IDEATE WEB HOSTING

SPECIALIZING IN Visual FoxPro AND WEBCONNECT HOSTING

- Home
- Static & ASP Hosting
- WebConnect Hosting
- AFP Hosting
- Data Center
- Ideate Consulting
- About Us
- White Papers
- Michael's Blog
- Contact Us

Resizing Images for Maximum Quality

The Problem:

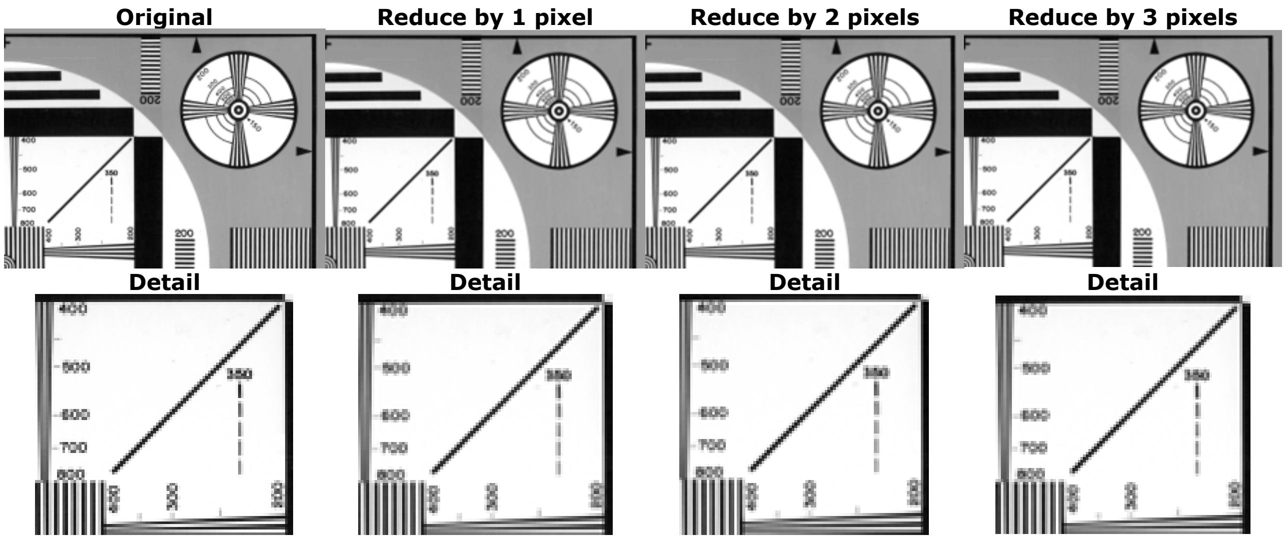
Resizing images ad-hoc creates severe image quality problems.

Our Homeowner Association Web Site, CondoConduit, allows associations to upload their own images to their personalized web site. The images can come from any source and can start at any size, but for display purposes must be resized to no more than 300 pixels on a side.

We originally used a simple algorythm which calculates the new width and height from the original size so that the proportions were unchanged and the longest side (width or height) became 300 pixels. This worked mathematically, but we found that the resulting image quality varied from great to downright embarrasing:

The Analysis:

I often ended up resizing the images manually. My process involved selective cropping and padding of the image so that the size of the longest edge was a multiple of 300 (the largest image edge size in my application). I then resized the image and uploaded the result directly to the web site so that my web application would not be forced to resize the image again.



The Theory:

The theory used was the nature of the GDI+ Resizing algorythm.

Digital images are made up of pixels. Zoom in close enough and you'll see them:

When resizing an image, GDI+ uses pixel avreaging to calculate the color of the resulting pixels. If GDI+ is combining 3 pixels into one to resize it down, it will simply take the mathematical average pixel color of the 3 original pixels to determine the color of the single pixel in the resized image. All that works great - but what happens when it's combining 2 1/2 pixels into one, or 3.14159 pixels into one?

Well, that forces the resizing algorithm to muddy up the color calculation for the final pixel color by averaging in the colors of adjoining pixels - resulting in what I will call the 'microblending' of resulting adjoining pixels, and 'fuzzing up' the result.

The Solution:

I'm OK at math, but no whiz. I asked around the forums and googled about for awhile, hoping someone had come up with a mathematically elegant way to determine the largest common divisor so that I could simply divide the image length and width by a whole number - avoiding the 'microblending' problem. I was expecting something using primes or something cutting edge.

Well, no luck. It looks like I would have to come up with something myself. Rather than using primes or the fibonocci series or the golden section, I'm using a brute force method:

```
************************************************************************
*** Function: IntegerResize
*** Assume:
*** Created: 03/28/2006
*** Revised:
*** Copyright: (c) 2006, Ideate, LLC
************************************************************************
FUNCTION IntegerResize()

LPARAMETERS pcDiskFileName, pnMaxHoriz, pnMaxVert

#IF .F.
   LOCAL Request as wwRequest, Response as wwResponse
#ENDIF

pnHoriz = 0
pnVert = 0
pnResolution = 0
pnNewHoriz = 0
pnNewVert = 0
pnClosestCrop = 1000
```

```
pnCropHoriz = 0
pnCropVert = 0
plCropFirst = .T.
pnCompression = 0

IF GetImageInfo(lcDiskFileName, @pnHoriz, @pnVert, @pnResolution)
   IF pnHoriz > pnMaxHoriz OR pnVert > pnMaxVert
      pcTempFileName = ADDBS(JUSTPATH(pcDiskFileName)) + "Delete_Me." + JustExt(pcDiskFileName)
      IF FILE(pcTempFileName)
         ERASE (pcTempFileName)
      ENDIF
      RENAME &pcDiskFileName TO &pcTempFileName
      IF pnMaxHoriz/pnHoriz > pnMaxVert/pnVert
         pcBiggerSide = [Vert]
      ELSE
         pcBiggerSide = [Horiz]
      ENDIF
      FOR i = 0 TO 50
         pnDenominator = pnMax&pcBiggerSide - i
         IF MOD(pnHoriz,pnDenominator) = 0 AND MOD(pnVert,pnDenominator) = 0
            * We have a hit
            pnNewHoriz = pnHoriz/(pn&pcBiggerSide/pnDenominator)
            pnNewVert = pnVert/(pn&pcBiggerSide/pnDenominator)
            plCropFirst = .F.
            EXIT
         ELSE
            * keep track of the closest match for cropping or padding
            IF MOD(pnHoriz,pnDenominator) + MOD(pnVert,pnDenominator) < pnClosestCrop
               pnClosestCrop = MOD(pnHoriz,pnDenominator) + MOD(pnVert,pnDenominator)
               pnNewHoriz = INT(pnHoriz/(pn&pcBiggerSide/pnDenominator))
               pnNewVert = INT(pnVert/(pn&pcBiggerSide/pnDenominator))
               pnCropHoriz = MAX(INT(pnHoriz/pnNewHoriz - pnMaxHoriz),0)
               pnCropVert = MAX(INT(pnVert/pnNewVert - pnMaxVert),0)
            ENDIF
         ENDIF
      NEXT
      IF plCropFirst && we didn't find a value
         * Crop and set variables for resizing
         *************************************
         ** TO DO!
         *************************************
         * PAD images for negative crop values
         *************************************
         pnLeft = INT(pnCropHoriz/2)
         pnTop = INT(pnCropVert/2)
         ReadImage(pcTempFileName,pcDiskFileName,pnLeft,pnTop,pnHoriz-pnCropHoriz,pnVert-pnCropVert)
         ERASE (pcTempFileName)
         RENAME &pcDiskFileName TO &pcTempFileName
      ENDIF
      * CreateThumbnail(pcTempFileName,pcDiskFileName,pnNewHoriz,pnNewVert)
      ResizeImage(pcTempFileName, pcDiskFileName, pnNewHoriz, pnNewVert, pnCompression)
      ERASE (pcTempFileName)
   ENDIF
ENDIF

ENDFUNC && IntegerResize
```

A few issues of interest:

- I am limiting the automated analysis to 50 pixels. That is, I don't want the final image size to be less than 50 pixels smaller than my defined maximum image size. This was somewhat arbitrary.
- If we don't find a 'common denominator' within 50 pixels of the final image size, we are keeping track of the 'closest match' and are then using the West-Wind WebConnect ReadImage() function to effectively crop away the right and/or bottom edge of the image so that the whole-number resize can be performed on the cropped image.
- The WestWind wwClientTools ResizeImage() and ReadImage() functions used here wrap GDI+ calls. GDI+ resizing Functions can be substituted with little trouble using the _gdiplus.vcx in the FFC subdirectory of VFP. In my brief search, I have not found a free solution for cropping images.

The Results:

Now my customers can upload their images to their CondoConduit site and get the best possible image quality, and I no longer have to manually resize and crop their uploaded images.