"Level II" Enhancement of PreCyber eBook Conversions

Don Lancaster
Synergetics, Box 809, Thatcher, AZ 85552
copyright c2011 pub 1/11 as GuruGram #117
http://www.tinaja.com
don@tinaja.com
(928) 428-4073

We looked at much that is involved in **remastering** a classic precyber book for web **eBook** distribution back in **GuruGram #115**. And details on how to improve any worn or scuffed covers **here** in **GuruGram #116**.

Adobe's **Acrobat X** does an incredible job of converting scanned text into viable PDF **eBook** files. But it does have some obvious and glaring shortcomings. But, by combining Acrobat X with my **Gonzo Utilities**, a "second pass" can be made at a generated PDF **ebook**. One that potentially has these compelling advantages...

- **Files can end up as much as 6:1 shorter.**
- **Text appearance is considerably better.**
- **Searchability is preserved during rework.**
- **Reduced images now cleanly magnifiable.**
- **Cleaner files have fewer prepress problems.**
- **Color and other value-added easily provided.**
- **Fully and automatically self-tracking web linkable.**
- **Images can be greatly improved, especially halftones.**

But with these limitations...

- **May be inappropriate with historic, legal or IP restrictions.**
- **Considerable time per page demands profitable project.**
- **Proudly non-WYSIWYG bare metal programming needed.**

The demo speaks for itself...

RUN LEVEL II DEMO

COMPARE LEVEL I ORIGINAL

VIEW LEVEL II SOURCECODE

The Level II Demo begins on page nine of the comparable Level I original. The new file sizes are about 8K per page.

The OCR in Acrobat X basically generates text and images. Each may be optimized separately…

Improving the Text

The usual text in a ClearScan generated .PDF file consists of bunches of custom generated font glyphs that were based on an average of the original bitmap characters. If done a page at a time, the number of embedded custom fonts in an **eBook** can range from several hundred to several thousand. Most fonts will remain slightly irregular and tend to favor rounded, rather than pointed serifs.

These fonts have their own coding scheme. Some elaborate **unicode conversion object tables** are needed if full text searchability is to be provided. **Such full text searchability is easily lost on any PDF-->PostScript-->PDF rework**.

Thus…

It is usually a good idea to replace all custom font glyphs with a standard unicoded font.

Benefits include file size, text appearance, and retaining full text searchability.

There is a procedure in **Acrobat X** to replace the ClearScan fonts with a standard font, but the process can be awkward and ungainly. Especially for page after page of fill justification. On the other hand, our **Gonzo Utilities** easily handle extremely fancy text **progressive microjustifications.**

GETTING FROM CLEARSCAN TO GONZO

1. Load the .pdf file into Acrobat X.

2. Do a Save As ---> More Options --> Text (plain)

3. Transfer the page textfile into a gonzo (...) cf

A number of variables give you Gonzo fill justification flexibility. These are covered in **this tutorial**. The two numbers preceding the **(...) cf** set the horizontal and vertical position. A predefined **yinc** sets the normal line spacing, while **txtwide** adjusts the fill justification width. Your definition of **/font0** through **/font:** lets you mix and match fonts.

Other commands provide for paragraph indents, half line ledding, emphasis, tabbing, hanging punctuation, last line stretch, and similar sophisticated features. Many use examples can be found in the **GuruGram** .PSL sourcecode files.

Note that **very little text rekeying is required**. Once your gonzo utilities .PSL text file is complete and edited, it is sent back to the Distiller in **Acrobat X** for .PDF reconversion. Per **these details**.

In our example, my favorite **Stone** ended up an amazingly close match to the ClearScan glyphs for the regular text, while the stock **MyriadPro-Bold** multiple master did a nice job on the larger point sizes. **Some editing may be needed if the ClearScan process got confused**. Or was unable to convert certain glyphs.

For instance, one recurring theme was **1/O** rather than **I/O**. Hyphenation can also bury hidden characters that may need removed. And some in-figure text can be easily confused with the main story. Such editing is fast and simple, but demands careful attention to detail.

Providing Text Value Added

Web delivery provides more benefits than any printed book could ever hope to provide. It is thus reasonable to try and "improve" any scanned text when it is legal and sensible to do so. At one time, it was outrageously expensive to add color to a printed book. But web color is essentially free. So, one simple add-on would be **adding color tints when and where appropriate.**

The real value-added biggie, of course, is URL linking. Gonzo provides for elegant **auto-tracking linking**. This is done in the text by adding a marker and filename trailer. Combined with a simple dictionary of filename to URL links. These links will automatically move around on the page as needed for any later editing.

Some more minor modifications can make your text more attractive and more readable. Such as adding half ledding between paragraphs. Or replacing italic emphasis with bolded color. I despise hyphenation, so adjusting minor words in your text to eliminate end-of-line hyphens can be a challenging exercise.

Improving Images

What is **not** text in a ClearScan result is usually in the form of a group of images. Any very small images are likely to be scanner speckle. These can be removed and ignored once you identify them. Somewhat larger images are sometimes text that ClearScan was unable to identify. These can get repaired in your text file editing. Alternately, you can go back to the original bitmap you scanned in Paint and then adjust these enough to become recognizable text.

"Real" images of stuff you actually want are likely to be larger. Several obvious improvement routes are (1) Leave them alone; (2) Retouch them as bitmaps; (3) Retouch them as bitmaps and improve the lettering with the **Bitmap Typewriter**;

(5) Retouch them as bitmaps and improve the lettering with **Custom Glyphs**; (5) Rework any halftones into grayscale or RGB; (6) Colorize black and white where appropriate; or **(7) Redo entirely as PostScript vector art**.

Choice (7) will give you by far the smallest file size. Plus the best appearance and maximum searchability. But it might end up quite time intensive for such things as electronic schematics. Other tools in the **Gonzo Utilities** greatly ease building top quality schematics or line art.

One of the most obvious image problems in our Level I example was the gray text boxes. ClearScan did these as halftones that did not look all that great to start out with and became unacceptable if file sizes were reduced. Worse, all of their text remained unrecognized and unsearchable.

As the Level II demo shows us, these were reworked as aqua background boxes with full text searchability. A minor amount of rekeying was involved here. Some similarly reworked text highlighting boxes appear in orange and magenta. An example of a total Gonzo rework of another gray halftone was the **Address Space** figure on page 17.

Here's how to go about…

EXPORTING IMAGES FROM ACROBAT

1. Raise the resolution by Edit --> Preferences --> Convert From PDF --> PNG --> Resolution to at least 600 pixels per inch.

2. To save an entire page as an image, do Save As --> Image --> PNG.

3. To save one object as an image, Use the selection tool to highlight the image in blue, then right click on Save Image As or Copy Image. A Control-V after Copy Image can move your image directly into Paint.

4. To save all image objects, go to Tools-->Document Processing --> Export All Images

5. Convert the .PNG image to .BMP by using Paint. Some other useful manipulation tools include ImageViewer32 or else IrfanView.

6. Your image must end up in .JPG format before returning to Acrobat Distiller.

And here's how your Gonzo textfile can link your images back...

RETURNING IMAGES TO ACROBAT DISTILLER

1. **Make sure your edited images are in .JPG format. Then very carefully note the image resolution.**

2. **Define the following procs early in your Gonzo textfile...**

```
/jpegimageprocwithlink { %offset voffset hres vres
save /snapjpg exch def /infilename exch store  /inurllink
exch store /photoscale exch store /vpixels exch store
/hpixels exch store inurllink setareaurl /DeviceRGB
setcolorspace 0 0 translate hpixels vpixels scale photoscale
dup scale /infile infilename (r) file def /Data {infile
/DCTDecode filter} def <</ImageType 1 /Width hpixels
/Height vpixels /ImageMatrix [hpixels 0 0 vpixels neg
0 vpixels ] /DataSource Data /BitsPerComponent 8 /Decode
[0 1 0 1 0 1]>>image ypos jpgsnap restore /ypos exch
def} def

/setareaurl {/cururlname exch store mark /Rect  [ 0 0
hpixels photoscale mul vpixels photoscale mul /Border
[ 0 0 0] /Color [ .7 0 0 ] /Action <</Subtype /URI
/URI cururlname>> /Subtype /Link /ANN pdfmark} def
```

3. **Define the custom Gonzo textfile linking info...**

```
/autoimageandlink21 {save /af1 exch store  xpos ypos
yinc add translate
0 0  1630 1252  .026 % xpos ypos xres yres pixscale
(http://www.tinaja.com/images/gorsort1.jpg)  % url first
(C:\\Docu....) % image source second
jpegimageprocwithlink af1 restore} store
```

4. **Call the image for placement when and where wanted...**

```
4.5 10 (|/autoimageandlink21 ) cl
```

These routines handle two tasks for you. They first get a .JPG image back into Acrobat's Distiller. And secondly, they provide a **click to expand** feature.

Normally, **a reduced file size .PDF image will look awful if it is magnified**. The workaround here is to replace the awful magnification with a web delivered larger image of chosen size, quality, and sharpness. Note that only a very few bytes are needed inside the .PDF file for this web-delivered linking.

Note that seven parameters need passed to **jpegimageprocwithlink**. These are the x and y offset from the present page position, the horizontal and vertical resolution, the image size scaling, the click-to-expand URL, and the local .JPG image source. Note that **the resolution values must be exact**; Otherwise the image will end up slanted or even totally trashed. Watch this detail.

While the passed JPG images may be quite large, they will get properly resized during Acrobats **Reduced File Size** operation. Should a bitmap or image ever end up wildly the wrong size, chances are its header was modified. The resolution is easily reset with **Irfanview**.

Should the quality of an image degrade, consider raising and maintaining its resolution in all process steps.

Our gorilla images in the demo show us several value-added features. They first were colorized by using the bucket in Paint. This is an amazingly fast, fun, and easy task. But you do have to get rid of the B/W line breaks and **make sure each "pour" area is surrounded by black or color pixels**. A single white edge pixel can cause a major breakout.

Both fonts and images in any .PDF file are easily viewed by getting into Acrobat X and selecting the rather buried **Tools-->Print Production-->Preflight-->Options -->Create Inventory.**

Be sure to remember that **Distiller versions after 8.1 default to preventing most disk access**. The workaround is to run **acrodist-F** from your XP command line.

For More Help

Additional examples and tutorials appear in our **GuruGram** and **PostScript** libraries.

Seminars, training, **consulting**, and direct eBook conversion projects are available. You can **email me** or call (928) 428-4073 for more details.