

[articles](#) [Q&A](#) [forums](#) [lounge](#)

Search for articles, questions, tips



Image Processing using C#



Saleth Prakash, 5 Mar 2009



4.84 (216 votes)

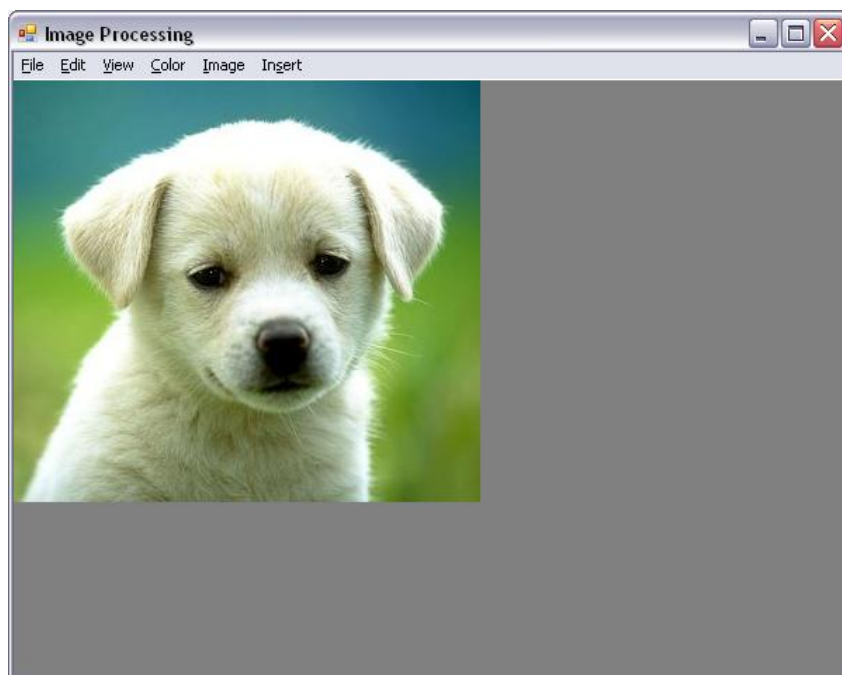
Rate this:

Image related operations are done using C#

[Download demo project - 18.01 KB](#)[Download source - 54.62 KB](#)

Introduction

This is my sixth article in C#. I got impressed with a similar article, so I tried this.



Overview

The purpose of the article is to be able to build a class that allows any C# programmer to perform image processing functionality. The reason we are doing it in C# is that it is very flexible for me. We can see that the code becomes somewhat more complex when we start moving pixels or changing values based on calculations that take into account all the pixel values.

Application

The application uses the basic Windows Forms application. I have handled the images with a separate class called **ImageHandler** in which all the image related operations are done including Saving, Graphics related operations. The Functionality includes getting image information, zooming, color filtering, brightening, contrasting, gamma filtering, grayscale filtering, invert filtering, resizing with full resolution, rotating and flipping, cropping and inserting text, any other image and some geometric shapes. Scrolling is achieved in the standard manner. The **Paint** method uses the **AutoScrollPosition** property to find out our scroll position, which is set by using the **AutoScrollMinSize** property.

1. Color Filter

Color filters are sometimes classified according to their type of spectral absorption: short-wavelength pass, long-wavelength pass or band-pass; diffuse or sharp-cutting; monochromatic or conversion. The short-wavelength pass transmits all wavelengths up to the specified one and then absorbs. The long-wavelength pass is the opposite. Every filter is a band-pass filter when considered generally.

It is very simple - it just adds or subtracts a value to each color. The most useful thing to do with this filter is to set two colors to -255 in order to strip them and see one color component of an image. For example, for red filter, keep the red component as it is and just subtract 255 from the green component and blue component.



Hide Shrink ▲ Copy Code

```

public void SetColorFilter(ColorFilterTypes colorFilterType)
{
    Bitmap temp = (Bitmap)_currentBitmap;
    Bitmap bmap = (Bitmap)temp.Clone();
    Color c;
    for (int i = 0; i < bmap.Width; i++)
    {
        for (int j = 0; j < bmap.Height; j++)
        {
            c = bmap.GetPixel(i, j);
            int nPixelR = 0;
            int nPixelG = 0;
            int nPixelB = 0;
            if (colorFilterType == ColorFilterTypes.Red)
            {
                nPixelR = c.R;
                nPixelG = c.G - 255;
                nPixelB = c.B - 255;
            }
            else if (colorFilterType == ColorFilterTypes.Green)
            {
                nPixelR = c.R - 255;
                nPixelG = c.G;
                nPixelB = c.B - 255;
            }
            else if (colorFilterType == ColorFilterTypes.Blue)
            {
                nPixelR = c.R - 255;
                nPixelG = c.G - 255;
                nPixelB = c.B;
            }
            nPixelR = Math.Max(nPixelR, 0);
            nPixelR = Math.Min(255, nPixelR);

            nPixelG = Math.Max(nPixelG, 0);
            nPixelG = Math.Min(255, nPixelG);

            nPixelB = Math.Max(nPixelB, 0);
            nPixelB = Math.Min(255, nPixelB);

            bmap.SetPixel(i, j, Color.FromArgb((byte)nPixelR,
                (byte)nPixelG, (byte)nPixelB));
        }
    }
    _currentBitmap = (Bitmap)bmap.Clone();
}

```

2. Gamma

Gamma filtering matters if you have any interest in displaying an image accurately on a computer screen. Gamma filtering controls the overall brightness of an image. Images which are not properly corrected can look either bleached out, or too dark. Trying to reproduce colors accurately also requires some knowledge of gamma. Varying the amount of gamma filtering changes not only the brightness, but also the ratios of red to green to blue. We produce a new color array and take the colors from that as the respective components in the image. The input values range between 0.2 to 5.



Hide Copy Code

```
public void SetGamma(double red, double green, double blue)
{
    Bitmap temp = (Bitmap)_currentBitmap;
    Bitmap bmap = (Bitmap)temp.Clone();
    Color c;
    byte[] redGamma = CreateGammaArray(red);
    byte[] greenGamma = CreateGammaArray(green);
    byte[] blueGamma = CreateGammaArray(blue);
    for (int i = 0; i < bmap.Width; i++)
    {
        for (int j = 0; j < bmap.Height; j++)
        {
            c = bmap.GetPixel(i, j);
            bmap.SetPixel(i, j, Color.FromArgb(redGamma[c.R],
                greenGamma[c.G], blueGamma[c.B]));
        }
    }
    _currentBitmap = (Bitmap)bmap.Clone();
}
```

Gamma array is created as:

Hide Copy Code

```
private byte[] CreateGammaArray(double color)
{
    byte[] gammaArray = new byte[256];
    for (int i = 0; i < 256; ++i)
    {
        gammaArray[i] = (byte)Math.Min(255,
            (int)((255.0 * Math.Pow(i / 255.0, 1.0 / color)) + 0.5));
    }
    return gammaArray;
}
```

3. Brightness

Brightening images are sometimes needed, it's a personal choice. Sometimes printing needs a lighter image than viewing. It is done just by adjusting the color components as per the user requirement. The input ranges between -255 and 255.



Hide Shrink ▲ Copy Code

```
public void SetBrightness(int brightness)
{
    Bitmap temp = (Bitmap)_currentBitmap;
    Bitmap bmap = (Bitmap)temp.Clone();
    if (brightness < -255) brightness = -255;
    if (brightness > 255) brightness = 255;
    Color c;
    for (int i = 0; i < bmap.Width; i++)
    {
        for (int j = 0; j < bmap.Height; j++)
        {
            c = bmap.GetPixel(i, j);
            int cR = c.R + brightness;
            int cG = c.G + brightness;
            int cB = c.B + brightness;
        }
    }
}
```

```

        if (cR < 0) cR = 1;
        if (cR > 255) cR = 255;

        if (cG < 0) cG = 1;
        if (cG > 255) cG = 255;

        if (cB < 0) cB = 1;
        if (cB > 255) cB = 255;

        bmap.SetPixel(i, j,
            Color.FromArgb((byte)cR, (byte)cG, (byte)cB));
    }
    _currentBitmap = (Bitmap)bmap.Clone();
}

```

4. Contrast

Contrasting of images is certainly a complex processing. Instead of just moving all the pixels in the particular direction, we must either increase or decrease the difference between the set of pixels. We accept values between -100 and 100, but we turn these into a double between the values of 0 and 4.


[Hide](#) [Shrink](#) [Copy Code](#)

```

public void SetContrast(double contrast)
{
    Bitmap temp = (Bitmap)_currentBitmap;
    Bitmap bmap = (Bitmap)temp.Clone();
    if (contrast < -100) contrast = -100;
    if (contrast > 100) contrast = 100;
    contrast = (100.0 + contrast) / 100.0;
    contrast *= contrast;
    Color c;
    for (int i = 0; i < bmap.Width; i++)
    {
        for (int j = 0; j < bmap.Height; j++)
        {
            c = bmap.GetPixel(i, j);
            double pR = c.R / 255.0;
            pR -= 0.5;
            pR *= contrast;
            pR += 0.5;
            pR *= 255;
            if (pR < 0) pR = 0;
            if (pR > 255) pR = 255;

            double pG = c.G / 255.0;
            pG -= 0.5;
            pG *= contrast;
            pG += 0.5;
            pG *= 255;
            if (pG < 0) pG = 0;
            if (pG > 255) pG = 255;

            double pB = c.B / 255.0;
            pB -= 0.5;
            pB *= contrast;
            pB += 0.5;
            pB *= 255;
            if (pB < 0) pB = 0;
            if (pB > 255) pB = 255;

            bmap.SetPixel(i, j,
                Color.FromArgb((byte)pR, (byte)pG, (byte)pB));
        }
    }
    _currentBitmap = (Bitmap)bmap.Clone();
}

```

5. Grayscale

Gray scale filtering is in reference to the color mode of a particular image. A gray scale image would, in layman's terms, be a black and white image, any other color would not be included in it.

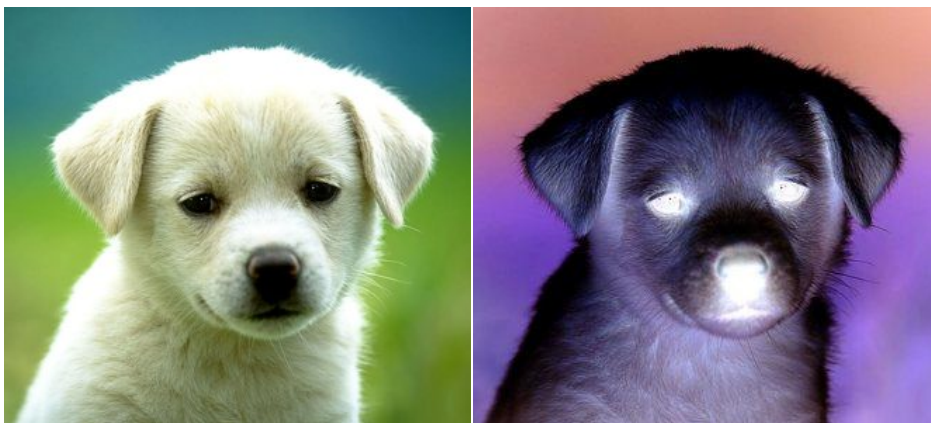
Basically, it's a black and white image, the colors in that image, if any will be converted to corresponding shade of gray (mid tones between black and white) thus, making each bit of the image still differentiable.


[Hide](#) [Copy Code](#)

```
public void SetGrayscale()
{
    Bitmap temp = (Bitmap)_currentBitmap;
    Bitmap bmap = (Bitmap)temp.Clone();
    Color c;
    for (int i = 0; i < bmap.Width; i++)
    {
        for (int j = 0; j < bmap.Height; j++)
        {
            c = bmap.GetPixel(i, j);
            byte gray = (byte)(.299 * c.R + .587 * c.G + .114 * c.B);
            bmap.SetPixel(i, j, Color.FromArgb(gray, gray, gray));
        }
    }
    _currentBitmap = (Bitmap)bmap.Clone();
}
```

6. Invert

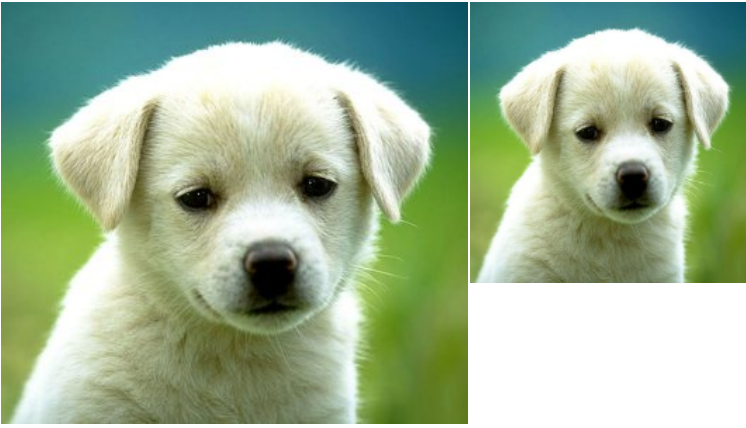
This is so simple that it doesn't even matter that the color components are out of order. it is just taking the opposite color of the current component, that is for example if the color component is 00 then the opposite we get is FF (255-0).


[Hide](#) [Copy Code](#)

```
public void SetInvert()
{
    Bitmap temp = (Bitmap)_currentBitmap;
    Bitmap bmap = (Bitmap)temp.Clone();
    Color c;
    for (int i = 0; i < bmap.Width; i++)
    {
        for (int j = 0; j < bmap.Height; j++)
        {
            c = bmap.GetPixel(i, j);
            bmap.SetPixel(i, j,
                Color.FromArgb(255 - c.R, 255 - c.G, 255 - c.B));
        }
    }
    _currentBitmap = (Bitmap)bmap.Clone();
}
```

7. Resize

This is resizing the width and height of the image without affecting any pixels of the image so that it does not affect the resolution of the image.


[Hide](#) [Shrink](#) [Copy Code](#)

```
public void Resize(int newWidth, int newHeight)
{
    if (newWidth != 0 && newHeight != 0)
    {
        Bitmap temp = (Bitmap)_currentBitmap;
        Bitmap bmap = new Bitmap(newWidth, newHeight, temp.PixelFormat);

        double nWidthFactor = (double)temp.Width / (double)newWidth;
        double nHeightFactor = (double)temp.Height / (double)newHeight;

        double fx, fy, nx, ny;
        int cx, cy, fr_x, fr_y;
        Color color1 = new Color();
        Color color2 = new Color();
        Color color3 = new Color();
        Color color4 = new Color();
        byte nRed, nGreen, nBlue;

        byte bp1, bp2;

        for (int x = 0; x < bmap.Width; ++x)
        {
            for (int y = 0; y < bmap.Height; ++y)
            {
                fr_x = (int)Math.Floor(x * nWidthFactor);
                fr_y = (int)Math.Floor(y * nHeightFactor);
                cx = fr_x + 1;
                if (cx >= temp.Width) cx = fr_x;
                cy = fr_y + 1;
                if (cy >= temp.Height) cy = fr_y;
                fx = x * nWidthFactor - fr_x;
                fy = y * nHeightFactor - fr_y;
                nx = 1.0 - fx;
                ny = 1.0 - fy;

                color1 = temp.GetPixel(fr_x, fr_y);
                color2 = temp.GetPixel(cx, fr_y);
                color3 = temp.GetPixel(fr_x, cy);
                color4 = temp.GetPixel(cx, cy);

                // Blue
                bp1 = (byte)(nx * color1.B + fx * color2.B);
                bp2 = (byte)(nx * color3.B + fx * color4.B);
                nBlue = (byte)(ny * (double)(bp1) + fy * (double)(bp2));

                // Green
                bp1 = (byte)(nx * color1.G + fx * color2.G);
                bp2 = (byte)(nx * color3.G + fx * color4.G);
                nGreen = (byte)(ny * (double)(bp1) + fy * (double)(bp2));

                // Red
                bp1 = (byte)(nx * color1.R + fx * color2.R);
                bp2 = (byte)(nx * color3.R + fx * color4.R);
                nRed = (byte)(ny * (double)(bp1) + fy * (double)(bp2));

                bmap.SetPixel(x, y, System.Drawing.Color.FromArgb
                (255, nRed, nGreen, nBlue));
            }
        }
        _currentBitmap = (Bitmap)bmap.Clone();
    }
}
```

8. Rotating and Flipping

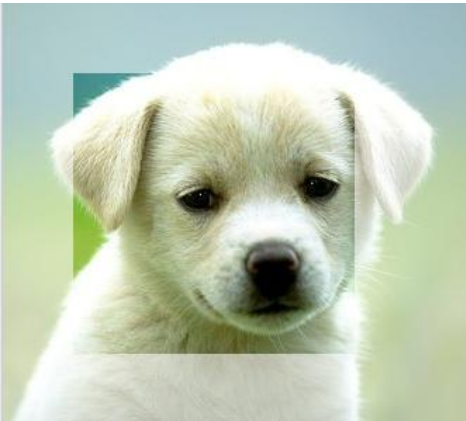
Rotating or flipping is also referred to as creating a mirror of a image. This is done in a very simple way by calling the **enums** available in C#.


[Hide](#) [Copy Code](#)

```
public void RotateFlip(RotateFlipType rotateFlipType)
{
    Bitmap temp = (Bitmap)_currentBitmap;
    Bitmap bmap = (Bitmap)temp.Clone();
    bmap.RotateFlip(rotateFlipType);
    _currentBitmap = (Bitmap)bmap.Clone();
}
```

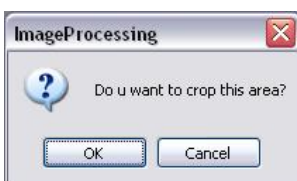
9. Crop

To cut out or trim unneeded portions of an image is crop. Here we perform this in 2 steps. First we mention the unneeded part as a semi transparent area. then as the users wish, we crop the image.


[Hide](#) [Copy Code](#)

```
public void DrawOutCropArea(int xPositon, int yPositon, int width, int height)
{
    _bitmapPrevCropArea = (Bitmap)_currentBitmap;
    Bitmap bmap = (Bitmap)_bitmapPrevCropArea.Clone();
    Graphics gr = Graphics.FromImage(bmap);
    Brush cBrush = new Pen(Color.FromArgb(150, Color.White)).Brush;
    Rectangle rect1 = new Rectangle(0, 0, _currentBitmap.Width, yPositon);
    Rectangle rect2 = new Rectangle(0, yPositon, xPositon, height);
    Rectangle rect3 = new Rectangle
    (0, (yPositon + height), _currentBitmap.Width, _currentBitmap.Height);
    Rectangle rect4 = new Rectangle((xPositon + width),
    yPositon, (_currentBitmap.Width - xPositon - width), height);
    gr.FillRectangle(cBrush, rect1);
    gr.FillRectangle(cBrush, rect2);
    gr.FillRectangle(cBrush, rect3);
    gr.FillRectangle(cBrush, rect4);
    _currentBitmap = (Bitmap)bmap.Clone();
}
```

Then accordingly, we get the option from the user to crop it or not.



If Ok, then the image is cropped as follows:

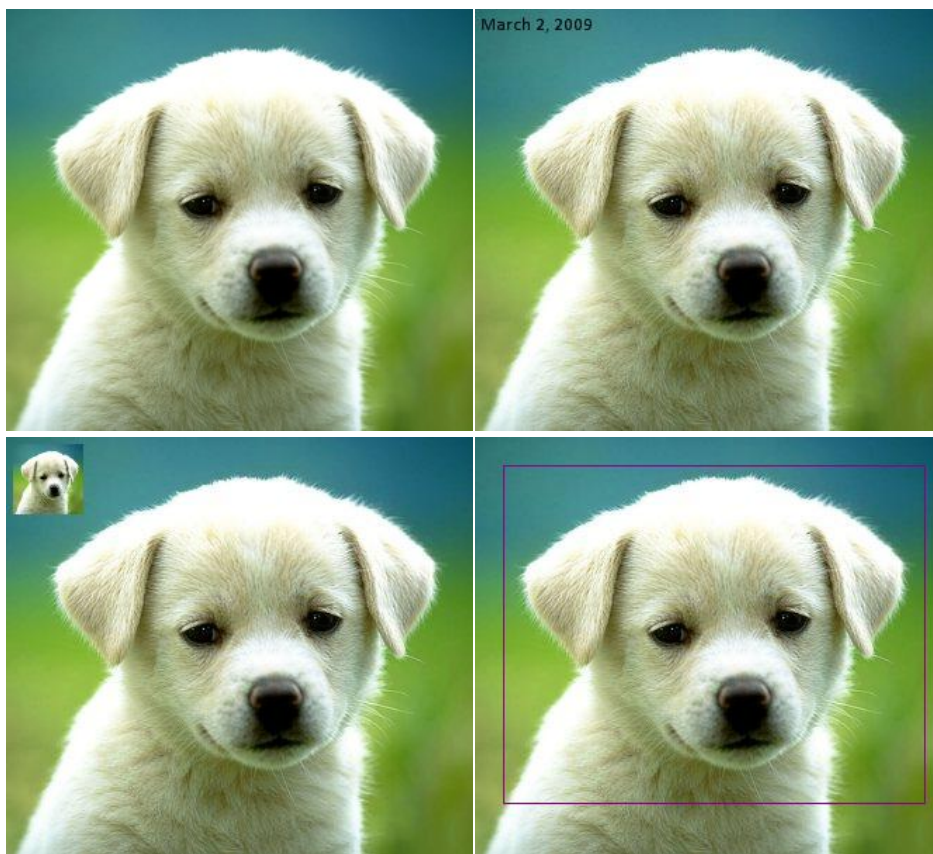


Hide Copy Code

```
public void Crop(int xPositon, int yPositon, int width, int height)
{
    Bitmap temp = (Bitmap)_currentBitmap;
    Bitmap bmap = (Bitmap)temp.Clone();
    if (xPositon + width > _currentBitmap.Width)
        width = _currentBitmap.Width - xPositon;
    if (yPositon + height > _currentBitmap.Height)
        height = _currentBitmap.Height - yPositon;
    Rectangle rect = new Rectangle(xPositon, yPositon, width, height);
    _currentBitmap = (Bitmap)bmap.Clone(rect, bmap.PixelFormat);
}
```

10. Inserting Text, Any Other Images and Shapes

This is just including any required things in the image. This is achieved by the **Graphics** object of the image.



Text

Hide Shrink ▲ Copy Code

```
public void InsertText(string text, int xPositon,
    int yPositon, string fontName, float fontSize,
    string fontStyle, string colorName1, string colorName2)
{
    Bitmap temp = (Bitmap)_currentBitmap;
    Bitmap bmap = (Bitmap)temp.Clone();
    Graphics gr = Graphics.FromImage(bmap);
    if (string.IsNullOrEmpty(fontName))
        fontName = "Times New Roman";
    if (fontSize.Equals(null))
        fontSize = 10.0F;
    Font font = new Font(fontName, fontSize);
```



```

        if (!string.IsNullOrEmpty(fontStyle))
        {
            FontStyle fStyle = FontStyle.Regular;
            switch (fontStyle.ToLower())
            {
                case "bold":
                    fStyle = FontStyle.Bold;
                    break;
                case "italic":
                    fStyle = FontStyle.Italic;
                    break;
                case "underline":
                    fStyle = FontStyle.Underline;
                    break;
                case "strikeout":
                    fStyle = FontStyle.Strikeout;
                    break;
            }
            font = new Font(fontName, fontSize, fStyle);
        }
        if (string.IsNullOrEmpty(colorName1))
            colorName1 = "Black";
        if (string.IsNullOrEmpty(colorName2))
            colorName2 = colorName1;
        Color color1 = Color.FromName(colorName1);
        Color color2 = Color.FromName(colorName2);
        int gw = (int)(text.Length * fontSize);
        gw = gw == 0 ? 10 : gw;
        LinearGradientBrush LGBrush =
        new LinearGradientBrush(new Rectangle(0, 0, gw, (int)fontSize), color1,
        color2, LinearGradientMode.Vertical);
        gr.DrawString(text, font, LGBrush, xPosition, yPosition);
        _currentBitmap = (Bitmap)bmap.Clone();
    }

```

Image

[Hide](#) [Copy Code](#)

```

public void InsertImage(string imagePath, int xPosition, int yPosition)
{
    Bitmap temp = (Bitmap)_currentBitmap;
    Bitmap bmap = (Bitmap)temp.Clone();
    Graphics gr = Graphics.FromImage(bmap);
    if (!string.IsNullOrEmpty(imagePath))
    {
        Rectangle rect = new Rectangle(xPosition, yPosition,
            i_bitmap.Width, i_bitmap.Height);
        gr.DrawImage(Bitmap.FromFile(imagePath), rect);
    }
    _currentBitmap = (Bitmap)bmap.Clone();
}

```

Shape

[Hide](#) [Shrink](#) [▲](#) [Copy Code](#)

```

public void InsertShape(string shapeType, int xPosition,
    int yPosition, int width, int height, string colorName)
{
    Bitmap temp = (Bitmap)_currentBitmap;
    Bitmap bmap = (Bitmap)temp.Clone();
    Graphics gr = Graphics.FromImage(bmap);
    if (string.IsNullOrEmpty(colorName))
        colorName = "Black";
    Pen pen = new Pen(Color.FromName(colorName));
    switch (shapeType.ToLower())
    {
        case "filledellipse":
            gr.FillEllipse(pen.Brush, xPosition,
                yPosition, width, height);
            break;
        case "filledrectangle":
            gr.FillRectangle(pen.Brush, xPosition,
                yPosition, width, height);
            break;
        case "ellipse":
            gr.DrawEllipse(pen, xPosition, yPosition, width, height);
            break;
        case "rectangle":
            gr.DrawRectangle(pen, xPosition, yPosition, width, height);
            break;
        default:
            gr.DrawRectangle(pen, xPosition, yPosition, width, height);
            break;
    }
    _currentBitmap = (Bitmap)bmap.Clone();
}

```

Points of Interest

I have also included Undo Options, Clear Image, Image Information which are quiet simple.

Conclusion

Thanks for viewing this article. I expect feedback from you. You expect more from me.

License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOL\)](#)

Share

[EMAIL](#)[TWITTER](#)

About the Author



Saleth Prakash

India

No Biography provided

You may also be interested in...

[Designing For DevOps](#)[A Solution Blueprint for DevOps](#)[Image Processing Using GDI+](#)[SAPrefs - Netscape-like Preferences Dialog](#)[Image Processing Lab in C#](#)[Generate and add keyword variations using AdWords API](#)

Comments and Discussions

You must [Sign In](#) to use this message board.

Search Comments

Go

First Prev Next

My vote of 5

Karthik Bangalore 22-Mar-17 2:22

Cropper replace the cropped image area with some color in vb.net

Member 11982849 19-Apr-16 21:31

My vote of 5

Marsowl Omar 19-Sep-15 9:07

My vote of 5

Tirujit 12-Sep-15 17:37

My vote of 5

DrABELL 30-Aug-15 9:47





























Nice work

pencilaz 2-Aug-15 19:59

nice work

gdshukla 19-Jul-15 0:59

Is there any algorithm or any way we could get the picture background color so that we can set the same color/theme on the page to reflect the

image 	Sreekanth Mothukuru 15-Jul-15 2:38
Unable to load PNG files 	thisisnotted 19-Jun-15 9:09
My vote of 5 	88388132 13-May-15 3:17
Could you give me some code explanation about "How the functions work" please? 	Mohammad_Daker 22-Oct-14 8:34
This Article of image processing is useful to me because I'm also planning to design image processing project using c# 	Takkies 22-Jul-14 23:40
PHOTO COMPARISON 	Member 10763061 27-Apr-14 18:44
Performance 	Nicke Da Silva Manarin 16-Feb-14 11:09
Saving the image requires some work 	eravindran 7-Jan-14 1:49
drop out color 	Member 7865626 18-Dec-13 19:56
crop white space and save 	Anand1988 11-Nov-13 0:23
Great Work! 	LukeZhang213 16-Oct-13 7:49
Generic Image editor. 	Member 10330080 14-Oct-13 1:21
great 	Member 10220464 23-Aug-13 7:35
after clearing the image.. 	nikheel bharat 22-Jul-13 11:17
Re: after clearing the image.. 	karthikin 26-Mar-14 20:52
Good Work..! 	GabrielCF 11-Jul-13 9:17
My vote of 5 	AJMAL SHAHZAD 14-May-13 0:50
Nice Explained 	AJMAL SHAHZAD 14-May-13 0:50
My vote of 5 	computergurubd 2-May-13 2:56
My vote of 5 	ridoy 30-Apr-13 21:00
It's very nice article 5+ :) 	Aarti Meswania 3-Apr-13 0:09
About this artical 	mekalasrinivas 26-Mar-13 3:01
Plz help 	Member 9919566 17-Mar-13 21:26
Voted 5 	Tharinda_tpw 7-Mar-13 3:18
nice job 	aanju05 4-Mar-13 2:13
Question 	Kiroloss 4-Feb-13 23:54
comment 	umang_soni01 28-Dec-12 3:39
Thanks 	one_database 5-Oct-12 2:39
My vote of 5 	

2017/7/31Image Processing using C# - CodeProject

Tamok 11-Aug-12 14:22
How to find out the Co-ordinates of any dot or line in image ✨ nannaajay 31-Jul-12 1:12
Good One ✨ Priyatosh Nath 20-May-12 19:48
My vote of 5 ✨ S.K.Sathish kumar 16-Apr-12 21:50
my vote ✨ chukong 2-Apr-12 3:33
My vote of 5 ✨ manoj kumar choubey 17-Feb-12 1:21
My vote of 5 ✨ marcos_zanella 16-Jan-12 15:55
My vote of 5 ✨ T. Abdul Rahman 15-Nov-11 23:02
My vote of 5 ✨ mpino 25-Jul-11 21:35
My vote of 5 ✨ witmann 22-Jul-11 9:30
Very Very Good! ✨ zzzserr 10-Jul-11 22:20
Very Nice Article ✨ projectzombie 25-Apr-11 9:29
My vote of 5 ✨ nddvn2008 29-Mar-11 18:07
My vote of 5 ✨ Avinash T P 20-Mar-11 22:07
How to do it? ✨ Mindyt 26-Feb-11 1:04
Refresh12Next »