PDFConv – PDF converter for the Apple Newton

(C) Copyright 2003 Hendrik Lipka

June 9, 2003

Contents

1 Introduction

The PDF Converter is designed to convert a PDF file into an format that is suitable for displaying on a Newton MessagePad or eMate. The program can generate output for a Web Browser, for Newton Book Maker, or for the Newton ToolKit. The output could probably also be used for Newton Press, if you can get that program to work correctly.

The output is in an image format, rather than HTML or some other formatted text. The reason is that it is virtually impossible to render a page accurately outside of a PDF, especially on a Newton. Many page elements, such as rotated text, are not available outside of PDF, at least on a Newton.

Therefore, PDFConvert writes its output to image files that can then be loaded onto the Newton and viewed in precisely their original format, with all of their fonts, graphics, shading, etc. Note that depending on the desired resolution, and on how many graphics the original PDF contained, the converted document will be two to ten times larger than the original PDF.

The PDFConverter program itself is written in Java 2, and should run on any computer that supports Java 1.3 or 1.4. Unfortunately, the 'classic' MacOS (8.x or 9.x) does not have a Java 2 runtime available, and therefore can not run PDFConverter, although the BookMaker output can be formatted for the Mac version of BookMaker.

Thanks to Eric Schneck (eschneck@mindspring.com) for testing the conversion on a Macintosh and for writing parts of this documentation.

2 Installation

The program is delivered as a single ZIP file. The installation consists of the following steps:

1. Install a Java 2 runtime if you have not already done so
2. Create a directory on your hard disc for installing the program
3. Unpack the ZIP file file into this directory. When asked, overwrite all files (except if you have files modified yourself)

After the installation, you should read the documentation fully[1].

[1] You are just reading the first chapter

3 Program usage

Command Line Usage:

run [options] sourcefile [sourcefile [...]]

where *sourcefile* is the name of the PDF file to convert (it is allowed to specify more than one source file). This argument is required.

NOTE: under windows, you can also use the supplied pdfconv.exe to start the program.

the supported options are:

-basename base name for all output files

-pdfscale the scale factor for generating the internal images from PDF (float value)

-maxwidth maximum width of output images

-maxheigth maximum height of output images

-scalefactor scale factor of output images

-quality quality of JPEG files as float value (0.85 as default)

-targetdir target directory for output files

-writePICT write PICT images (no value expected)

-writeBMP write BMP image (gray)

-writeBMP2 write BMP image (black/white)

-writeJPEG write JPEG images

-writeHTML write HTML file

-writeBook write BookMaker file (Win version if BMP files, Mac version if PICT files are written, or both)

-writeNTK write NTK file (Win version if BMP files, Mac version if PICT files are written, or both)

-verbose write informational messages during conversion

-debug write debug output

-pages specify which pages should be converted (page:page:from-to:from-:-to)

-help print usage

-version print usage

Options are not required[1].

If no *basename* is given, it will be determined from the name of the given PDF file. If more than one *sourcefile* is specified, output files are created for each source file. Of multiple source files and a *basename* are specified, all source files are combined into a single output file.

3.1 image scaling

When creating the images, they are scaled to the largest dimension satisfying the given conditions (if given any). For example: The images would be 800x400 pixels. The scale is 0.5, *maxwidth* is set to 300 pixel. The the resulting images are 300x150 pixels. If *maxwidth* would be set to 500 pixels, the images would be 400x200 pixels.

Note, that scaling the images is not always good. It provides better quality than scaling them on the Newton during display (and reduces the image size), but may be lead to unreadable text. Feel free to experiment a little bit to determine the best compromise between size and readability.

PDFConv will report the final scale factor, when debug mode is enabled. You can get better results when using the "-pdfscale" parameter. This sets the internal scale factor, when the PDF file is rendered into an internal image. If you set "-pdfscale" to the correct scale factor, no scaling of the finished is required. This results in a better image quality[2].

3.2 output targets

When writing BookMaker or NTK book files, the output format to be used (Mac or Win) will be determined from the written images. If PICT images are written, the book will be written in Mac format. If BMP images are written, the book will be written in Win format. If no images are written, both formats will be generated.

Creation of HTML or book files and images are independent of each other.

NOTE: output files will always be overwritten!

[1] but no output will be written if no options are given

[2] PDFConv renders each page of the PDF into an internal image, using the jpedal library. This image is then scaled down to the right size. If the internal size is the right one, no interpolation errors from the scaling can happen.

3.3 Specify page ranges

The command line option "-pages" can be used to specify the pages to be converted. The page numbers are separated by ":". To specify a range, use "from-to". If "from" is omitted, all pages from the beginning to the given page are converted. If "to" is omitted, all pages from the given page to the end are converted.

The allowed page numbers are "1" to the number of pages in the document. Pages can be listed multiple times, but are converted only once.

The "-pages" option is only useful when using a single input file, but will also be used if multiple files are given.

3.4 Using the configuration file

If the file "pdfconv.cfg" is present in the current directory, it will be loaded by PDFConv. It contains options, which are then used. These options can be overridden by the options on the command line. If a boolean option is set in the config file, like "imagetarget=BMNP", it can be overridden by specifying "-writeBMP=false" on the command line. The following options are known:

basename -baseame

targetdir -targetdir

pdfscale -pdfscale

imagetarget list of BMP, BMP2, PICT, JPEG, like the corresponding -write* options

maxwidth -maxwidht

maxheigth -maxheigth

quality -quality

scalefactor -scalefactor

texttarget list of NTK, Book, HTML, like the correspondig -write* options

verbose -verbose

debug -debug

pages -pages

Options can be set as comment by having "#" as the first character in a line.

4 Creating books with NTK

If you don't have a web server available, or you do not have a TCP/IP connection on your Newton, you can generate a NewtonBook on your desktop computer, and download it to your Newton (or post the finished package file on-line, if there is no copyright problem). This conversion can be done via writing a Newton ToolKit book source file, or by writing a book file for the BookMaker application.

The BookMaker file can be generated for the Windows and the Mac version of BookMaker [1], as can the files for the NTK. Currently, the NTK files for the Mac cannot be used for conversion, as PDFConv cannot write the images as resource fork.

4.1 Generate a Newton book on Windows

1. Run the PDFConvert program. On a Windows NT machine, this would be:

   D:\PDFConv> run -writeBMP -writeNTK myfile.PDF

   The *-writeBMP* switch tells PDFConvert to generate BMP files, the *-writeNTK* switch to write a NTK book source file. This switch must come before the file names on the command line. The *'myfile.PDF'* is the name of your PDF. The basename will also be the .title and .isbn of your finished document.

   There are additional arguments to specify the size and location of your BMP images. See the readme for more information.

2. copy the generated *.f file and *.bmp files to a NTK project folder.

3. Start NTK

4. create a new project ('Project' / 'New Project...') in the project folder containing your files

5. add the generated *.f file to the project ('Project' / 'Add File...')

6. change the project settings ('Project' / 'Settings...') :

   - under 'Application', set the name of your files

[1] the files differ slightly

- under 'Project'
  - set Platform to 'Newton21'
  - enable 'Newton OS 2.0 only'
  - disable 'Compile for Debugging'
- under 'Package'
  - set a name for the package on your Newton
  - enable compression
- under 'Output', set output to 'Book'

7. build the project ('Project' / 'Build Package')
8. the generated package gets the same name as your project file

That's it. You have converted a PDF to a Newton Book.

4.2 How to change the page format

Open the *.f file generated by PDFConv in an editor. (On a Mac, you need to use BBEdit[2] because the file is too large for SimpleText, and NTK will damage the resource fork of the file. If you get a "missing resource" error when you build your project, it is because you used the wrong editor. Re-run PDFConv and edit the file again)

Find the text that looks like this:

```
// Bounds List
bnd1 := [0,16,288,334];

// Pages
pageList := pageSize: left: 0, top: 0, right: 240, bottom: 318,
contents: [], pages: [];
```

Change the bounds and pageSize. Note that the pagesize in your original file will always be 240 x 318, but the bounds list will vary based on your image size. In any case, change both of them to 320 x 432. The result looks like this:

```
// Bounds List
bnd1 := [0,16,320,432];

// Pages
```

[2] http://www.barebones.com/products/bblite/index.shtml

```
pageList := pageSize: left: 0, top: 0, right: 320, bottom: 432,
contents: [], pages: [];
```

Save the file and quit the editor. Then you can run NTK.

5 Creating Newton books with BookMaker

If you don't have a web server available, or you do not have a TCP/IP connection on your Newton, you can generate a NewtonBook on your desktop computer, and download it to your Newton (or post the finished package file on-line, if there is no copyright problem).

1. Run the PDFConvert program. On a Windows NT machine, this would be:

   D:\PDFConv> run -writePICT -writeBook myfile.PDF

   The -writePICT switch tells PDFConvert to generate PICT files for BookMaker. (use -writeBMP instead to generate Win-BookMaker output) This switch must come before the file names on the command line. The 'myfile.PDF' is the name of your PDF. The basename will also be the .title and .isbn of your finished document.

   There are additional arguments to specify the size and location of your PICT images. See the readme for more information.

2. Transfer the .txt file and the .pict files from the output directory of your Java computer to your Mac (or to a different folder on your Mac, if you are running MacOS X.)

   It is recommended to store the files in a folder called PDFConvert, in the Documents folder, on your startup disk. The PDFConvert program assumes that the PICT files will be in a folder with this name. (Note that PDFConvert is spelled out in full. The program folder was PDFConv.)

   Also copy the PDFConvert.mac file from your PDFConv directory on your PC or Unix computer to the PDFConvert folder in the Documents folder on your Mac. This file is the NTK project file for your book.

3. If you have not already done so, install NewtonDev on your Mac.

   http://www.unna.org/unna/apple/development/NTK/macntk/NTKMac164b3.sit.hqx

   This will install the Newton Book Maker and Newton Tool Kit that you need to build Newton Books. (It will also install Newton Press. Good luck).

Increase the memory allocation of BookMaker and NTK, since they are both too small for any reasonably sized book. Increase BookMaker to 28MB, and NTK to 32MB.

4. (Optional) If your hard disk is not named 'Macintosh HD', or you did not create a PDFConvert folder in the Documents folder of your hard disk, then you need to edit the file names in your NewtonBook document.

   Use BBEdit (or SimpleText) to open the text file that you generated in PDFConvert. In the example above, the file would be myfile.txt, and would be in the myPDF folder on your desktop. It will look something like this:

   .title PDF
   .isbn PDF:EMSI
   .picture ':Macintosh HD:Documents:PDFConvert:myfile_page_1.pict'
   .picture ':Macintosh HD:Documents:PDFConvert:myfile_page_2.pict'
   .picture ':Macintosh HD:Documents:PDFConvert:myfile_page_3.pict'

   Change each .picture line to include the full path to your .pict files. In the example, the name of the hard disk has been changed. Note the leading colon in the path:

   .title PDF
   .isbn PDF:EMSI
   .picture ':Clara:Documents:myPDF:myfile_page_1.pict'
   .picture ':Clara:Documents:myPDF:myfile_page_2.pict'
   .picture ':Clara:Documents:myPDF:myfile_page_3.pict'

   Save the BookMaker file and exit the editor.

5. Launch Newton Book Maker. It is located in the Newton Book Maker folder of the Book Tools folder of the NewtonDev folder on your hard disk.

   Select File-¿Open, and open your myfile.txt document.

   Press Do It. If there are any errors, fix the text file and try again. The most likely problem is mistyping the path to your pict files.

   If all goes well, Book Maker will run for a few seconds, and then prompt you for a location to save the file. Give it the name PDFConvert.txt.f, and save it in your PDFConvert folder. (You need to call the file PDFConvert.txt.f, because that is the name that the NTK project file will be looking for. If you are competent in NTK, you can rename the file to whatever you like.)

6. Exit BookMaker

7. (Optional) Resize your PDF for the Newton 2x00 screen size. See the NTK creation chapter for this.

8. Now we are ready to build our package. See the NTK creation chapter for information how to do this.

That's it. You have converted a PDF to a Newton Book.

6 Creating HTML

If your Newton has a TCP/IP connection and you have a web server available to which you can upload your files, then this is the easiest option for using PDF Converter. The steps are as follows:

1. Copy your PDF into the same directory as the run.bat file.

2. Run the program. On a Windows machine, this would be:

   D:\PDFConv> run -writeHTML -writeJPEG myfile.PDF

   The *'myfile.PDF'* is obviously the name of your PDF.

   The *basename* will also be the <Title> of your finished document.

   There are additional command-line arguments to specify the size and quality and location of your JPG images. See the online help for more information.

3. Upload the .html file and the .jpg files from the output directory to your Web server. The output directory will be under the directory that has your run.bat file. In the example above, it would be D:\PDFConv\output .

4. Install Newtscape on your Newton if you have not already done so:

   http://mywebpages.comcast.net/saweyer/newton/newtscape.htm

   Make sure that you have the JPG plugin (JPEG10f2.pkg) and the NewtPack plugin (pack32.pkg) installed. It is also recommended to turn off image scaling in Newtscape, since the text will be too small to read. To turn off scaling, uncheck *i→General→Images→scale to fit.*

5. Connect your Newton to the Internet, and launch Newtscape. Open the URL for the html file that you uploaded to the web server.

6. Once the pages appear on-screen, you can read them on your Newton, or select *File→Save as Package* to save the document as a Newtonbook.

7. You can then erase the pages from your web server, if you want to.

7 Modify the output

The output files (except for the images) are generated from template files. When generating a file, PDFConv[1] reads the template, parses the contained data, replaces are placeholders with the proper data, and writes the output file.

The templates are located in the "templates" subdirectory. The names should make clear when each file is used. If you want to modify a template, please make a backup.

7.1 template syntax

The syntax used in the templates is very simple. Basically, the tags look like HTML tags. Only 2 of them are used normally:

tmpl_var replace the tag with the value of the given variable

tmpl_loop loop over all elements in the given iterator

7.1.1 tmpl_var

The syntax is "<tmpl_var VARNAME>". In the written output, the tag will be replaced with the value of the variable. The variable name is case sensitive.

7.1.2 tmpl_loop

The syntax is "<tmpl_loop ITERATORNAME> loop content >/tmpl_loop>". For all elements of the iterator, the loop content will be processed and written to the output file. Iterator names are case sensitive. Inside of the loop, only variables defined in the current row of the iterator are available (but no variables defined in the outer scope).

7.2 available variables and loops

Global variables:

- Title

[1] better: the used htmltemplate library does

- ImageCount
- ISBN

Loops:

- Images, with the following variables available:
  - FileName
  - Width
  - Heigth
  - PageNum

7.3 example

The example is taken from the template "html.tmpl".

```
<HTML>
<HEAD>
<TITLE><tmpl_var Title></TITLE>
</HEAD>
<BODY>
<tmpl_loop Images>
<IMG SRC="<tmpl_var FileName>.jpg" ALT="<tmpl_var FileName>"
BORDER="0"><br>
</tmpl_loop>
</BODY>
</HTML>
```

Note the usage of the htmltemplate tags inside of the HTML tags (even in attributes). This works because the template engine does only a textual scan for its tags, but does not care about all the other text.

Inside the loop, additionally to "FileName", the variables "PageNum", "Width" and "Heigth" are available, but neither "Title" nor "ImageCount".