如何:使用 PicturBox 实现类似淘宝网站图片的局部放大功能

选自：http://xuzhihong1987.blog.163.com/blog/static/267315872011822113131823/

概要：

本文将讲述如何使用 PictureBox 控件实现图片的局部放大浏览功能，效果类似淘宝网站的图片局部放大浏览，通过鼠标悬浮查看局部大图，同时扩展了鼠标滚轮放大缩小功能。本文将详细讲述实现该功能的主要思路，例子虽是在 Winform 的环境下实现（当时开发的系统用的是 winform），但是代码实现思路在其他环境（如 WPF）应该是通用的。

关键词：图片局部放大、PictureBox、图片细节展示

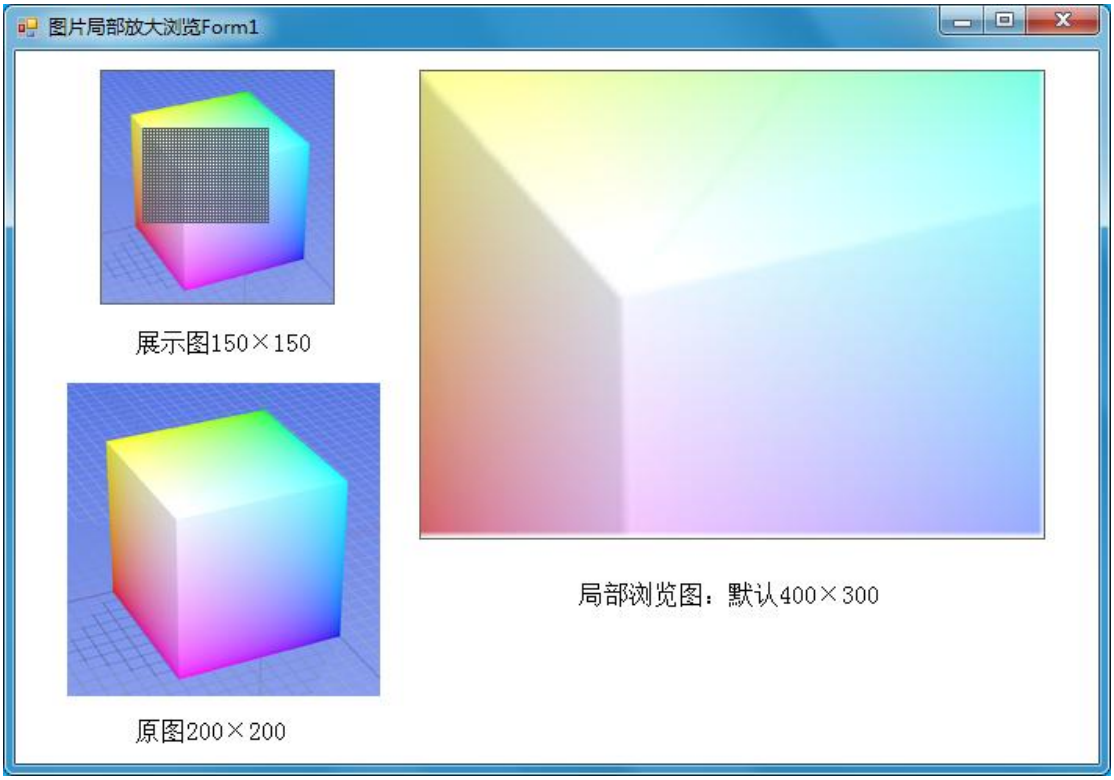
解决方案：

下面要实现的功能就是类似淘宝网站的图片局部放大功能，既然是山寨淘宝的功能，那么我们首先来看一下淘宝网站图片放大的效果图：

当然这个图片是在淘宝首页上随便选的一张（呵呵，不含任何宣传的成分）。不管这个功能淘宝官网是如何实现的，但是毋庸置疑，该功能非常实用（至少我个人还是感觉还是不错的），即用户友好度很高。如果能把该功能做到我们自己的系统或网站中，那岂不是挺好？主动学人之长，到哪都好用。可惜，我百度，google 了一下，竟然没人肯透露具体怎么做的，偶尔有人问到，但是回答似乎不尽人意，笔者想想也对，虽然很大一部分人知道怎么做或是已经做成功过了，但是没能把思想分享出来。那么就有我来抛砖引玉吧，期待更多的人参与讨论和指导。

言归正传，我们按老规矩，还是先看看我们自己实现的效果图吧，由于只是为了实现功能，布局什么的都没考虑，所以美观方面就不能和上面的图片效果进行比较了，大家暂时将

就一下。

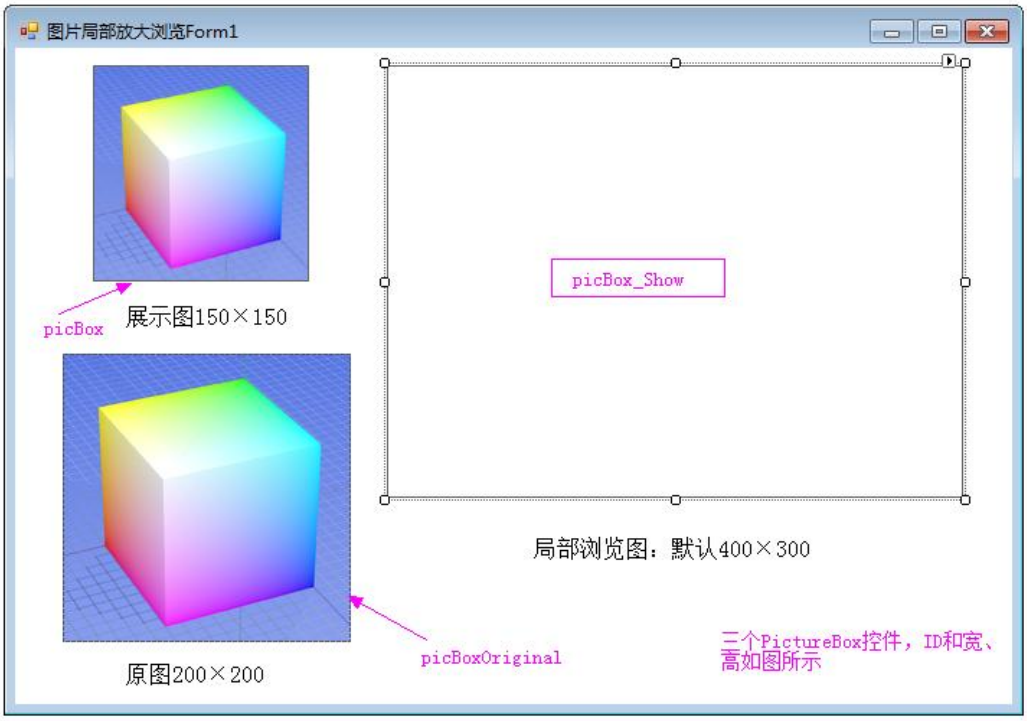


功能需求：

该功能的需求就是当鼠标悬浮在图片上，将该图片的固定大小（以鼠标点为中心的一个矩形标识区域），右边以大图的方式显示出来，同时鼠标移动时，矩形区域随鼠标而动，右边的浏览大图位置相应改变，便于用户查看图片细节。矩形标识区域和浏览大图都是是在鼠标悬浮时出现，鼠标离开后消失，而且矩形标识区域边框只能在图片中，不能离开图片。

实现思路：

第一步：布局



按照上面的布局方式，在 Form 中放入三个 PictureBox 控件，ID 分别为：picBox、picBox_Show、picBoxOriginal。该功能的实际应用上用到的只要两个就行，这里多加一个是为了对比用。

- picBox:展示图，用于固定图片的大小，这里设置为 150×150px
- picBox_Show:图片局部放大显示的区域，默认为 400×300，大小可根据鼠标滚轮进行缩放。
- picBoxOriginal：是实际图片的尺寸，在这里是为了对比效果。找的一张 200×200px 的原图。

将 picBox_Show 的 BorderStyle 的属性设置为 FixedSingle，即有边框，Visible 的属性设置为 false，即开始运行是不显示 picBox_Show。

将 picBox 的 SizeMode 属性设置为 Zoom。【重要】设置为等比例缩放，避免图片显示变形。

为 picBox 和 picBoxOriginal 选择一张图片（Image 属性），注：两个是同一张图片

第二步：鼠标事件

由于 picBoxOrginal 只是为了对比效果，仅仅是显示而已，我们不需要对其操作。

对 picBox 注册三个事件：鼠标移动 MouseMoveve、鼠标离开 MouseLeave、Paint 事件。只属性的事件中双击即可自动在.cs 文件中生成事件（当然你喜欢的话，后加代码注册也可以，笔者比较懒，喜欢双击的）。

注：你们自动生成的都应该是 picBox_Paint、picBox_MouseMove、picBox_MouseLeave 事件，因为一开始自动生成用的是默认的 ID（pictureBox1），后来为了正规点就换了个 ID，这里就没改了，当然这不影响我们的功能。

```
private void pictureBox1_Paint(object sender, PaintEventArgs e)
{
    //定位逻辑，详细后面实现
}
```

Paint 事件中处理逻辑：当鼠标移动在图片的某个位置时，我们需要绘个长方形区域，同时显示局部放大图片（picBox_Show）。当执行 picBox.Refresh()方法时将触发该事件。

```
private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
{
    picBox.Focus(); //否则滚轮事件无效
    isMove = true;
    movedPoint_X = e.X;
    movedPoint_Y = e.Y;
    picBox.Refresh();
}
```

在鼠标移动事件中，我们需要记录当前鼠标点的位置 ，有全局变量movedPoint_X, movedPoint_Y存储。

```
//鼠标移动后点的坐标
int movedPoint_X, movedPoint_Y;
```

同时我们需要设置一个鼠标移动状态isMove ，作为全局变量[bool isMove = false;]，标识是否需要重新绘图。

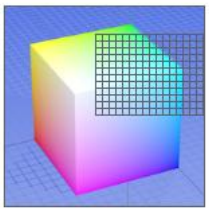
```
private void pictureBox1_MouseLeave(object sender, EventArgs e)
{
    picBox_Show.Visible = false;
    picBox.Refresh();

    picBox_Show.Width = 400;
    picBox_Show.Height = 300;
}
```

鼠标移开后，局部显示图片(picBox_Show)隐藏，picBox 绘制的长方形也要去掉，最简单的就是调用Refresh()方法了。

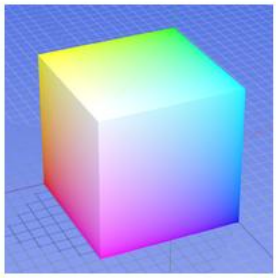
```
void picBox_Show_MouseWheel(object sender, MouseEventArgs e)
{
    double scale = 1;
    if (picBox_Show.Height > 0)
    {
        scale = (double)picBox_Show.Width / (double)picBox_Show.Height;
    }
    picBox_Show.Width += (int)(e.Delta * scale);
    picBox_Show.Height += e.Delta;
}
```

鼠标滑轮事件，当鼠标滑动时，picBox_Show 的大小可以改变。这个事件需要代码注册：picBox.MouseWheel += new MouseEventHandler(picBox_Show_MouseWheel);写在构造函数中。[当然取名为:picBox_MouseWheel似乎更合理，呵呵]

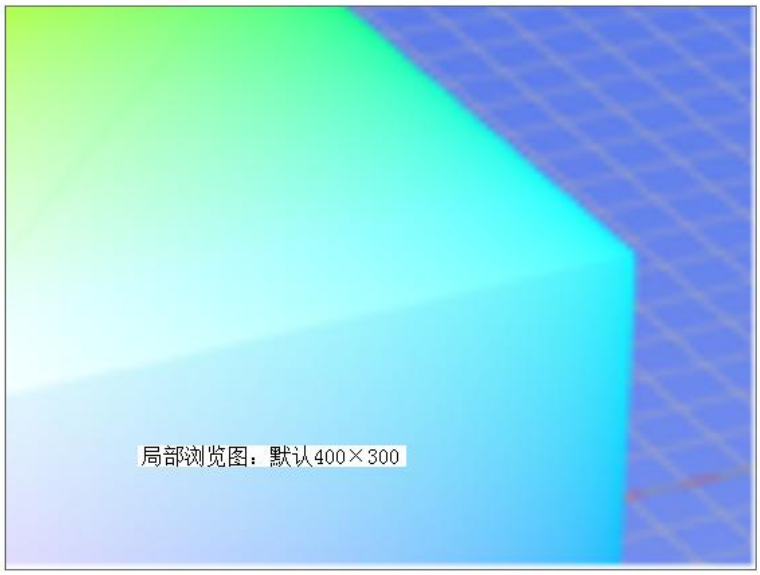
效果如下所示，picBox_Show 随鼠标滚轮等比例放大：



展示图150×150



原图200×200

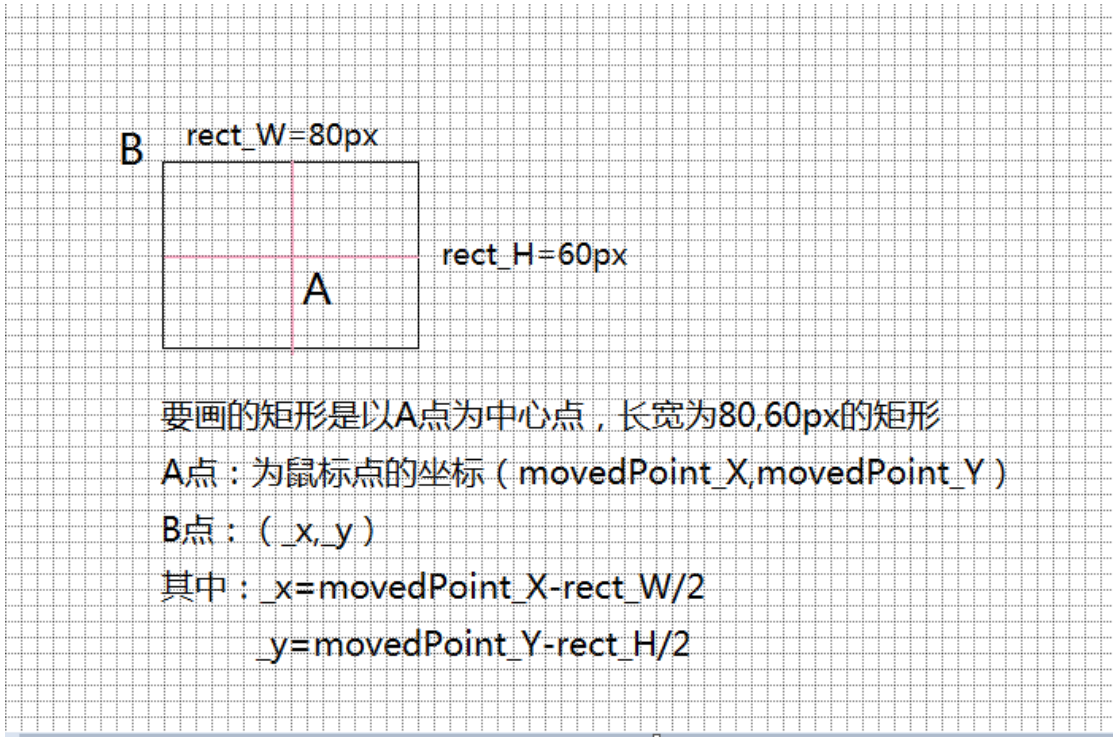


第三步：区域定位

这一步我们就是写 Paint 方法了，也就是这个功能的核心。需要做的功能就是画带网格的矩形，和显示矩形选择区域对于的大图。

- 画带网格的矩形

  我们选择先画矩形（DrawRectangle 方法），再填充网格的方式解决。为什么不直接使用更简单的阴影画笔画网格（FillRectangle 方法）呢?等一下我会讲到。



画矩形的原理如上图所示，A 点时刻记录鼠标点的位置，坐标为（movedPoint_X,movedPoint_Y）,在 MoveMove 事件中改变值。

```
movedPoint_X = e.X;
movedPoint_Y = e.Y;
```

知道鼠标点位置我们就开始画长方形了，直接使用 DrawRectangle 方法，改方法需要两个参数：画笔和长方形。

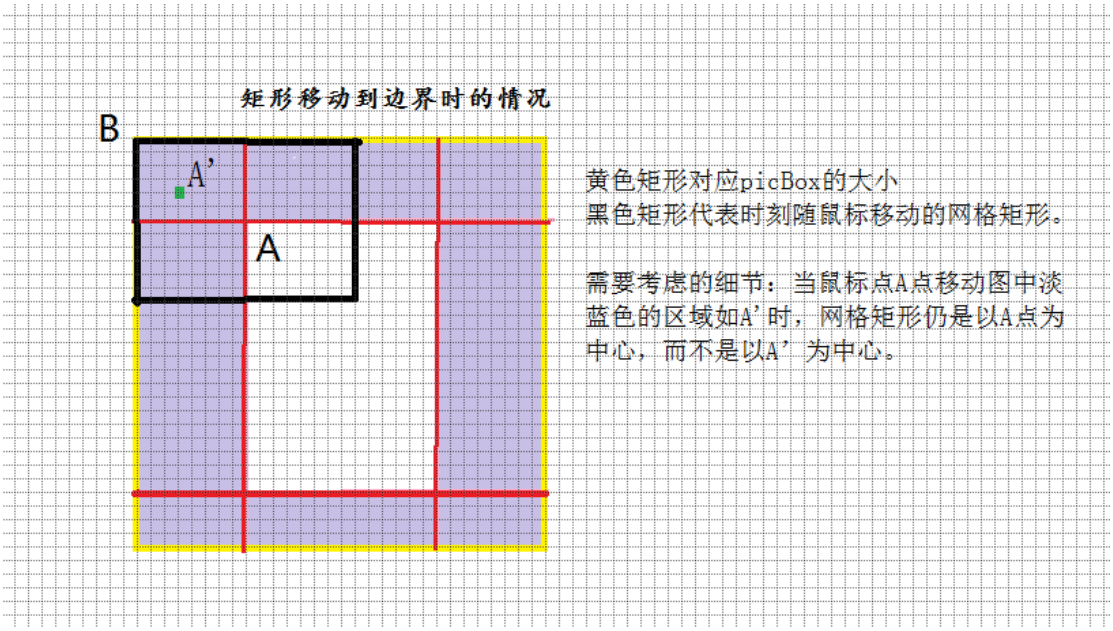
画笔全局定义为：

```
Pen pen = new Pen(Color.FromArgb(91, 98, 114)); //画笔颜色
```

长方形需要知道左上角的坐标 B 点（_x,_y），计算如下：

```
/*画长方形*/
int _x = movedPoint_X - rect_W/2;
int _y = movedPoint_Y - rect_H/2;
```

需要注意的是，就是边界问题，如下图所示：



边界问题解决如下：

```
_x = _x < 0 ? 0 : _x;
_y = _y < 0 ? 0 : _y;
_x = _x >= picBox.Width-rect_W ? picBox.Width-rect_W-3 : _x; //减3px的目的就是为了让长方形的边框不会刚好被picBox的边框挡住了
_y = _y >= picBox.Height-rect_H? picBox.Height-rect_H-3: _y;
Rectangle rect = new Rectangle(_x,_y, rect_W, rect_H);
g.DrawRectangle(pen, rect);//其中： Graphics g = e.Graphics;
```

长方形好了，那么我们就开始填充网格了，网格填充其实就是在矩形区域中画线平均分割成小方格。

预先定义一下网格的形式：

```
//网格边长：5px 一格
const int gridSize = 2;
//网格的行、列数
int rowGridCount = rect_H / gridSize;
int columnGridCount = rect_W / gridSize;
```

那么画网格直接循环即可，横竖画线，如下代码：

```
//*填充网格*/
int x1, x2, y1, y2;
x1 = x2 = _x;
y1 = y2 = _y;
```

```
x2 += rect_W;
for (int i = 1; i < rowGridCount; i++)
{
    y1 += gridSize;
    y2 += gridSize;
    g.DrawLine(pen, x1, y1, x2, y2);
}
x1 = x2 = _x;
y1 = y2 = _y;
y2 += rect_H;
for (int j = 1; j < columnGridCount; j++)
{
    x1 += gridSize;
    x2 += gridSize;
    g.DrawLine(pen, x1, y1, x2, y2);
}
```

- 显示矩形选择区域对于的大图

  该步重点就是根据等比例裁剪图片，计算缩放比例：

```
Bitmap bmp = (Bitmap)picBox.Image;
double rate_W = Convert.ToDouble(bmp.Width) / picBox.Width;
double rate_H = Convert.ToDouble(bmp.Height) / picBox.Height;
```

  bmp 得到的是实际的图片大小，由于我们前面设置了 picBox 的 SizeMode 属性为 Zoom，所以我们看到的 picBox 大小可能是经过了缩放的。所以不要错误地认为 rate_W=rate_H==1

  这样的话，我们使用位图的 Clone 方法，截取网格矩形对应原图的局部图形 bmp2:

```
Bitmap bmp2 = bmp.Clone(new Rectangle(Convert.ToInt32(rate_W*_x),
Convert.ToInt32(rate_H*_y), Convert.ToInt32(rate_W*rect_W), Convert.ToInt32(rate_H*rect_H)),
picBox.Image.PixelFormat);
```

最后个picBox_Show的Image属性赋值：

```
picBox_Show.Image = bmp2;
picBox_Show.SizeMode = PictureBoxSizeMode.Zoom;
picBox_Show.Visible = true;
```

当然没错赋值之前记得释放一下资源：

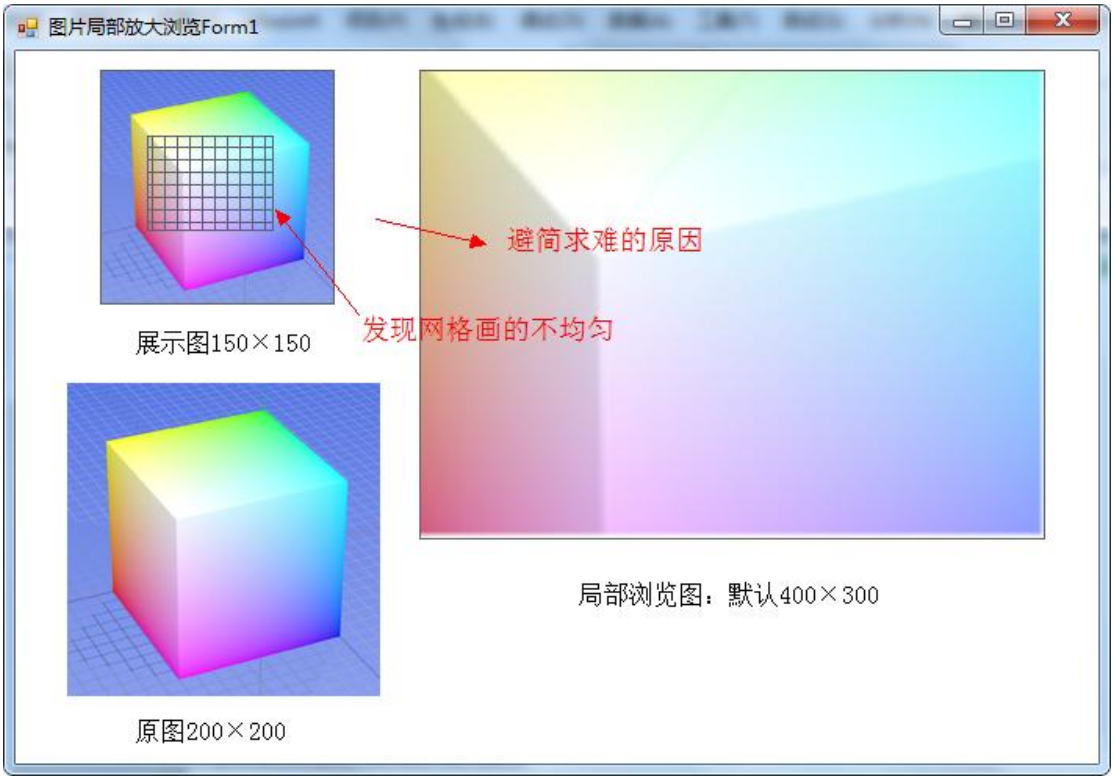
```
if (picBox_Show.Image != null)
{
    picBox_Show.Image.Dispose();
}
```

到这里就完成了我们所需要的功能，是不是感觉很简单？是的，我也这么认为，以后碰到没实现过的东西，仔细研究一下原理，那么就容易实现了，祝大家顺利成功。

画网格为什么避简求难？

最后简单解释一下：我们选择先画矩形（DrawRectangle 方法），再填充网格的方式解决。为什么不直接使用更简单的阴影画笔画网格（FillRectangle 方法）呢？

确实这样做效率低，而且实现逻辑还比较麻烦。
大家先看看下图就明白了：



附后台完整代码：

```
public Form1()
{
    InitializeComponent();
    picBox.MouseWheel += new MouseEventHandler(picBox_Show_MouseWheel);
}


void picBox_Show_MouseWheel(object sender, MouseEventArgs e)
{
    double scale = 1;
    if (picBox_Show.Height > 0)
    {
        scale = (double)picBox_Show.Width / (double)picBox_Show.Height;
    }
    picBox_Show.Width += (int)(e.Delta * scale);
    picBox_Show.Height += e.Delta;
}
```

```
        bool isMove = false;
        //鼠标移动后点的坐标
        int movedPoint_X, movedPoint_Y;
        //画笔颜色
        Pen pen = new Pen(Color.FromArgb(91, 98, 114));
        HatchBrush brush = new HatchBrush(HatchStyle.Cross, Color.FromArgb(91, 98,
114),Color.Empty); //使用阴影画笔画网格


        //选取区域的大小
        const int rect_W = 80;
        const int rect_H = 60;
        //网格边长：5px 一格
        const int gridSize = 2;
        //网格的行、列数
        int rowGridCount = rect_H / gridSize;
        int columnGridCount = rect_W / gridSize;


        private void pictureBox1_Paint(object sender, PaintEventArgs e)
        {

            if (isMove == true)
            {
                Graphics g = e.Graphics;
                /*画长方形*/
                int _x = movedPoint_X - rect_W/2;
                int _y = movedPoint_Y - rect_H/2;
                _x = _x < 0 ? 0 : _x;
                _y = _y < 0 ? 0 : _y;
                _x = _x >= picBox.Width-rect_W ? picBox.Width-rect_W-3 : _x; //减3px的目的就
是为了让长方形的边框不会刚好被picBox的边框挡住了
                _y = _y >= picBox.Height-rect_H? picBox.Height-rect_H-3: _y;
                Rectangle rect = new Rectangle(_x,_y, rect_W, rect_H);
                g.DrawRectangle(pen, rect);
               // g.FillRectangle(brush, rect);

                ///*填充网格*/
                int x1, x2, y1, y2;
                x1 = x2 = _x;
                y1 = y2 = _y;
                x2 += rect_W;
                for (int i = 1; i < rowGridCount; i++)
```

```
                {
                    y1 += gridSize;
                    y2 += gridSize;
                    g.DrawLine(pen, x1, y1, x2, y2);
                }
                x1 = x2 = _x;
                y1 = y2 = _y;
                y2 += rect_H;
                for (int j = 1; j < columnGridCount; j++)
                {
                    x1 += gridSize;
                    x2 += gridSize;
                    g.DrawLine(pen, x1, y1, x2, y2);
                }

                /*裁剪图片*/
                if (picBox_Show.Image != null)
                {
                    picBox_Show.Image.Dispose();
                }
                Bitmap bmp = (Bitmap)picBox.Image;
                //缩放比例
                double rate_W = Convert.ToDouble(bmp.Width) / picBox.Width;
                double rate_H = Convert.ToDouble(bmp.Height) / picBox.Height;

                Bitmap bmp2 = bmp.Clone(new Rectangle(Convert.ToInt32(rate_W*_x),
Convert.ToInt32(rate_H*_y), Convert.ToInt32(rate_W*rect_W), Convert.ToInt32(rate_H*rect_H)),
picBox.Image.PixelFormat);
                picBox_Show.Image = bmp2;
                picBox_Show.SizeMode = PictureBoxSizeMode.Zoom;
                picBox_Show.Visible = true;

                isMove = false;

            }

        }


        private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
        {
            picBox.Focus(); //否则滚轮事件无效
            isMove = true;
            movedPoint_X = e.X;
```

```
    movedPoint_Y = e.Y;
    picBox.Refresh();
}

private void pictureBox1_MouseLeave(object sender, EventArgs e)
{
    picBox_Show.Visible = false;
    picBox.Refresh();

    picBox_Show.Width = 400;
    picBox_Show.Height = 300;
}
```

Feitianxinhong 2011-09-22