDown、MouseMove、MouseUp 三个事件

与上一章的十字光标类似，也需要分别调用 Begin()、Move()和 End()。

```
// OnCenterPlot_MouseDown
if (panningManipulator.IsActive())
    panningManipulator.Begin(pt);

// OnCenterPlot_MouseMove
if (e.Button == MouseButtons.Left && panningManipulator.IsActive()
                && panningManipulator.IsStarted())
    panningManipulator.Move(pt);

// OnCenterPlot_MouseUp
if (e.Button == MouseButtons.Left && panningManipulator.IsActive())
    panningManipulator.End();
```

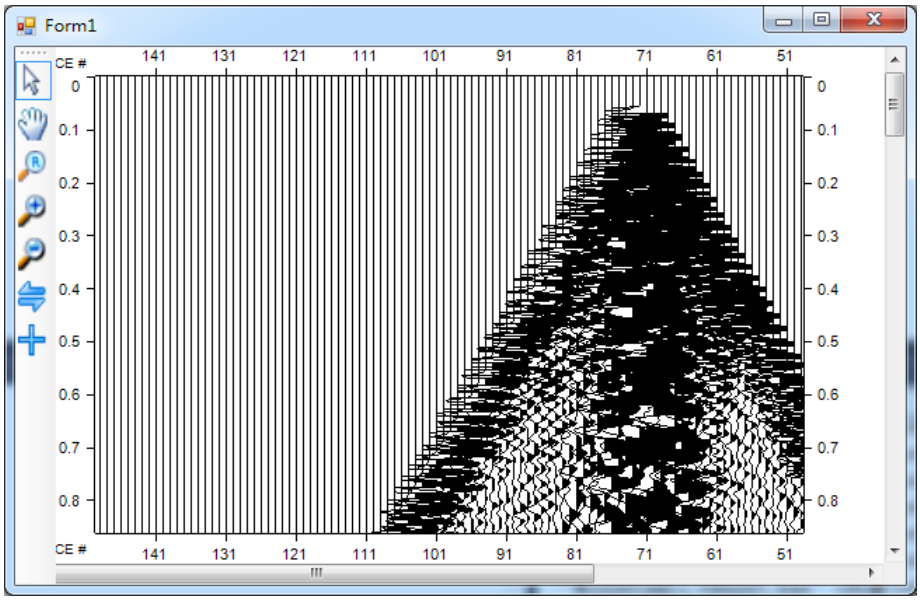
4）加一个工具栏图标，解除漫游状态

拉框放大操作和漫游操作是互斥的，所以需要加一个按钮分别激活不同的操纵器，如果 2 个操纵器同时激活，则在鼠标拖动时会出现奇怪的现象。

8 例子八：道反序显示

8.1 程序功能

工具栏上加一个按钮，可以反序显示地震道。



8.2 主要步骤

1）加一个工具栏按钮

2）响应此按钮的事件

```
cgSeismicTransformation trans = (cgSeismicTransformation)view.Transformation;

// inversed 是指 RightToLeft 或 BottomToTop
if (trans.TraceOrientation.IsInversed())
    view.Pipeline.Image.SetOrientation(trans.BaseLineOrientation,
                        cgTraceOrientation.LeftToRight, cgTracePolarity.Normal);
else
    view.Pipeline.Image.SetOrientation(trans.BaseLineOrientation,
                        cgTraceOrientation.RightToLeft, cgTracePolarity.Normal);
```

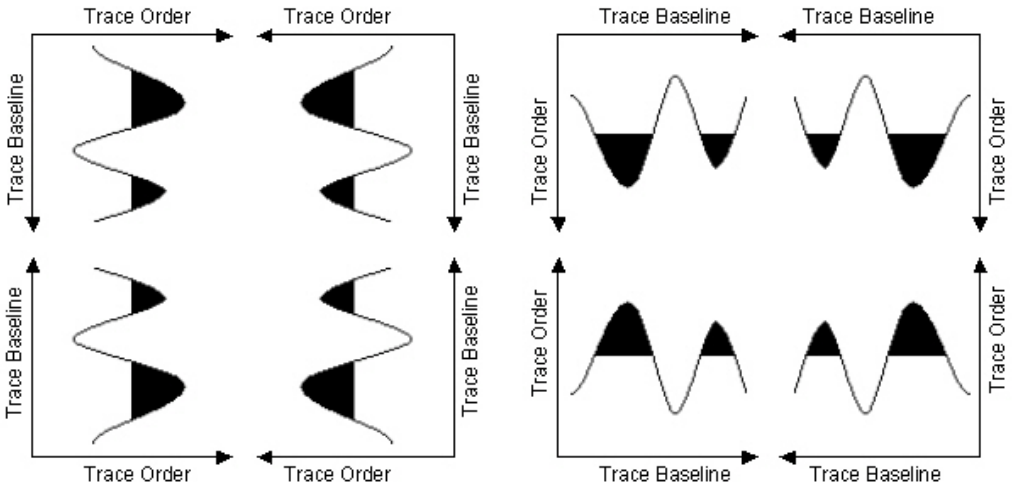
8.3 要点解释

需要注意 cgSeismicTransformation 中有 TraceOrientation 这个属性，但改变它并不能让剖面反序显示。

```
 //注意下面这条语句不起作用
```

```
trans.TraceOrientation = cgTraceOrientation.RightToLeft;
```

剖面显示的时候，内部是生成了一幅图像，Seismic.NET 中专门有一个类叫 cgSeismicImage，它可以支持多种显示方向。



它的 3 个重要属性 TraceOrientation、BaseLineOrientation 和 TracePolarity 分别用于控制显示的方向以及填充正极性还是负极性。这 3 个属性是只读的，需要通过 SetOrientation 改变显示的方向和极性。

```
public virtual void SetOrientation (
    cgBaseLineOrientation baseOrientation,
    cgTraceOrientation traceOrientation,
    cgTracePolarity tracePolarity
)
```

看一看 cgBaseLineOrientation 的定义。

```
public sealed class cgBaseLineOrientation : cgAbstractOrientation
{
    public static readonly cgBaseLineOrientation BottomToTop;
    public static readonly cgBaseLineOrientation LeftToRight;
    public static readonly cgBaseLineOrientation RightToLeft;
    public static readonly cgBaseLineOrientation TopToBottom;

    public static cgBaseLineOrientation GetOrientation(bool vertical, bool inversed);
}
```

看一看 cgTracePolarity 的定义：

```
public enum cgTracePolarity
```

```
{
        Normal = 0,
        Inverse = 1,
}
```

9 例子九：输出 CGM

9.1 程序功能

可以将剖面输出为石油行业标准的 CGM 图像文件。

9.2 主要步骤

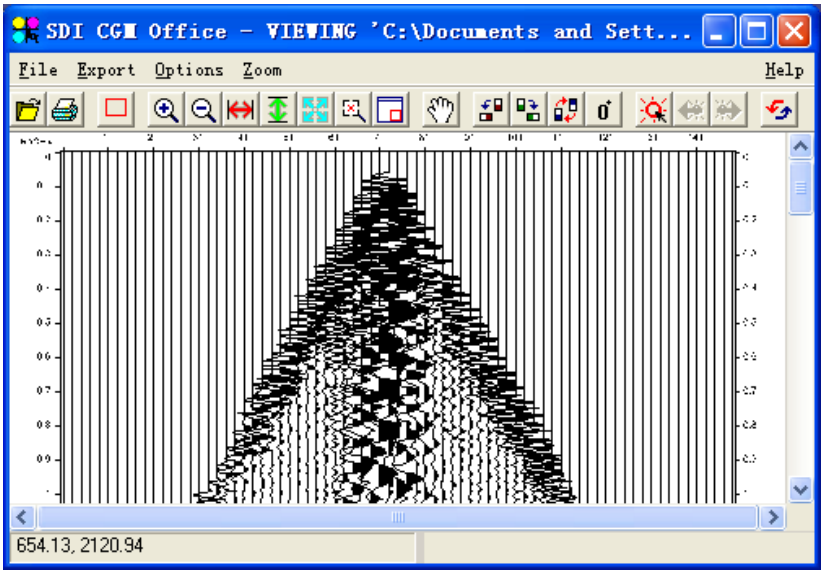
1）加上 CGM.NET 和 CGMSeismic.NET 库的引用

2）工具栏上加一个保存 CGM 的按钮

3）编写保存 CGM 相关的事件代码

```
SaveFileDialog saveDlg = new SaveFileDialog();
saveDlg.Filter = "CGM 文件(*.cgm)|*.cgm|所有文件(*.*)|*.*";
if (saveDlg.ShowDialog() == System.Windows.Forms.DialogResult.OK)
{
    string cgmFilename = saveDlg.FileName;
    using (FileStream stream = new FileStream(cgmFilename, FileMode.OpenOrCreate))
    {
        cgCGMSeismicWriter cgm = new cgCGMSeismicWriter();
        cgm.Stream = stream;
        cgm.TextMode = cgCGMTextMode.SHAPE;
        cgm.Visual = plot;
        cgm.Write();
    }
}
```

4）可以安装 CGM 浏览软件打开此 CGM 文件。



9.3 要点说明

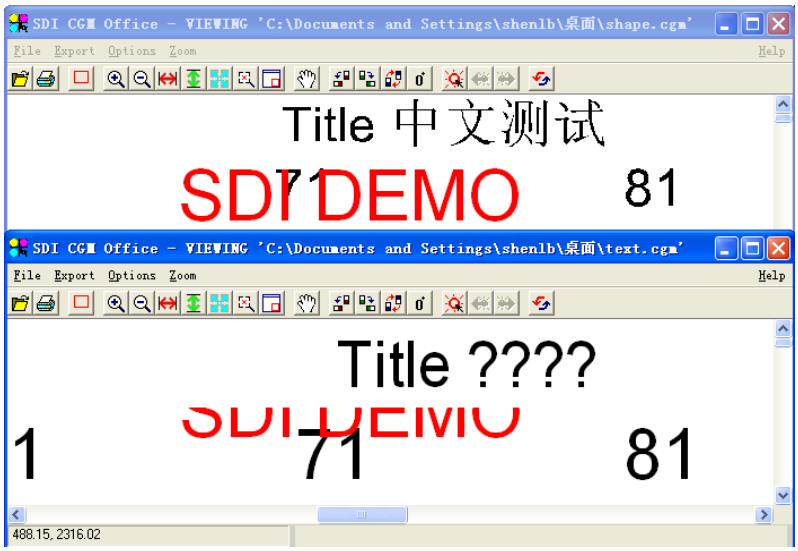
Carnac.NET 和 Seismic.NET 中的所有 plot 都可以输出为 CGM 文件。输出过程也相当简单，只需要指明输出文件名、文本编码方式和 plot 即可。

```
cgCGMWriter cgm = new cgCGMWriter();
cgm.FileName = “yourfilename.cgm”;
cgm.TextMode = cgCGMTextMode.TEXT; // 不推荐这种方式，无法显示中文
cgm.Visual = plot;
cgm.Write();
```

cgm.TextMode 表示图形中文本的编辑方式，可以用图形，也可以保留按文本编码。

```
public enum cgCGMTextMode
{
    TEXT = 0,
    SHAPE = 1,
}
```

对照下面的 2 个 CGM 文件，第一个图是按 Shape 方式输出，第二个图是按 TEXT 方式输出，可以看出文本方式的比较平滑。另外更重要的是 CGM 对中文支持不好，所以建议按 Shape 输出，否则可能显示乱码。


SDI CGM Office - VIEWING 'C:\Documents and Settings\shenlb\桌面\shape.cgm'
File Export Options Zoom Help
Title 中文测试
SDI DEMO
1
71
81
SDI CGM Office - VIEWING 'C:\Documents and Settings\shenlb\桌面\text.cgm'
File Export Options Zoom Help
Title ????
1
71
81
488.15, 2316.02

10 例子十：重构 zoom 和 panning

10.1 程序功能

将拉框放大和漫游功能重构，分别写出一个类，使主程序更简洁。

10.2 主要步骤

1）封装类 ZoomController

将以前建立 cgDragRectangleManipulator 的一系列过程封装起来，将 Activate、Begin、Move、End 以及 Deactivate 都封装起来，让外面的主程序调用更方便。

2）封装类 PanningController

与上面类似把漫游功能也封装起来，同时把改变光标形状的代码也放在这里面。

3）外面主窗体的代码更简洁

只需要在 plot 上面加上 2 个 Controller 即可，因为 zoom 和 panning 是互斥的，所以在程序启动时默认激活 zoomController。

```
// 拉框放大控制
zoomController = new ZoomController(plot);
zoomController.Activate();

// 漫游拖动
panningManipulator = new PanningController(plot);
```

其它几个事件的代码只需要简单调用 Activate()、Begin()、Move()、End()以及 Deactivate()即可，几个控制器内部会根据状态处理好 manipulator。

下面是处理鼠标移动事件的代码，可以看到只要调用各自的 Move 方法即可，如果这些 manipulator 处于 Active 和 Started 状态，Move 才会起作用。

```
    zoomController.Move(pt);
    panningManipulator.Move(pt);
    crossHairView.Move(pt);
```

11 例子十一：状态栏显示测线号和 CDP 号

11.1 程序功能

鼠标移动时，显示出当前处于的道号、测线号、CDP 号和时间值。

11.2 主要步骤

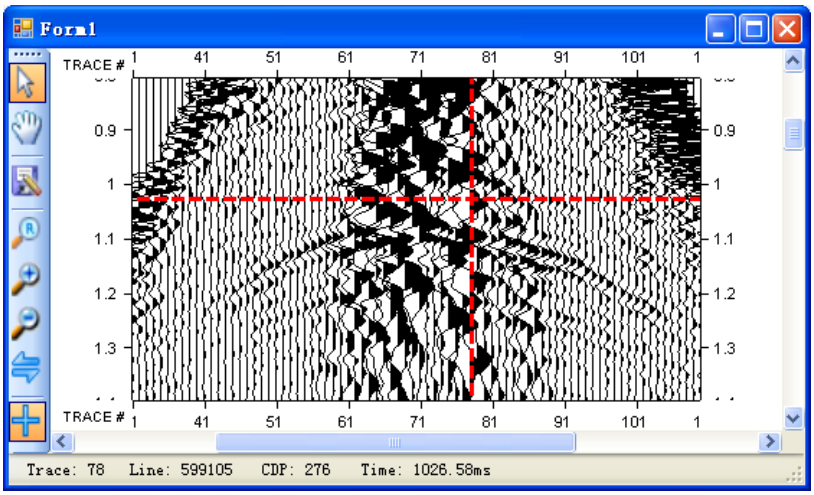
1）在主窗体上加状态栏，再加 4 个 ToolStripStatusLabel，名称分别为 statusTrace、statusTime、statusLine 和 statusCDP，将控件宽度都调为 100。

2）在鼠标移动事件的方法中添加如下代码

```
try
{
    // 从 0 开始的道号
    int traceIdx = view.CoordinatesConverter.GetTraceIndex((int)pt.x, (int)pt.y);
    statusTrace.Text = "Trace: " + traceIdx.ToString() + " ";

    double timeVal = view.CoordinatesConverter.GetSampleLocation((int)pt.x,
(int)pt.y);
    statusTime.Text = "Time: " + (timeVal * 1000).ToString("####.##") + "ms";
    string line =
view.Pipeline.SeismicReader.GetTraceMetaData(traceIdx).GetField(203).ToString();
    string cdp =
view.Pipeline.SeismicReader.GetTraceMetaData(traceIdx).GetField(205).ToString();
    statusLine.Text = "Line: " + line;
    statusCDP.Text = "CDP: " + cdp;
}
catch (cgConverterException)
{
    statusTrace.Text = "Trace: N/A ";
    statusTime.Text = "Time: N/A";
    statusLine.Text = "Line: N/A";
    statusCDP.Text = "CDP: N/A";
}
```

3）运行程序，观察鼠标移动时状态栏的变化

同时也试试反序显示时，状态栏是否显示正确。



11.3 重点讲解

1）坐标转换 CoordinatesConverter

在剖面中的坐标转换统一用 CoordinatesConverter，因为剖面有多种显示方向，所以用 Transformation 求的坐标会不准确。

GetTraceIndex()可以取得道号（从 0 开始），GetSampleLocation()取得采样点的时间值，GetSampleIndex()可以取得采样点的序号（也是从 0 开始）。

2）取道头字信息

测线号、CDP 号这些信息都放在 SEGY 的道头字中，不同公司处理完成后的 SEGY 中道头字放得有些不同，所以有时取不出正确的测线号和 CDP 号。

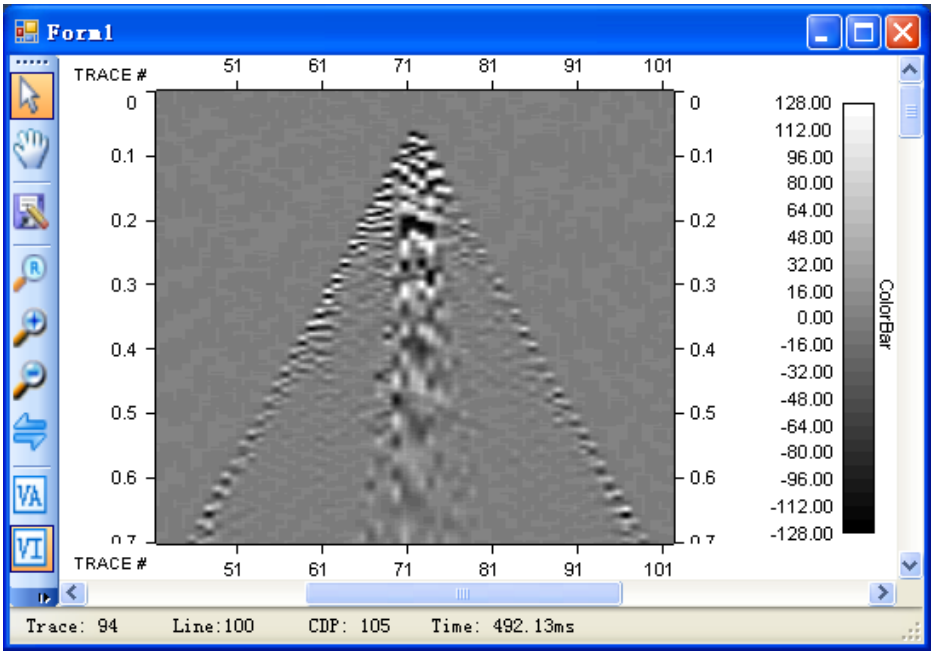
```
view.Pipeline.SeismicReader.GetTraceMetaData(traceIdx).GetField(203);
view.Pipeline.SeismicReader.GetTraceMetaData(traceIdx).GetField(205);
```

这 2 行代码中的 203 和 205 会有点费解，这里只说明是道头的 Identifier，SEGY 有 240 字节的道头，根据不同的标识数字，可以取出相应的道头信息来，详细内容放在第 15 章读取 SEGY 信息中讲解。

12 例子十二：变密度与彩色显示

12.1 程序功能

支持在变面积与变密度彩色显示方式之间切换。



12.2 主要步骤

1）工具栏上加 2 个按钮

VA分别用于控制变面积，VI用于变密度显示。

2）变面积

```
pipeline.TraceRasterizer.PlotType = cgSeismicPlotType.PositiveFillAndWiggle;
// 不显示颜色棒
plot.Annotations.ColorBar.Position = cgAnnotationPosition.None;
```

3）变密度

此时，一般需要显示颜色棒。

```
pipeline.TraceRasterizer.PlotType = cgSeismicPlotType.InterpolatedDensity;
// 右侧显示颜色棒
plot.Annotations.ColorBar.Position = cgAnnotationPosition.Right;
```

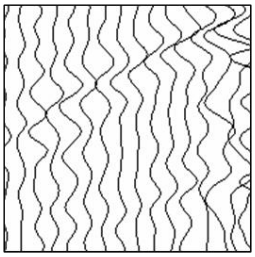
4）运行程序，切换 2 种显示方式

12.3 重点说明

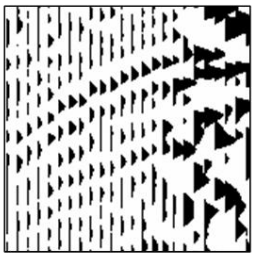
1）地震道的显示方式设置

pipeline.TraceRasterizer.PlotType 可以设置地震道的各种显示方式。

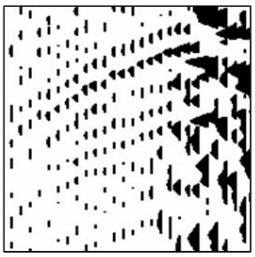
```
public enum cgSeismicPlotType {
    None = 0, // 不显示任何内容
    Empty = 0, // 过时不用了，用 None
    Wiggle = 1, // 只显示地震道
    PositiveFill = 2, //正向填充
    PositiveFillAndWiggle = 3, // 正向填充，并显示地震道
    NegativeFill = 4, //负向填充
    NegativeFillAndWiggle = 5, // 负向填充，并显示地震道
    PositiveAndNegative = 6, // 正向、负向都填充
    Density = 8, // 变密度
    DensityAndWiggle = 9, // 变密度，并显示道
    DensityAndPositiveFill = 10, // 变密度，正向填充
    DensityAndNegativeFill = 12, // 变密度，负向填充
    InterpolatedDensity = 16, // 内插的变密度
    PositiveColorFill = 32, // 正向用颜色填充
    NegativeColorFill = 64, // 负向用颜色填充
    ColorFill = 97, // 正向和负向都用颜色填充，并显示地震道

    // 大块颜色填充，在每一小块区域内是一种颜色，颜色值是由该区域内的最大振幅决定的
    LobeFill = 128,
}
```



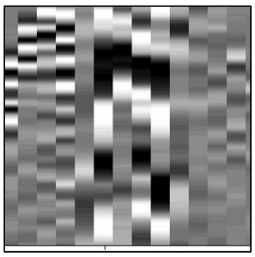
1: Wiggle



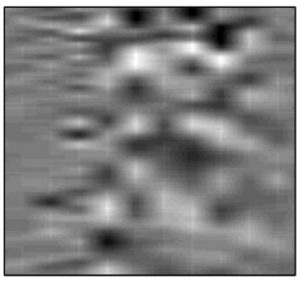
2: PositiveFill



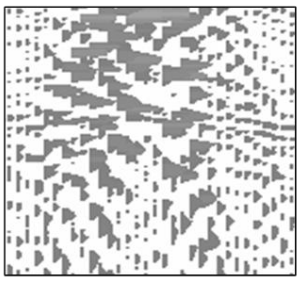
4: NegativeFill



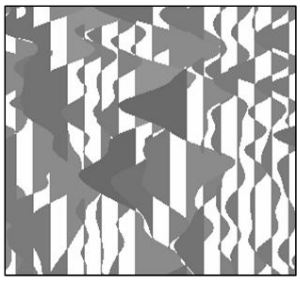
8:Density



16: InterpolatedDensity



32: PositiveColorFill



128: LobeFill

2）TraceRasterizer 还有一些其它可以设置的属性

	Name	Description
	Antialiasing	Turns on or off the wiggle antialiasing mode.
	ClippingValue	Gets or sets the clipping value that determines to which extent traces are displayed.
	ColorMap	Sets or Gets the color map to use by the rasterizer
	ColorMapping	Gets or sets the color mapping strategy (to be) used by the rasterizer.
	DecimationThreshold	Gets or sets the auto decimation threshold in pixels.
	DeltaPixels	Sets or gets the spacing between traces in screen coordinates (pixels)
	DensityDeltaPixels	Sets or gets the spacing between density traces in screen coordinates (pixels)
	Pipeline	Gets or sets the pipeline that contains this rasterizer.
	PlotType	Sets or Retrieves the plot type used to draw the traces
	ReverseFill	Sets or returns the reverse fill flag indicating using reverse order of colors for positive and negative filling including monochrome filling.
	TraceHighlighter	Gets or set the trace highlighter.

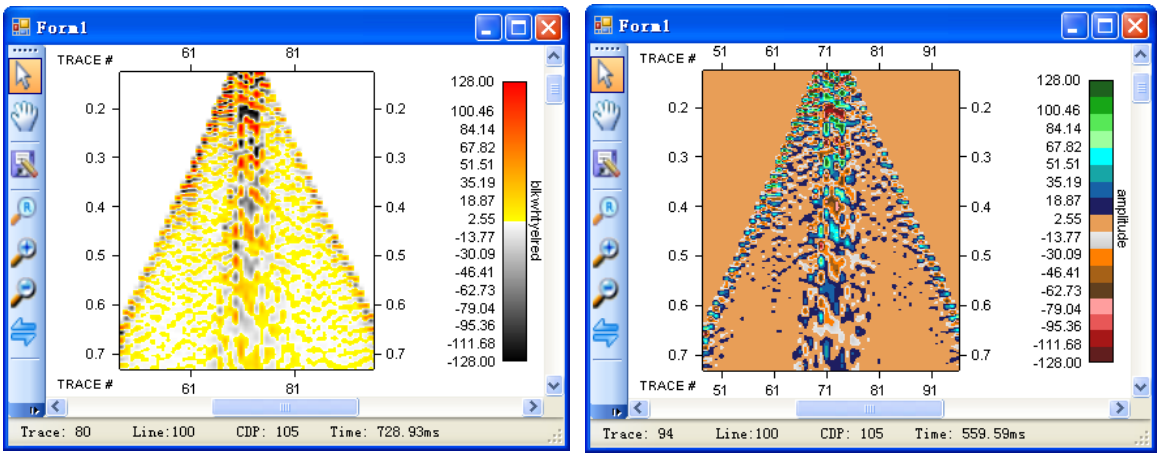
3）试一试把颜色棒放在其它位置

修改 plot.Annotations.ColorBar.Position 即可。

13 例子十三：改变颜色棒

13.1 程序功能

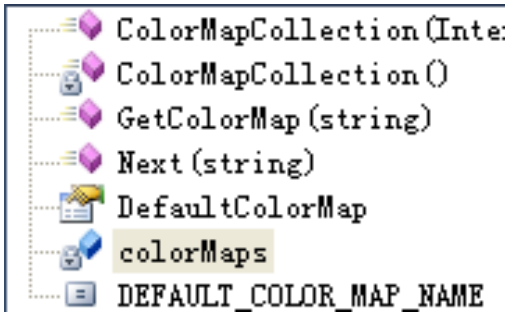
内置了 16 种颜色棒，单击一次按钮，切换到下一种颜色棒。



13.2 主要步骤

1）新建一个类 ColorMapCollection

这个类里面实现 16 种常用颜色表，这 16 种颜色棒借鉴了国内外主流的地震解释软件。第一个颜色表的名称为 DefaultColorMap，可以根据名称取得颜色表 GetColorMap，调用 Next() 可以得到颜色表集合中的下一个颜色表。



2）主窗体中加上一段代码

```
// 设置色棒
plot.Annotations.ColorBar.Text = ColorMapCollection.DEFAULT_COLOR_MAP_NAME;
pipeline.TraceRasterizer.ColorMap = ColorMapCollection.DefaultColorMap;

// 建立颜色表映射机制，剖面程序本身也建立了一个默认的颜色映射表，我们可以替换掉它
cgOrigAmplColorMapping mapping = new cgOrigAmplColorMapping();
double min = seismicReader.MetaData.MinimumAmplitude;
```

```
double max = seismicReader.MetaData.MaximumAmplitude;
mapping.SetValueRange(min, max);
mapping.Apply = true;
pipeline.TraceRasterizer.ColorMapping = mapping;
```

3）工具栏上加一个按钮，编写相应事件的代码

当为变密度显示时，可以点击它，切换到下一个颜色棒。

```
// 取得当前使用的颜色表的名称
string colorMapName = plot.Annotations.ColorBar.Text;
// 取得下一个颜色表的名称
string nextColorMap = ColorMapCollection.Next(colorMapName);
// 根据名称得到颜色表，并赋给剖面
pipeline.TraceRasterizer.ColorMap = ColorMapCollection.GetColorMap(nextColorMap);
// 记得把颜色棒的文本修改，因为它用于存储当前颜色表的名称
plot.Annotations.ColorBar.Text = nextColorMap;
```

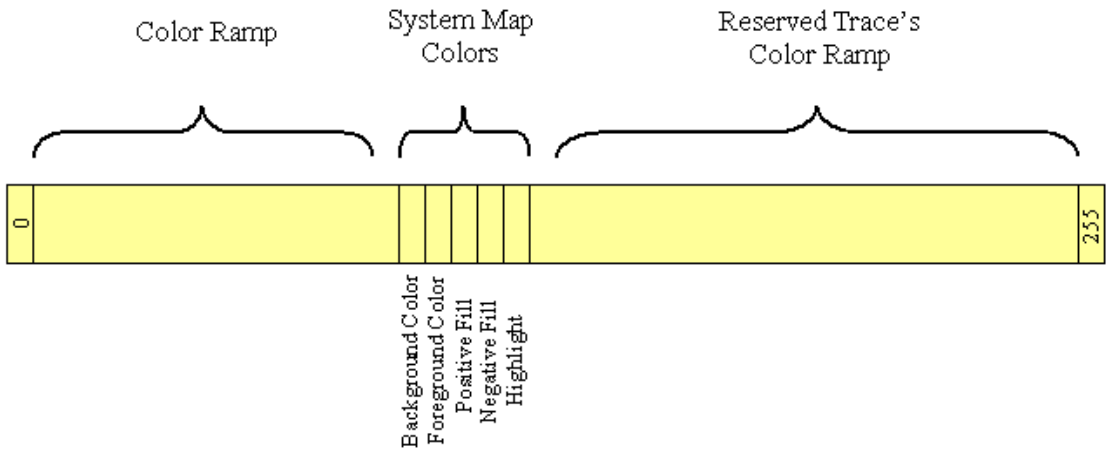
4）运行程序，切换 16 种颜色棒

当到最后一个颜色棒之后，再点击会切换到第一个颜色棒。

13.3 重点说明

1）颜色表 ColorMap

Carnac.NET 中就提供了 ColorMap 类，在 Seismic.NET 中有一个增强的类 cgSeismicColorMap。



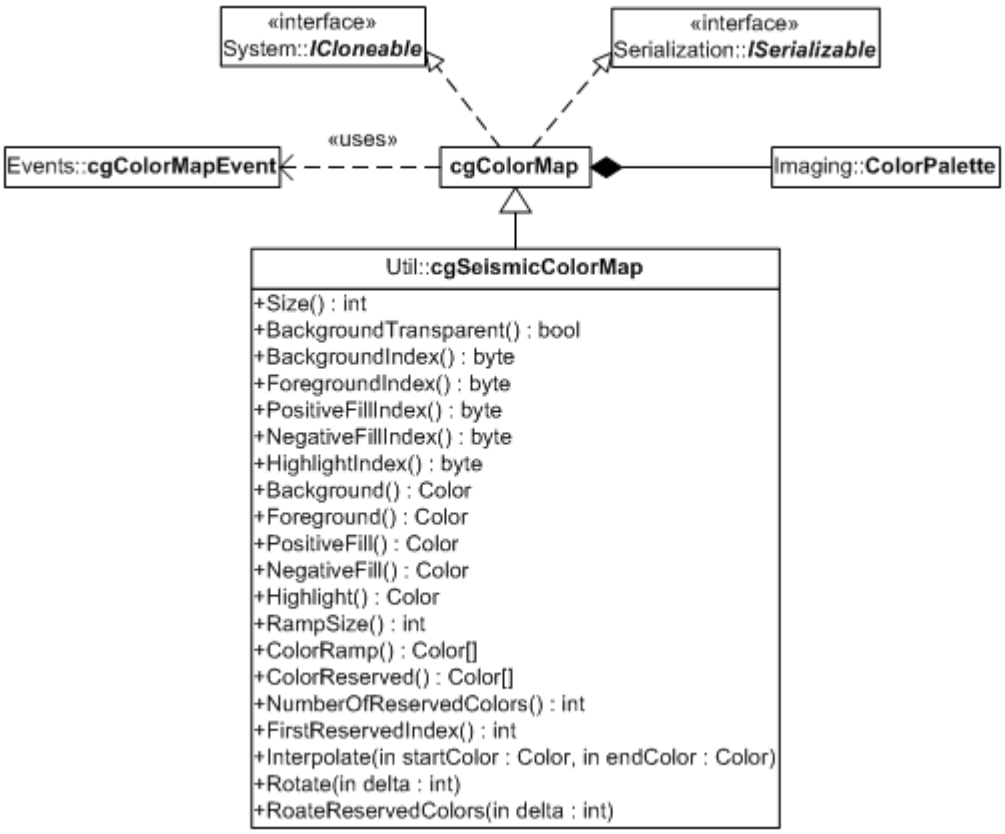
ColorMap 最多支持 256 种颜色，cgSeismicColorMap 内置了 5 种系统颜色，称为 System Map Colors，包括背景色 background，前景色 foreground，正向填充色 positive fill，负向填充色 negative fill 及高亮色 highlight。

在一些特殊的绘制需求中需要用到 Reserved Color Ramp，例如把一组道的颜色用特殊的颜色显示时。

要注意 cgSeismicColorMap 构造函数的使用，通常下面这种方式：

```
cgSeismicColorMap colorMap = new cgSeismicColorMap(256);
```

这条构造函数表示这个 colorMap 共有 256 种颜色，有 5 种是系统颜色，你只能修改前面 251 种（从 0 到 250）。



Interpolate()是把 2 种颜色之间插入渐变色的方法，下面这条语句表示在第 0 号颜色为黑色，第 125 号颜色为白色，中间内插渐变色。

```
colorMap.Interpolate(0, Color.Black, 125, Color.White);
```

2）颜色棒的名称

程序需要记住当前正在使用的颜色棒，这里直接使用了 plot.Annotations.ColorBar.Text 属性。

3）颜色映射表 ColorMapping

不要把 ColorMapping 与 ColorMap 弄混了，剖面中可能会有成千上万种振幅值，需要映射

到这仅有的 256 种颜色表中，这就是 ColorMapping 的工作。

你只需要构造一个 cgOrigAmplColorMapping 对象，并把映射值的范围设置好即可，记得要把这些映射表赋给剖面的 pipeline.TraceRasterizer.ColorMapping。

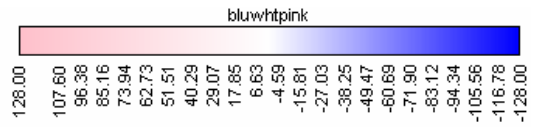
```
pipeline.TraceRasterizer.ColorMapping = mapping;
```

13.4 要点回顾

颜色表 ColorMap：就像一个调色板，里面就保存了许多颜色。

颜色映射表 ColorMapping：用于把不同的振幅值映射到不同的颜色上

颜色棒 ColorBar：通常放在剖面右侧（有些人习惯放在底部）的颜色条，旁边还标有刻度，必须有 ColorMap 和 ColorMapping 才能把颜色棒画出来。



13.5 存在的问题

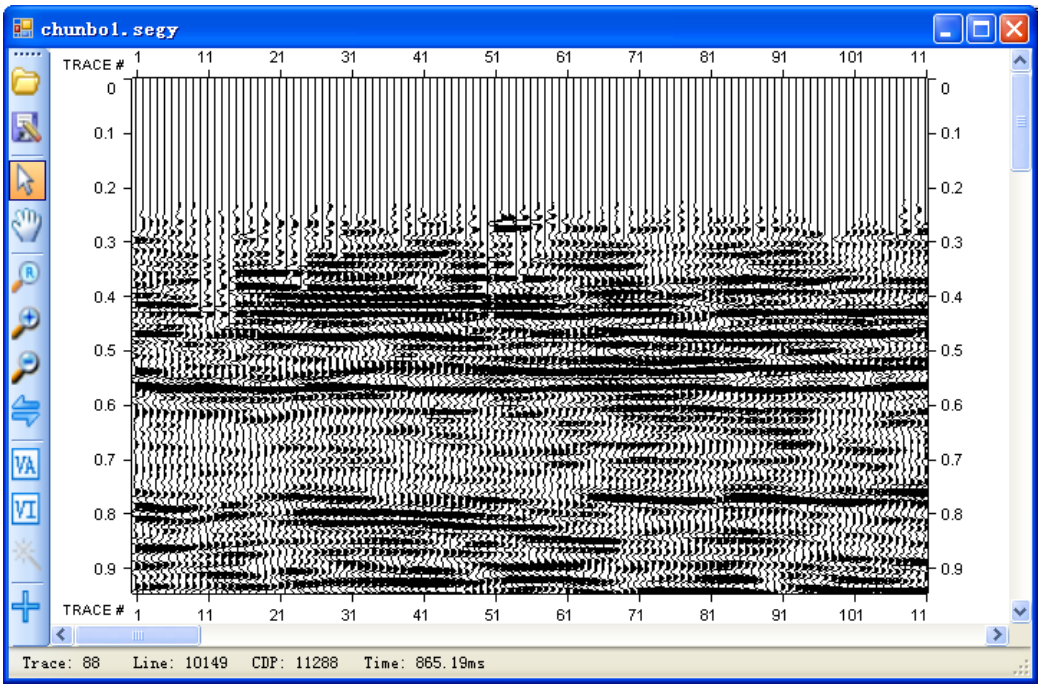
本来想在颜色棒上加上单击事件，但事件没有响应，可能是 eastPlot 不对。

```
Common.cgWinFormTool toolColorBar = new Common.cgWinFormTool();
//Common.cgPlot eastPlot = plot.Annotations.ColorBar as Common.cgPlot;
eastPlot = plot.GetAnnotation(cgGenericPlotLayout.EAST) as Common.cgPlot;
eastPlot.Tool = toolColorBar;
toolColorBar.MouseDown += new MouseEventHandler(MouseDownColorBarEventHandler);
```

14 例子十四：打开任意 SEGY 文件

14.1 程序功能

以前的例子都是打开一个指定的 SEGY 文件，此例子可以打开任何符合规范的 SEGY 文件，窗口的标题栏是 SEGY 文件的名称，不过本程序仍是一个单窗口程序。



14.2 主要步骤

1）工具栏上加一个“打开”按钮

2）把以前初始化 Seismic 所有对象的方法封装起来

```
/// <summary>
/// 构建 Seismic.NET 的一些成员变量
/// cgScrollablePlotPanel plotControl 一个可视的控件
/// cgSeismicPlot plot 一个没有可视窗口的绘图区
/// cgSeismicView view 一个视图
/// cgSeismicPipeline pipeline 地震道处理的流水线
/// 里面还加上一些默认的事件处理
/// </summary>
/// <param name="seismicReader"></param>
/// <returns>一个可视的 PlotPanel</returns>
private cgScrollablePlotPanel InitializeSeismicComponent(
                        cgSeismicReader seismicReader)
```

```
{
    pipeline = new cgSeismicPipeline(seismicReader);

    view = new cgSeismicView(pipeline);
    // 设置为标准剖面显示比例
    view.Transformation = new cgSeismicTransformation(
                view.Pipeline.SeismicReader.MetaData,
                8, cgTraceUnits.TracesPerCm,
                10, cgSampleUnits.CmPerSecond,
                new cgGDIPlusDriver());

    plot = new cgSeismicPlot(view,
                cgTraceAxisPosition.Both,   // 上下都显示道号轴
                cgSampleAxisPosition.Both); // 左右都显示时间轴

    // 默认的显示方式是变面积显示，不显示颜色棒，但设置色棒也不会出错
    plot.Annotations.ColorBar.Position = cgAnnotationPosition.None;
    plot.Annotations.ColorBar.Text = ColorMapCollection.DEFAULT_COLOR_MAP_NAME;
    pipeline.TraceRasterizer.ColorMap = ColorMapCollection.DefaultColorMap;

    // 建立颜色表映射机制，剖面程序本身也建立了一个默认的颜色映射表，我们可以替换掉它
    cgOrigAmplColorMapping mapping = new cgOrigAmplColorMapping();
    double min = seismicReader.MetaData.MinimumAmplitude;
    double max = seismicReader.MetaData.MaximumAmplitude;
    mapping.SetValueRange(min, max);
    mapping.Apply = true;
    pipeline.TraceRasterizer.ColorMapping = mapping;

    // 漫游拖动控制，拉框放大控制，十字光标 View。里面都在 seismicView 上叠了一层 view
    panningController = new PanningController(plot);
    zoomController = new ZoomController(plot);
    crossHairView = new CrossHairView(view);

    // 在中间的剖面显示区上注册一个右键、漫游拖动、拉框放大等事件
    // cgSeismicPlot 中有 5 个 cgPlot 构成，中间、东、西、南、北
    Common.cgPlot centerPlot = plot.GetAnnotation(cgGenericPlotLayout.CENTER) as
Common.cgPlot;

    Common.cgWinFormTool tool = new Common.cgWinFormTool();
    centerPlot.Tool = tool;
    tool.MouseDown += new MouseEventHandler(OnCenterPlot_MouseDown);
    tool.MouseMove += new MouseEventHandler(OnCenterPlot_MouseMove);
```

```
    tool.MouseUp += new MouseEventHandler(OnCenterPlot_MouseUp);

    // 下面这 2 个事件与十字光标有关系
    tool.MouseEnter += new System.EventHandler(OnCenterPlot_MouseEnter);
    tool.MouseLeave += new System.EventHandler(OnCenterPlot_MouseLeave);

    // 创建一个 PlotPanel 用来容纳 plot，这个 plot 本身是无窗口的
    cgScrollablePlotPanel plotControl = new cgScrollablePlotPanel(plot);
    plotControl.Dock = DockStyle.Fill;
    plotControl.PrimaryView.Background.SetFill(Color.White);
    return plotControl;
}
```

3）打开 SEGY 文件的代码

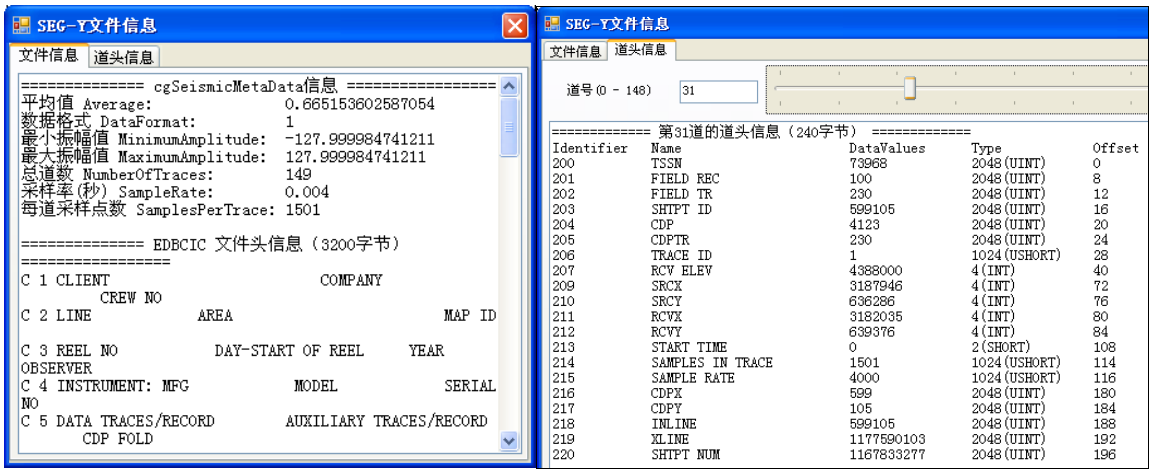
```
reader = new cgSegyReader(segyFilename);
cgScrollablePlotPanel plotControl = InitializeSeismicComponent(reader);
seismicPanel.Controls.Clear(); //把以前的 plotControl 删除掉
seismicPanel.Controls.Add(plotControl);
this.Text = Path.GetFileName(segyFilename); //在主窗体的标题栏上显示 SEGY 文件名
```

4）试着打开其它 SEGY 文件

15 例子十五：读取 SEGY 的详细信息

15.1 程序功能

可以显示 SEGY 文件的文件头和道头信息。



15.2 主要步骤

1）新建一个窗体 SegyInfoForm

构造函数需要传入一个 SegyReader。

```
public SegyInfoForm(cgSeismicReader segyReader)
```

初始化时，读取 SEGY 文件的文件头元数据和第 0 个道头信息。

2）主窗体上加一个按钮

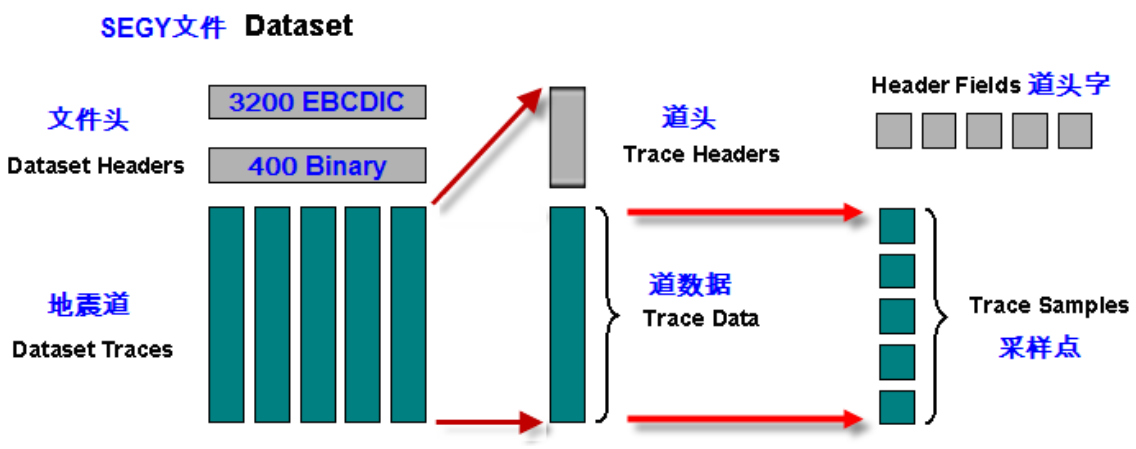
3）编写点击该按钮的事件

```
SegyInfoForm segyInfoForm = new SegyInfoForm(reader);
segyInfoForm.Show();
```

15.3 重点讲解

1）cgSeismicReader

这个类可以读取 SEGY 文件、内存 Segy 文件，甚至可以自定义一种格式，当然最常用的还是 cgSegyReader。



通过上图简单介绍一下 SEGY 格式，一个 SEGY 文件也可称为一个 Dataset，里面有几个文件头(这里叫 Dataset Headers),SEGY 分为 3200 个 EBCDIC 文本头和 400 字节的二进制 Binary 头，然后就顺序存放所有的地震道(Dataset Traces)。

一个地震道由道头(Trace Headers)和道数据(Trace Data)组成，SEGY 里有 240 个字节的道头字，道数据里有若干个采样点数据，采样点的存放格式通常是 32 位 IBM 浮点数。

2）cgSeismicMetaData

Seismic.NET 中已经封装了一些信息，可以直接用 cgSeismicMetaData 得到 SEGY 文件的元数据，比较常用的有：

Average：平均值；

DataFormat：数据格式；

MinimumAmplitude：最小振幅值；

MaximumAmplitude：最大振幅值；

NumberOfTraces：总道数；

SampleRate：采样率(秒)；

SamplesPerTrace：每道采样点数。

好像有些时候统计得不太准（或者 Reader 并没有统计），需要自己统计，用 SetDataStatistics()方法把 min、max、average 和 rms(root-mean-square)值放进去。

Name	Description
Average	Retrieves the average for the absolute sample values in this dataset.
ConstantSamplesPerTrace	Determines whether the number of samples per trace is constant throughout the dataset.
DataFormat	Gets and sets the trace data format.
DeltaTrace	Sets or gets the interval between two traces in model coordinates.
HeaderValues	Retrieves the hash table used to store values of pre-defined header fields. You will use this method only for developing your own reader of seismic data.
IsStartValueConstant	Indicates if the start value (usually time) is the same for every trace.
MaximumAmplitude	Retrieves an estimate of the maximum amplitude value for the dataset. Normalization and other trace processing options use that value.
MinimumAmplitude	Retrieves an estimate of the minimum amplitude value for the dataset. Normalization and other trace processing options use that value.
NumberOfTraces	Sets or retrieves the total number of traces in this dataset.
RMS	Retrieves the dataset RMS value. Some trace normalization options use this value. The RMS value can be an estimation of the actual RMS.
SampleRate	Gets and sets the sample rate for the seismic data. The sample rate is specified in the units returned by the SampleUnits property.
SamplesPerTrace	Gets or sets the number of samples in each trace.
SampleStart	Gets or sets the start sample for each trace.
SampleUnits	Gets or sets the units used along the sample axis.
StartValue	Sets or gets the start value (time or depth) for dataset.

3）3200 字节的 EDBCIC 文件头

下面这条语句可以得到整个 3200 字节的 EBCDIC 文本，并转换为 C#的字符串。

```
reader.EbcdicHeader.GetEbcdicHeaderAsString()
```

由于 3200 字节里面没有回车换行符，不方便阅读，可以用下面办法：

```
string[] header3600 = reader.EbcdicHeader.GetEbcdicHeaderAsStringArray();
for (int i = 0; i < 40; i++)
    txtFileInfo.Text += header3600[i] + Environment.NewLine;
```

4）cgDataFormat

Seismic 设计成能够读取多种格式的地震数据体，所以 reader 依赖于 cgDataFormat，我们常用的是 cgStandardSegyFormat，另外的 cgXMLSegyFormat 和 cgMemoryDataFormat 可能你以后会用到。

标准 SEGY 的 LineHeaderFormats 只有 2 个，一个是指 3200 字节的 EBCDIC 头，另一个是指 400 字节的二进制头，所以要自己读取 400 字节文件头信息时，就要用下面一行代码。

```
cgHeaderFormat binaryHeaderFormat =
                (cgHeaderFormat)reader.DataFormat.LineHeaderFormats[1];
```

这个格式定义中包含了许多 Field 的描述信息，实际上是与 SegyFormat.xml 相对应的：

```
<LineHeader Size="400" SampleRateField="SAMPLE INTERVAL"
       SamplesPerTraceField="SAMPLES PER TRACE" DataFormatField="DATA FORMAT">
    <Field Name="Line Number"           Format="UINT32"  Offset="4"/>
    <Field Name="Traces per Ensemble"    Format="UINT16"  Offset="12"/>
    <Field Name="SAMPLE INTERVAL"        Format="UINT16"  Offset="16"/>
    <Field Name="SAMPLES PER TRACE"      Format="UINT16"  Offset="20"/>
    <Field Name="DATA FORMAT"           Format="UINT16"  Offset="24"/>
    <Field Name="Traces per CDP"         Format="UINT16"  Offset="26"/>
    <Field Name="Measurement System"     Format="UINT16"  Offset="54"/>
    <Field Name="Segy Revision Number"   Format="UINT16"  Offset="300"/>
    <Field Name="Fixed Length Flag"      Format="UINT16"  Offset="302"/>
    <Field Name="Max Samples per Trace"  Format="UINT16"  Offset="304"/>
</LineHeader>
```

例如：偏移量 20 的位置记录着每道采样点个数，名称是"SAMPLES PER TRACE"，数据类型是 UINT16，这里把这些 Field 都用整数编了号，称为 Identifier，你需要用这个 Identifier 去得到相应的道头字。刚才的"SAMPLES PER TRACE"这个 Field 的 Identifier 是 104，所以你想得到每道的采样点个数，就要这样：

```
int samplePerTrace = metaData.HeaderValues[104];
```

如果不是标准 SEGY，你直接用 104 就可能出错。所以用元数据来得到"SAMPLES PER TRACE"这类常用的信息更方便。

5）道头的读取

用 TraceHeaderFormats 可以访问道头，Seismic.NET 设计得有些复杂，支持多个道头，但标准 SEGY 只有一个道头，所以永远访问 TraceHeaderFormats[0]即可。

```
cgHeaderFormat traceHeaderFormat =
            (cgHeaderFormat)reader.DataFormat.TraceHeaderFormats[0]; //只有第 0 个道头
```

一个地震数据体里有许多的道，用下面语句来获得指定的道的元信息：

```
// traceID 是从 0 开始的道号
cgTraceMetaData traceMeta = reader.GetTraceMetaData(traceID);
```

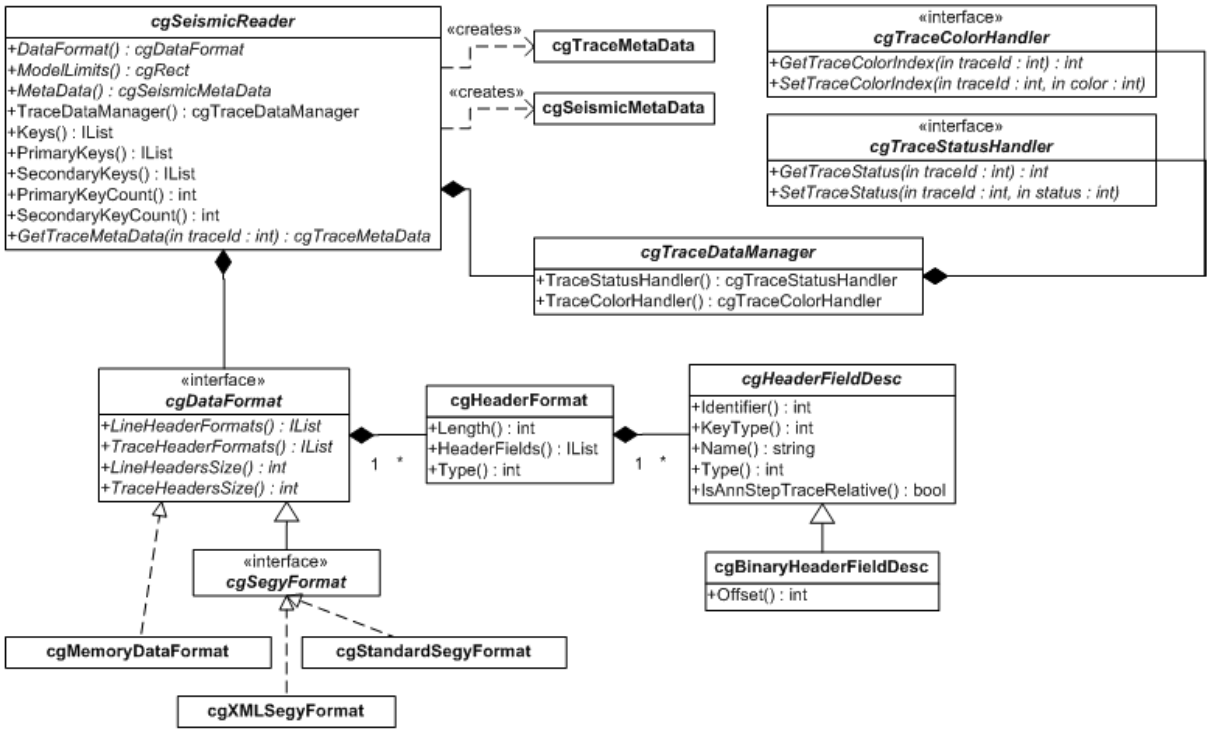
道头里面也有大量的 Field，与前面文件头的描述类似，也需要 Identifier 来访问。例如：标准 SEGY 的 240 道头字中，偏移量 20 的位置存放在 CDP 值，它的 Identifier 为 204，数据类型为 UINT，我们如果想读出这一道上的 CDP 号，就需要：

```
int cdp = traceMeta.DataValues[204];
```

最后再来看一下 MouseMove 事件中的代码，其中的 203、205、209、210 就是一些道头字的 Identifier。

```
string line = reader.GetTraceMetaData(traceIdx).GetField(203).ToString();
string cdp = reader.GetTraceMetaData(traceIdx).GetField(205).ToString();
string x = reader.GetTraceMetaData(traceIdx).GetField(209).ToString();
string y = reader.GetTraceMetaData(traceIdx).GetField(210).ToString();
```

6）整个 cgSeismicReader 的类图



15.4 试验

在 Seismic.NET 的 tutorials 中提供了一个 XMLSegyFormat 的例子，如果你的 SEGY 不太标准，你可以定义这个 xml，然后只用元数据就可以访问常用的道头了，而不用学习复杂难懂

的 LineHeaderFormats 和 TraceHeaderFormats。