Pysilfont User Documentation

Installation

Pull the repo from https://github.com/silnrsi/pysilfont.git

In the pysilfont directory, install by running:

```
sudo python setup.py install --record installed-files.txt
```

This will install some scripts (currently just ufoConvert) as well as the python libraries needed by all scripts.

Look in INSTALL.txt for information on uninstalling Pysilfont.

If running on a Mac, there is extra information in scripts/tools/actionosx for setting up so that UFOs can be normalized via right-clicking on a UFO.

If you have downloaded from the github repository, there will be many more scripts available which by be run using:

```
python <path>/<script name>.py ...
```

Basic usage

Most commands provide basic help with the "-h" option, eg

```
$ UFOconvert -h
usage: UFOconvert [-h] [-d] [-v VERSION] [-p PARAMS] ifont [ofont]

Convert/normalise a UFO.
- If no options are chosen, the output font will simply be a normalised version
of the font.

positional arguments:
  ifont                 Input font file
  ofont                 Output font file

optional arguments:
  -h, --help            show this help message and exit
  -d                    Display help with info on default values
  -q, --quiet           Quiet mode - only display errors
  -v VERSION, --version VERSION
                        UFO version to output
  -p PARAMS, --params PARAMS
                        Other font parameters

Version: 1.0.0
```

Reporting

Most UFO scripts support standard reporting.

Reporting is both to screen and a log file, with different levels of reporting set for each via loglevel (default W) and scrlevel (default P) parameters which can be set to one of:

- E Errors
- P Progress - Reports basic progress messages and all errors
- W Warning - As P but with warning messages as well
- I Info - As W but with information messages as well

- V Verbose - even more messages!

Log levels can be set using -p (eg -p scrlevel=v).

-q --quiet sets quiet mode where the scrlevel is set to E (and some additional messages suppressed) so only errors are reported on screen.

Default values

Font/file names

Once the initial input file (eg input font) has been given, most other font and file names will have defaults based on those.

This applies to other input font names, output font names, input file names and output file names and is done to minimise retyping repeated information like the path the files reside in. For example, simply using:

```
python UFOsetPSnames.py   /path/font.ufo
```

will:

```
open (and update)          /path/font.ufo
backup the font to         /path/backups/font.ufo.nnn~
read its input from        /path/font_psnames.csv
write its log to           /path/font_psnames.log
```

If just part of a file name is supplied, other parts will default, eg if just 'test' was supplied for the output font name, the font would be output to `/path/test.ufo`.

If a full file name is supplied, **but no path**, the the current working directory will be used, so if "test.ufo" is supplied it **won't** have `/path/` added.

Other parameters

Other parameters will just have standard default value

Displaying defaults for a command

Use the "-d" option to see what defaults are for a given command, eg for UFOsetPSnames.py, `-d` will output its help text with the following appended:

```
Defaults for parameters/options

  Font/file names
    -i                  _PSnames.csv
    -l                  _PSnames.log

If the default value starts with "_" (as above) then the input font name will be
prepended to the default value.
```

Backups for fonts

If the output font name is the same as the input font name (which is the default behaviour for most scripts), then the original font is backed up prior to outputting the revised font.

By default, the last 5 copies of backups are kept in a sub-directory called "backups". These defaults can be changed using parameters.

Parameters

There are many parameters that can be set to change the behaviour of scripts.

(Details still to be documented)

Specific scripts

UFOconvert

```
UFOconvert [-h] [-d] [-v VERSION] [-p PARAMS] ifont [ofont]
```

This will convert between UFO 2 and UFO3 (if -v is used to specify the alternative version) or otherwise simply normalize the UFO by 'converting' to the existing version.

Current default normalization behaviours include:

- XML formatting
    - Use 2 spaces as indents
    - Don't indent the <dict> for plists
    - Sort all <dicts>s in ascending key order
    - Where values can be "integer or float", store integer values as <integer>
    - Limit <real> precision to 6
- glif file names - use the UFO 3 suggested algorithm
- order glif elements and attributes in a predetermined order

Most of the above can be overridden by parameters supplied using -p - see above.

Current Limitations

The following are known limitations that are due to be addressed in the future

- UFO 3 specific folders (data and images) are not copied
- Converting from UFO 3 to UFO 2 only handles data that has a place in UFO 2 - but does include converting UFO 3 anchors to the standard way of handling them in UFO 2