MakeFont Utility

Adding new fonts and encoding support

This section explains how to use **TrueType** or **Type1** fonts so that you are not limited to the standard fonts any more. The other interest is that you can choose the font encoding, which allows you to use other languages than the Western ones (the standard fonts having too few available characters).

There are two ways to use a new font: embedding it in the PDF or not. When a font is not embedded, it is sought in the system. The advantage is that the PDF file is lighter; on the other hand, if it is not available, a substitution font is used. So it is preferable to ensure that the needed font is installed on the client systems. If the file is to be viewed by a large audience, it is better to embed the fonts.

Adding a new font requires three steps for **TrueType** fonts:

- Generation of the metric file (.afm)
- Generation of the font definition file (.xml)
- Declaration of the font in the program

For **Type1**, the first one is theoretically not necessary because the AFM file is usually shipped with the font. In case you have only a metric file in PFM format, it must be converted to AFM first.

Generation of the metric file

The first step for a **TrueType** font consists in generating the AFM file (or UFM file in case of a **Unicode TrueType** font). A utility exists to do this task: `ttf2ufm` - a special version of `ttf2pt1` - allowing to create AFM and/or UFM files. `ttf2ufm` has been modified to generate AFM and UFM files containing all the information which is required by the utility program **makefont**. An archive containing the modified source code of `ttf2ufm` and a Windows executable can be downloaded from here. The command line to use is the following:

```
ttf2ufm -a font.ttf font
```

For example, for Comic Sans MS Regular:

```
ttf2ufm -a c:/windows/fonts/comic.ttf comic
```

Two files are created; the one we are interested in is comic.afm.

Remarks:

Starting with **wxPdfDocument** version 0.8.5 this step may be ommitted for TrueType and OpenType fonts.

Generation of the font definition file

The second step consists in generating a **wxPdfDocument** font metrics XML file containing all the information needed by **wxPdfDocument**; in addition, the font file is compressed. To do this, a utility program, **makefont**, is provided.

```
makefont {-a font.afm | -u font.ufm | -i } [-f font.{ttf|pfb}] [-e encoding] [-p
patch] [-t {ttf|otf|t1}] [-o outdir]
```

`-a font.afm` AFM font metric file for **TrueType** or **Type1** fonts

`-u font.ufm` UFM font metric file for **TrueType Unicode** or **OpenType Unicode** fonts

`-i` Extract font metrics directly from **TrueType Unicode** or **OpenType Unicode** fonts

`-f font.{ttf|otf|pfb}` font file (`.ttf` = TrueType, `.otf` = OpenType, `.pfb` = Type1).
If you own a Type1 font in ASCII format (`.pfa`), you can convert it to binary format with t1utils.
If you don't want to embed the font, omit this parameter. In this case, type is given by the type parameter.

`-e encoding` font encoding, i.e. cp1252. Omit this parameter for a symbolic font.like *Symbol* or *ZapfDingBats*.

The encoding defines the association between a code (from 0 to 255) and a character. The first 128 are fixed and correspond to ASCII; the following are variable. The encodings are stored in .map files. Those available are:

- cp1250 (Central Europe)
- cp1251 (Cyrillic)
- cp1252 (Western Europe)
- cp1253 (Greek)
- cp1254 (Turkish)
- cp1255 (Hebrew)
- cp1257 (Baltic)
- cp1258 (Vietnamese)
- cp874 (Thai)
- iso-8859-1 (Western Europe)
- iso-8859-2 (Central Europe)
- iso-8859-4 (Baltic)
- iso-8859-5 (Cyrillic)
- iso-8859-7 (Greek)
- iso-8859-9 (Turkish)
- iso-8859-11 (Thai)
- iso-8859-15 (Western Europe)
- iso-8859-16 (Central Europe)
- koi8-r (Russian)
- koi8-u (Ukrainian)

Of course, the font must contain the characters corresponding to the chosen encoding. The encodings which begin with cp are those used by Windows; Linux systems usually use ISO. Remark: the standard fonts use cp1252.

Note: For TrueType Unicode and OpenType Unicode fonts this parameter is ignored.

`-p patch` patch file for individual encoding changes. Use the same format as the `.map` files for encodings. A patch file gives the possibility to alter the encoding. Sometimes you may want to add some characters. For instance, ISO-8859-1 does not contain the euro symbol. To add it at position 164, create a file containing the line

```
!A0 U+20AC Euro
```

Note: The Unicode character id will not be interpreted.

	For TrueType Unicode and OpenType Unicode fonts this parameter is ignored.
`-t {ttf|otf|t1}`	font type (ttf = TrueType, otf = OpenType, t1 = Type1). Only needed if omitting the font file.
`-o outdir`	the output directory for all generated files (default: current working directory)

Note: in the case of a font with the same name as a standard one, for instance arial.ttf, it is mandatory to embed. If you don't, Acrobat will use its own font.

Executing `makefont` generates an .xml file, with the same name as the `.afm` file resp. `.ufm` file. You may rename it if you wish. In case of embedding the font file is compressed and gives a file with `.z` as extension. For **Unicode TrueType** fonts a file with extension `.ctg.z` is created containing the character to glyph mapping.You may rename these files, too, but in this case you have to alter the file name(s) in the file tag in the `.xml` file accordingly.

You have to copy the generated file(s) to the font directory.

Declaration of the font in the script

The last step is the most simple. You just need to call the AddFont() method. For instance:

```
pdf.AddFont(wxT("Comic"),wxT(""),wxT("comic.xml"));
```

or simply:

```
pdf.AddFont(wxT("Comic"));
```

And the font is now available (in regular and underlined styles), usable like the others. If we had worked with Comic Sans MS Bold (comicbd.ttf), we would have put:

```
pdf.AddFont(wxT("Comic"),wxT("B"),wxT("comicbd.xml"));
```

Reducing the size of TrueType fonts

Font files are often quite voluminous; this is due to the fact that they contain the characters corresponding to many encodings. zlib compression reduces them but they remain fairly big. A technique exists to reduce them further. It consists in converting the font to the **Type1** format with `ttf2pt1` by specifying the encoding you are interested in; all other characters will be discarded. For instance, the arial.ttf font shipped with Windows 98 is 267KB (it contains 1296 characters). After compression it gives 147. Let's convert it to **Type1** by keeping only cp1250 characters:

```
ttf2ufm -b -L cp1250.map c:/windows/fonts/arial.ttf arial
```

The `.map` files are located in the `makefont` directory. The command produces arial.pfb and arial.afm. The arial.pfb file is only 35KB, and 30KB after compression.

It is possible to go even further. If you are interested only in a subset of the encoding (you probably don't need all 217 characters), you can open the .map file and remove the lines you are not interested in. This will reduce the file size accordingly.

Since **wxPdfDocument** version 0.8.0 automatic font subsetting is supported for TrueType und TrueType Unicode fonts. Since version 0.8.5 subsetting of OpenType Unicode fonts is supported as well. **Note**: The font license must allow embedding and subsetting.
