**EE 356**

**Notes on LockBits**

The following four examples how the LockBits function can be used to make the individual pixels of a bitmap image available for manipulation. Each example also has a link at the top showing its source.

http://csharpexamples.com/fast-image-processing-c/

private void ProcessUsingLockbitsAndUnsafe(Bitmap processedBitmap)

{

    unsafe

    {

        BitmapData bitmapData = processedBitmap.LockBits(new Rectangle(0, 0, processedBitmap.Width, processedBitmap.Height),

ImageLockMode.ReadWrite, processedBitmap.PixelFormat);

        int bytesPerPixel = System.Drawing.Bitmap.GetPixelFormatSize(processedBitmap.PixelFormat) / 8;

        int heightInPixels = bitmapData.Height;

        int widthInBytes = bitmapData.Width \* bytesPerPixel;

        byte\* ptrFirstPixel = (byte\*)bitmapData.Scan0;

        for (int y = 0; y < heightInPixels; y++)

        {

            byte\* currentLine = ptrFirstPixel + (y \* bitmapData.Stride);

            for (int x = 0; x < widthInBytes; x = x + bytesPerPixel)

            {

                int oldBlue = currentLine[x];

                int oldGreen = currentLine[x + 1];

                int oldRed = currentLine[x + 2];

                // calculate new pixel value

                currentLine[x] = (byte)oldBlue;

                currentLine[x + 1] = (byte)oldGreen;

                currentLine[x + 2] = (byte)oldRed;

            }

        }

        processedBitmap.UnlockBits(bitmapData);

    }

}

http://stackoverflow.com/questions/1563038/fast-work-with-bitmaps-in-c-sharp

/\*Note unsafe keyword\*/

public unsafe Image ThresholdUA(float thresh)

{

Bitmap b = new Bitmap(\_image);//note this has several overloads, including a path to an image

BitmapData bData = b.LockBits(new Rectangle(0, 0, \_image.Width, \_image.Height), ImageLockMode.ReadWrite, b.PixelFormat);

byte bitsPerPixel = GetBitsPerPixel(bData.PixelFormat);

/\*This time we convert the IntPtr to a ptr\*/

byte\* scan0 = (byte\*)bData.Scan0.ToPointer();

for (int i = 0; i < bData.Height; ++i)

{

for (int j = 0; j < bData.Width; ++j)

{

byte\* data = scan0 + i \* bData.Stride + j \* bitsPerPixel / 8;

//data is a pointer to the first byte of the 3-byte color data

}

}

b.UnlockBits(bData);

return b;

}

/\*No unsafe keyword!\*/

public Image ThresholdMA(float thresh)

{

Bitmap b = new Bitmap(\_image);

BitmapData bData = b.LockBits(new Rectangle(0, 0, \_image.Width, \_image.Height), ImageLockMode.ReadWrite, b.PixelFormat);

/\* GetBitsPerPixel just does a switch on the PixelFormat and returns the number \*/

byte bitsPerPixel = GetBitsPerPixel(bData.PixelFormat);

/\*the size of the image in bytes \*/

int size = bData.Stride \* bData.Height;

/\*Allocate buffer for image\*/

byte[] data = new byte[size];

/\*This overload copies data of /size/ into /data/ from location specified (/Scan0/)\*/

System.Runtime.InteropServices.Marshal.Copy(bData.Scan0, data, 0, size);

for (int i = 0; i < size; i += bitsPerPixel / 8 )

{

double magnitude = 1/3d\*(data[i] +data[i + 1] +data[i + 2]);

//data[i] is the first of 3 bytes of color

}

/\* This override copies the data back into the location specified \*/

System.Runtime.InteropServices.Marshal.Copy(data, 0, bData.Scan0, data.Length);

b.UnlockBits(bData);

return b;

}

https://msdn.microsoft.com/en-us/library/5ey6h79d(v=vs.110).aspx

private void LockUnlockBitsExample(PaintEventArgs e)

{

// Create a new bitmap.

Bitmap bmp = new Bitmap("c:\\fakePhoto.jpg");

// Lock the bitmap's bits.

Rectangle rect = new Rectangle(0, 0, bmp.Width, bmp.Height);

System.Drawing.Imaging.BitmapData bmpData =

bmp.LockBits(rect, System.Drawing.Imaging.ImageLockMode.ReadWrite,

bmp.PixelFormat);

// Get the address of the first line.

IntPtr ptr = bmpData.Scan0;

// Declare an array to hold the bytes of the bitmap.

int bytes = Math.Abs(bmpData.Stride) \* bmp.Height;

��� byte[] rgbValues = new byte[bytes];

// Copy the RGB values into the array.

System.Runtime.InteropServices.Marshal.Copy(ptr, rgbValues, 0, bytes);

// Set every third value to 255. A 24bpp bitmap will look red.

for (int counter = 2; counter < rgbValues.Length; counter += 3)

rgbValues[counter] = 255;

// Copy the RGB values back to the bitmap

System.Runtime.InteropServices.Marshal.Copy(rgbValues, 0, ptr, bytes);

// Unlock the bits.

bmp.UnlockBits(bmpData);

// Draw the modified image.

e.Graphics.DrawImage(bmp, 0, 150);

}