TTFLIB

COLLABORATORS			
	TITLE : TTFLIB		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		April 15, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

TTFLIB

1.1   main

                       ttf.library version 0.8.2

A truetype font engine for Amiga computers.

                    Overview

                    Credits/Legal

                    Requirements

                    Installation

                    Usage

                      ttfmanager

                      ttfinstall

                      ttflist

                      ttfcp

                      Bitline

                      ftview

                   Limits

                   Recent Changes

                   Future

                   FAQ
                 The most current publicly available version of this software
can be found at the following web address:

```
      http://ragriffi.home.sprynet.com/
```

Send comments, suggestions, and bug reports to:

```
  Richard Griffith
  ragriffi@sprynet.com
```

1.2   ttf.library Overview

```
Overview
```

ttf.library is a truetype compatible font engine for Amiga OS.
It functions in a manner compatible with the outline font engine
standard established by Commodore with the bullet.library engine
for compugraphic format fonts. This means that Amiga applications
which use normal system fonts are now able to use truetype fonts.

1.3   Credits and Legal Issues

```
              Credits and Notes
```

The ttf.library and related programs were made possible by
the outstanding achievement known as the FreeType Project.
For more information, visit http://www.freetype.org/

Also, Amish S. Dave's type1.library (which does for postscript
type1 fonts what ttf.library does for truetype) served as an
invaluable inspiration and an initial guide for the rather
poorly documented amiga outline font engine format.

Bitline, as distributed here, is an adaptation of a program
of the same name written by Georg Steger.

```
Legal
```

I am not a lawyer, nor do I want to be, so expect plain
english here.

First, this library comes with NO WARRANTY.

ttf.library is free, think of it as my gift to the faithful.
I do not restrict its use and/or distribution, however, some
of the technology used is covered by a separate license, see
the
              FreeType license
               for details. I do ask that if you
distribute it, please don't try and call it your own work. I
know better, you know better, and I'm sure karma will get you.
If you sell it, be advised you will have stiff competition,

as it will continue to be available free. (As a side note, no
I don't think all software should be free. I make my living as
a programmer, consultant, general geek type, so I do place a
great value on software. This one, though, I feel should be
free. So there...)

Also note that it is virtually impossible to refer to anything
relating to the subject of fonts without the mention of at least
some trademarked names. ttf.library asserts NO CLAIM to ANY sort
of trademark whatsoever. The following list of trademarks is
provided, but may not be complete:

 Amiga® is a trademark of Amiga, Inc.
 TrueType is a trademark of Apple Computers, Inc.
 Unicode® is a registered trademark of Unicode, Inc.
 Microsoft® and Windows® are registered trademarks of Microsoft
   Corporation
 M$ and Windoze are derisive terms, unregistered by any party
   to the best of my knowledge.

1.4 FreeType Project License

```
                   The FreeType Project LICENSE
                   ----------------------------


                      Copyright 1996-1999 by
             David Turner, Robert Wilhelm, and Werner Lemberg



Introduction
============

  The FreeType  Project is distributed in  several archive packages;
  some of them may contain, in addition to the FreeType font engine,
  various tools and  contributions which rely on, or  relate to, the
  FreeType Project.

  This  license applies  to all  files found  in such  packages, and
  which do not  fall under their own explicit  license.  The license
  affects  thus  the  FreeType  font  engine,  the  test  programs,
  documentation and makefiles, at the very least.

  This  license  was  inspired  by  the  BSD,  Artistic,  and  IJG
  (Independent JPEG  Group) licenses, which  all encourage inclusion
  and  use of  free  software in  commercial  and freeware  products
  alike.  As a consequence, its main points are that:

    o We don't promise that this software works.  However, we are be
      interested in any kind of bug reports. ('as is' distribution)

    o You can  use this software for whatever you  want, in parts or
      full form, without having to pay us. ('royalty-free' usage)
```

    o You may not pretend that you wrote this software. If you use
      it, or only parts of it, in a program, you must acknowledge
      somewhere in your documentation that you've used the FreeType
      code. ('credits')

  We   specifically   permit and encourage   the inclusion of this
  software, with or without modifications, in commercial products,
  provided that all warranty or liability claims are assumed by the
  product vendor.

Legal Terms
===========

0. Definitions

  Throughout this license, the terms 'package', 'FreeType Project',
  and 'FreeType archive' refer to the set of files originally
  distributed by the authors (David Turner, Robert Wilhelm, and
  Werner Lemberg) as the 'FreeType project', be they named as alpha,
  beta or final release.

  'You' refers to the licensee, or person using the project, where
  'using' is a generic term including compiling the project's source
  code as well as linking it to form a 'program' or 'executable'.
  This program is referred to as 'a program using the FreeType
  engine'.

  This license applies to all files distributed in the original
  FreeType archive, including all source code, binaries and
  documentation, unless otherwise stated in the file in its
  original, unmodified form as distributed in the original archive.
  If you are unsure whether or not a particular file is covered by
  this license, you must contact us to verify this.

  The FreeType project is copyright (C) 1996-1999 by David Turner,
  Robert Wilhelm, and Werner Lemberg. All rights reserved except as
  specified below.

1. No Warranty

  THE FREETYPE ARCHIVE IS PROVIDED 'AS IS' WITHOUT WARRANTY OF ANY
  KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
  WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
  PURPOSE. IN NO EVENT WILL ANY OF THE AUTHORS OR COPYRIGHT HOLDERS
  BE LIABLE FOR ANY DAMAGES CAUSED BY THE USE OR THE INABILITY TO
  USE, OF THE FREETYPE PROJECT.

  As you have not signed this license, you are not required to
  accept it. However, as the FreeType project is copyrighted
  material, only this license, or another one contracted with the
  authors, grants you the right to use, distribute, and modify it.
  Therefore, by using, distributing, or modifying the FreeType
  project, you indicate that you understand and accept all the terms
  of this license.

2. Redistribution

  Redistribution and use in source and binary forms, with or without
  modification, are permitted provided that the following conditions
  are met:

    o Redistribution of source code must retain this license file
      ('licence.txt') unaltered; any additions, deletions or changes
      to   the original   files must   be clearly   indicated in
      accompanying documentation.   The copyright notices of the
      unaltered, original files must be preserved in all copies of
      source files.

    o Redistribution in binary form must provide a disclaimer that
      states that the software is based in part of the work of the
      FreeType Team, in the distribution documentation. We also
      encourage you to put an URL to the FreeType web page in your
      documentation, though this isn't mandatory.

  These conditions apply to any software derived from or based on
  the FreeType code, not just the unmodified files. If you use our
  work, you must acknowledge us. However, no fee need be paid to
  us.

3. Advertising

  The names of FreeType's authors and contributors may not be used
  to endorse or promote products derived from this software without
  specific prior written permission.

  We suggest, but do not require, that you use one or more of the
  following phrases to refer to this software in your documentation
  or advertising materials: 'FreeType Project', 'FreeType Engine',
  'FreeType library', or 'FreeType Distribution'.

4. Contacts

  There are two mailing lists related to FreeType:

    o freetype@freetype.org

      Discusses general use and applications of FreeType, as well as
      future and wanted additions to the library and distribution.
      If you are looking for support, start in this list if you
      haven't found anything to help you in the documentation.

    o devel@freetype.org

      Discusses bugs, as well as engine internals, design issues,
      specific licenses, porting, etc.

    o http://www.freetype.org

Holds the current FreeType web page, which will allow you to
download our latest development version and read online
documentation.

You can also contact us individually at:

David Turner   <david.turner@freetype.org>
Robert Wilhelm   <robert.wilhelm@freetype.org>
Werner Lemberg   <werner.lemberg@freetype.org>

--- end of license.txt ---

1.5 System Requirements

Requirements

OS3.0 or higher

Separate archives are available with optimized versions for
68020, '030, '040, '060, as well as generic 68000 series CPU's.

1.6 Installation

Installation

Install-ttflib is a standard installer script, just double
click the icon.

If you prefer a manual installation, here is a breakdown
of the primary components:

```
File          Installation instructions
------------  -------------------------------------------------------
ttf.library  copy to your LIBS: drawer
ttfmanager   copy to your favorite tool drawer (i.e. sys:utilities)
ttfinstall    "   "   "    "      "     "     (optional)
ttflist       "   "   "    "      "     "     (optional)
ttfcp         "   "   "    "      "     "     (optional)
Bitline       "   "   "    "      "     "     (optional)
ftview        "   "   "    "      "     "     (optional)
```


1.7 Using ttf.library

Usage

The following tools are available:

ttfmanager
make truetype fonts available to system

```
        ttfinstall
        alternate (CLI) tool to install fonts

        ttflist
        list names and other details from font files

        ttfcp
        codepage tool for non ECMA-Latin1 usage

        Bitline
        create bitmap versions of an outline font

        ftview
        preview a font
```

1.8   Using ttfmanager

```
        ttfmanager
```

To use a truetype font, it must first be installed to some part
of the system FONTS: drawer. Installation is a bit different than
with the compugraphic intellifont program. First, outlines (the
actual truetype font files) are NOT copied to the font directory.
Instead only the ".font" and ".otag" files are created in the
selected directory, and those reference the outlines at the same
locations from which they were installed. This makes it much
easier to use large font libraries directly from CD-ROM, for
example. The font files used by the bullet.library in some cases
are actually modified by the installation process, making the
copying a requirement. In contrast, ttf.library uses unmodified
files, so there is no reason to restrict your choices in disk
space management. If you really want a "_ttfoutlines" directory
in FONTS:, create it, and copy your font files there before the
installation.

The second major difference is that the fonts may be installed
to any drawer, not just a FONTS: component. (Note that to USE a
font in many programs, the drawer must be assigned to FONTS:,
but this allows you to selectively add the required font path
components as needed.) This allows for a rotating assignment
for FONTS: which is practically required with any large font
collection. (Image how long it would take to load the standard
system font requester if you had 6000 fonts actually available!)

Also see:

        Quick Instructions

        Main Window

        Options Window

        Preview Window
```

Mapping Test Window

Information Window

Keyboard Controls

Workbench Options

Shell Options

## 1.9   Quick Font Installation

Quick Instructions

To install truetype fonts, enter a source directory (where the
ttf files are located) in the source directory box. Clicking
the 'set' button to the right will bring up a standard file
requester to allow you to choose the directory. A list of the
internal names of all installable truetype fonts in the source
directory will be generated in the Available Fonts list.

Set the Destination Drawer in a similar fashion.

To install a single font, click on it in the Available Fonts
list. If you desire, you can change the name in the Font Name
box. Then click the 'Install' button.

To install all of the fonts in the Available Fonts list, click
on the 'Install All' button. All fonts installed this way
will get default names and options.

Please note that ttfmanager will OVERWRITE the .font and .otag
files if they already exist, thus installing two fonts with
the name 'CoolNewFont' will result in only one available font.

## 1.10   ttfmanager Main Window

ttfmanager Main Window

ttfmanager features a multi-window interface, providing only
as much as you decide you want to see. The main window provides
the most basic functions, selecting and installing truetype
fonts. Closing the main window exits ttfmanager.

Available Fonts

 Shows all the available fonts in the Font Source directory. To
 select a font for further processing, click on it in this list.

Font Source

 The location of the actual font file. The Set button will bring
 up a standard file requester, from which a suitable directory
 can be chosen. This gadget can also accept a wildcard pattern to
 match only specific files. For example, CD0:A#?.ttf will restrict
 the files shown in the Available Fonts list to only those fonts
 where the filename starts with 'A' and ends in ".ttf".

  See
                Workbench Options
                 to set a default value for this field.


Font Destination

 The directory into which the .font and .otag files which allow
 a font to be used by the system will be placed by Install. The
 Set button allows choosing this from a standard file requestor.

  See
                Workbench Options
                 to set a default value for this field.


New Font Name

 ttfmanager will fabricate a name for a font from information
 within the font file itself, and place it here. To change the
 name before installing, overtype it here.


Install

 Installs the currently selected truetype font as an Amiga font
 named as specified in New Font Name, in the directory specified
 by Font Destination, using any options specified in the options
 window.


Install All

 Installs all the fonts in the Available Fonts list into the Font
 Destination directory. All fonts will be given default names and
 options.


Preview

 Opens the
                Preview
                 window.


Info

Opens the

Font Information
window.

Options

Opens the

Options
window.

The message area across the bottom of the window displays any
status or error messages.

## 1.11   ttfmanager Options Window

ttfmanager Options Window

The options window provides access to virtually all of the
information used in creation of the .otag file which describes
a font to the system. Most of the time, the default settings
should be adequate, but for 'problem' fonts, changing some of
these settings may be required.

Encodings

 Displays all encoding tables available in the currently
 selected font. ttfmanager will attempt to identify an
 appropriate unicode table. If the chosen table is incorrect,
 click on the desired one to change it.

Selected/Raw/Offset

 If desired, you may choose to bypass the encoding table and
 access the glyphs of a font directly. The raw option maps
 glyphs in order, beginning at the specified Offset, to
 ASCII printable characters only (values from 33 to 126.) This
 is intended for use with dingbat and clipart fonts, or for
 accessing otherwise unreachable glyphs in large fonts for
 artistic purposes only.

Code Page/Use/Set

 This cluster of gadgets allows a specific codepage to be
 associated with a specific font. In the string entry gadget,
 specify the full path of the desired codepage definition file.
 The Set button brings up a standard file requestor which can
 be used to select a file. For more information on the codepage
 definition format, see

ttfcp
.

The Use gadget turns the font specific codepage on and off.
If on, the needed codepage information is stored directly in
the .otag file created by an Install operation, so the source
codepage file can be moved or deleted after the install with
no impact on the operation of the fonts. The .otag codepage,
if specified, overrides the default one, either built-in or
set with ttfcp.

Note that these options are NOT reset by selecting a new face
from the Available Fonts list, so, by setting the codepage
once, a series of fonts can be installed using it.


New Font Name

This is a second copy of the field of the same name on the

                main window
                , placed here for convenience sake. Note that
you can create multiple Amiga fonts from a single truetype
file, each with its own options, but remember to give each
a unique name here. (It is strongly recommended to press
the 'enter' key to complete the entry of a new name.)


Space Width

 Controls the advance width used for the space character.
 This also serves as the advance width for all glyphs if
 the font is declared as Fixed Width.

Mapping

 The following attributes are used by the diskfont.library, or
 similar font mappers, to determine the attributes of the outline
 font. The default setting are usually adequate, but if you feel
 that ttfmanager has misidentified a font, they may be changed.

 The quality of the default settings is directly related to the
 quality of the information provided in the font, which in turn
 depends on the designer's attention to detail.

 Slant Style - Controls Italic attribute
 Weight      - Controls the Bold attribute
 Width       - Controls the Extended (wide) attribute
 Serif       - Seems to be a comment(?)
 Spacing     - Controls Monospaced (fixed pitch) attribute


Sizes

 A list of sizes which will be listed for this font in the
 system available fonts list. Use the Add, Del, and size
 entry gadget to modify this list. See
                Workbench Options
                 to

change the defaults used here.


Metric Source

 Perhaps the most visually significant switch on this screen.
 A truetype font typically contains no fewer than four distinct
 sets of information indicating the amount of space glyphs in
 the font will occupy. This information is critical to the
 proper operation of ttf.library, especially when used through
 the diskfont.library mechanisms. Unfortunately, these numbers
 are inconsistently used by different programs and designers.

 By default, ttf.library will fit a fonts entire bounding box
 into the amiga sized box, which for screen fonts is typically
 'size' pixels high. If a font has qlyphs (even ones you may not
 use) that extend the bounding box significantly from the needs
 of the more typical glyphs, the resulting font may be much too
 small. For this problem, the following options are available:

  Bound Box – uses the global font bounding box as the basis
              for size calculations. This is the default, and
              is correct for all but a few fonts. This is the
              most consistently reliable metric found in a
              real world sample of hundreds of fonts.

  Raw EM    – uses the em size directly This is equivalent to
              the previous version's 'Stretch' option. It can
         give a giant boost to the visual size, but also
              can create significant problems. For uses that
              do not use diskfont.library, like word processors
              such as final writer, this can also be useful.

  Mac       – uses Ascender and Descender values from the
              horizontal header. This metric is most often
              used by Apple systems.

  Typo      – uses supposedly 'system independent' typographic
              values from the OS/2 table. The definition of
              this value seems to have a varied interpretation
              and is missing, or identical to the usWin value
              below in many fonts.

  usWin     – uses the usWindow values from the OS/2 table.
              This value is a bounding box for latin-1 glyphs.
              It is also subject to wild variation in free and
              shareware fonts.

  Custom    – uses the values entered in the yMax and yMin gadgets
              for calculations. These values are in font units,
              and may be directly entered, or generated using
              the Calculate button. The calculated value is the
              highest ascent and lowest decent from only the
              glyphs used by the active code page (either as
              specified for this font, or as system default.)

 Note that Preview will show the chosen size, but it may not

properly illustrate any imperfections which may result. In
particular, if the height is too small, a jumping baseline,
where some glyphs sit some varying distance above or below
the expected baseline may result. Choose wisely.

For the curious, the shifting baselines are the result of
the diskfont.library trying to fit the glyphs into the pixel
height box. If the ascent is too high, the bit map is pushed
down, resulting in a low baseline. If the descent is too
low, the bit map is moved up. Interestingly, ttf.library
does not explicitly tell diskfont.library how to locate the
baseline, as it infers it instead from the glyph descriptions.

Test
 Opens the
                Mapping Test
                 window.

Info

 Opens the
                Font Information
                 window.

Preview

 Opens the
                Preview
                 window.

Install

 Installs the currently selected truetype font as an Amiga font
 named as specified in New Font Name, in the directory specified
 by Font Destination (on the
                main window
                ) using any specified
 options.

The message area across the bottom of the window displays any
status or error messages.

## 1.12   ttfmanager Preview Window

                ttfmanager Preview Window

This optional window will display a sample of the currently
selected font. Resizing the window will redisplay the sample
in a size determined by the new window size.

See
                  Workbench Options
                   to set the default string used for the preview.


Preview also has several
                  keyboard control
                  options:

ESC or DEL     - clears the current preview string
Backspace      - remove the last character from the preview string
any character - adds the character to the preview string
Cursor Left   - Apply kerning to preview display
Cursor Right  - Turn off kerning in preview display


## 1.13   ttfmanager Keyboard Controls

ttfmanager Keyboard Controls

The following keys are available in any ttfmanager window:

Cursor Down    - Select the next font from the Available Fonts list
Cursor Up      - Select the previous font from the Available Fonts list


## 1.14   ttfmanager Mapping Test Window

                  ttfmanager Mapping Test Window

This optional window displays the font on a 32 by 8 grid, applying
all codepage and encoding selections made on the
                  Options
                   window.
The grid is square and can only be changed in size by a
                  tooltype
                  setting. Some fonts which are wider than tall will look rather bad
in this window. For more typical character fonts, however, it will
aid in identifying non-defined positions in the character map, as
well as other mapping anomalies.

The default
                  MAPGRID
                   size is 18 pixels, which will display on any
hires workbench. If the window size needed to place the 32x8 grid
at the specified MAPGRID size is larger than the actual display,
it will be truncated to the top left corner that will fit.


## 1.15   ttfmanager Font Information Window

```
ttfmanager Font Information Window
```

This optional window shows Name and Copyright information, as
well as various statistics and metrics from the currently
selected font.

1.16   ttfmanager Workbench options

```
ttfmanager Workbench options
```

```
ttfmanager supports the following ToolTypes in its icon:
```

```
SOURCE
  Specifies the initial Font Source in which to look for truetype
  font files. Example: SOURCE=Work:myfonts
```

```
FONTS
  Specifies the initial Font Destination, a directory in which the
  .otag and .font files for an installed truetype font will be
  placed. Example: FONTS=Sys:ttfonts
```

```
SIZES
  Specifies the default Sizes as seen on the options page. This
  specifies the font sizes which will be reported in the system
  AvailFonts list for use in font requestors. Up to 20 sizes may
  be specified in a comma separated list. Example:
  SIZES=10,12,16,24,32,48,64,72,98,122
```

```
PREVIEW
  Specifies the string used for the Preview option Example:
  PREVIEW=Every Good Boy Does Fine
```

```
MAPGRID
  Specifies the pixel size for the Mapping Test option. Example:
  MAPGRID=20
```

Additional arguments, such as shift-clicked icons, will be used
as the Font Source. For example, select a drawer on the workbench,
hold down the shift key and double click the ttfmanager icon,
and ttfmanager will use the drawer as the initial Font Source.
Also, a project icon associated with a truetype font file could
refer to ttfmanager as its default tool, invoking ttfmanager
when clicked, which would set the Font Source to the single file.
Note that the project file does not have to be a font file. Any
ToolTypes in the project icon are evaluated as well.

1.17   ttfmanager Shell options

```
ttfmanager Shell options
```

```
The current command line interface will change soon
```

```
FROM/A,TO/A,SIZES/F
```

```
Example: ttfmanager work:coolnewfonts fonts: sizes=12,24,36,48
```

1.18   Using ttfinstall

```
                ttfinstall
```

```
Note:
                ttfmanager
                 is now available and will perform font
installation from the Workbench environment. This tool is
included for those who want CLI and scripting power. For
an overview of font installation, including differences
from the system intellifont program, see
                ttfmanager
                .
```

```
The command line for installing a truetype font looks like
the following:
```

```
  ttfinstall fontfile.ttf sys:fonts
```

```
Note that 'fontfile.ttf' is the actual file to be installed, and
'sys:fonts' can be any existing directory. In addition, a file
name pattern can be specified for 'fontfile.ttf', allowing the
installation of several fonts at once. For example:
```

```
  ttfinstall cd0:funky/#?.ttf work:ttfonts
```

```
Would install all the font found in the cd0:funky drawer to the
ttfonts drawer on volume work:
```

```
Once a truetype font is installed on the fonts: path, it should
be possible to select that font from any decent font requester.
```

```
Please note that ttfinstall will OVERWRITE the .font and .otag
files if they already exist, thus installing two fonts with
the name 'CoolNewFont' will result in only one available font.
It is possible to rename both the .font and .otag files once
they have been created, so multiple fonts with the same internal
name can be used if you work at it. ttfmanager has an option
to change the name when installing individual fonts.
```

```
Note: switches EXACT, TYPO, and DESIGN are no longer valid.
```

1.19   Using ttflist

```
ttflist
```

ttflist works a lot like the normal system 'list' command, but
is designed to display readable names along with the file name.
Also, it will optionally provide a great deal of perhaps useful
information from a font file. The command line would look like:

  ttflist [pattern] [all] [verbose]

where:

 pattern is an optional amigados file pattern or
          drawer name. By default it is "#?.(ttf|ttc)"

 all   is an optional switch to list matching drawer
          contents as well. Note that the pattern specified
          must match the drawer names to be searched. When
          all is specified the default pattern is "#?".

 verbose gives a lengthy, and probably ignorable, report for
          each font found.

example: ttflist cd0: all

1.20   Using ttfcp and alternate codepages

                    ttfcp and alternate codepages

ttfcp is a command line tool to install codepage tables that
allow the use of ttf.library on systems which do not use the
default ECMA Latin-1 character set. ttfcp uses simple text files
to describe the mapping of the 256 possible Amiga character set
positions to any unicode characters. The text file format is
quite simple:

- Each mapping line contains two numbers separated by space
  and/or tab characters. These numbers may be in decimal,
  octal, or hexadecimal. (decimal numbers must start with
  a digit 1-9, octal with a "0", and hexidecimal with "0x")
  The first number is the Amiga character set position being
  mapped, and must be between 0 and 255. The second number
  is the unicode character to be assigned to the position.
- A "#" character begins a comment
- any line without two valid numbers as the first non space
  items are ignored without warning
- any character positions not mapped will be set to 0

Sample mapping file lines: -------------------------------
full line comment
0x20 0x0020
32 32   # functionally identical to the previous line
325 0x0042 # this line would be ignored (index>255)
49 72659 # this line would be ignored (unicode>0xFFFF)
0xA1 0x040E # CYRILLIC CAPITAL LETTER SHORT U
0xB6 0x0386 # GREEK CAPITAL LETTER ALPHA WITH TONOS

Note: A variety of mapping tables suitable for use with ttfcp
      are available at ftp.unicode.org in Public/MAPPINGS.

ttfcp translates the mapping table into a more efficient form
for use by the library. The command line is:

  ttfcp mapfile [ENV] [TO outfile]

where:

 mapfile is the required mapping file as described above.

 ENV   is a switch which will cause the translated table
          to be placed in the "ttfcodepage" environment
          variable. (And copied to ENVARC:)

 TO file allows writing the translated table to "file"
          (This option is not very useful, as currently
          nothing else uses the created files.)

example: ttfcp 8859-2.TXT ENV

If neither ENV or TO options are specified, no translated
table is written, but an normalized ascii version of the
input is displayed.

To enable alternate codepage support:

1) obtain or create a mapping file
2) use ttfcp with the ENV option to install the mapping file

These steps need only be performed once. The next use of the
ttf.library will begin using the new codepage mapping. To
remove codepage mapping delete the file "ttfcodepage" from
ENV: and ENVARC:.

For codepage support to work correctly, the fonts used must
contain the glyphs needed to represent the characters of
the chosen codepage. Also, the font must have a suitable
unicode encoding table. You can check the results of the
code page and encoding selections with the
                Test
                 button on
the
                Options
                 page.

By default, ttf.library will use straight Unicode translations.
The codepage definitions only affect the first 256 character
positions. If you need full unicode support, either the system
codepage or the font specific codepage will need to provide a
straight conversion (i.e. ISO-8859-1).

1.21 Using Bitline

Bitline

Bitline is a small shell utility which allows you to convert
Amiga outline fonts to Amiga Bitmap fonts. Bitline works with
the diskfont.library, so any font format your system can use
can be converted.

Bitline is an adaptation of Bitline 0.5 by Georg Steger, and
is being distributed with permission from Georg. Thanks Georg!

Bitline can only be used from shell. It has the following
argument template:

```
    FONTNAME/A,SIZES/N/M/A
```

FONTNAME: Path of source outline font. You must specify the
          font contents file (the one that ends with '.font'.)
          The ".font" suffix is optional, and the entire FONTS:
          assign will be searched if no directory is specified.
          example: "fonts:foboz.font"

SIZES   : One or more font sizes you want to have created.

For example,

```
    Bitline foboz 10 20 30 40
```

would creates bitmaps at 10, 20, 30 and 40 sizes from the font
'Fonts:foboz.font'

Warning – this is the first public release of this version of
Bitline. Do use caution, and please report any problems.

Legal Note – fonts are usually copyrighted entities, and
the use thereof may be limited by license or other agreements.
It is the users responsibility to verify that the use of this
utility does not violate the terms of such license.

1.22 Using ftview

ftview

ftview is a truetype font display program from the FreeType
project test suite. Its command line format is:

```
  ftview [-g] [-r res] pointsize fontfile ...
```

where

```
-g          is an optional grey scale (smoothing) rendering.
            Smoothing looks very wonderful, but is not available
            through ttf.library. Also note that ftview opens on
            the default public screen, and if that screen does
            not have sufficient available or matching pen colors,
            it may look worse. The -g option uses five distinct
            grey levels.

-r res      specifies an optional resolution to use in rendering
            the qlyphs of the font. 'res' is in Dots Per Inch.
            Note this doesn't change the display resolution
            only the value used for font rendering.

pointsize is a required argument specifying the point size
            to display.

fontfile   is one or more truetype font files to view
```

Example: ftview 30 flyingp.ttf

While the font is displayed, options to change many rendering
characteristics are available by keystroke or standard amiga
menu selections.

1.23   Limits and known problems

Limits, warnings

Portions of this software are new, and have had minimal testing.
The performance of this software has been heavily monitored on
the development machine with enforcer, mungwall, poolwatch and
other tools, however, it is still unproven. Please use caution,
and use at your own risk.

A few features of the bullet library standard have not yet been
implemented (algorithmic emboldening) as they are not required
for proper font operation in many circumstances. Programs that
depend on these unimplemented features, or on undocumented
features of the bullet.library may not work as expected with
ttf.library.

Application Notes

PersonalPaint

 Without 'Keep Ratio' selected, there are several quirks
 noted using ttf.library with the PPaint Vector Text tool.
 Specifically, rotations approaching 90 degree angles do
 not display. Also, when the rotation angle and the natural
 diagonal angle of the text box are quite different (such
 as a 45 degree rotation in a short, wide box,) the aspect
 of the glyphs can become distorted. No specific cause in
 ttf.library has been identified.

 For the technically curious, the vector text tool specifies

a fixed point size and the sin and cos for the rotation
angle, and appears to attempt to find a perfect fit within
the box by changing the dpi. Whatever formula it is using
is not satisfied by the ttf returned values, and, in the
case of the 90 degree rotations, the dpi are modulated to
totally unworkable values (0x3FFF by 4, in one case.)

FinalWriter

Under some circumstances 'Justify All' formatting will be
broken when printing. It appears that the justification
is based on width and advance information from separate
resolutions, with the width data taken from a small dpi,
such as the screen, and the character advance info from
the printing dpi. TrueType font hinting can change the
width of a glyph to visually compensate at low resolutions.
FinalWriter, unlike several other high end packages, does
not ask for the bullet style width list, which would be
resolution independent.

1.24   Recent Change History

                    Changes in v0.8.2

  Fixes problems with diacritic accents in some fonts

Changes in v0.8.1

  Many fonts which previously failed to work (missing OS/2
    table, common in fonts from Mac sources) should now work.
  Unicode support fixed.
  SpaceWidth entry gadget added to ttf.manager options.
  CPU specific archives available for 68k series processors.

Changes in v0.8.0

  Full unicode support now enabled
  Many fonts which previously reported 'broken font - invalid
    reference' should now work. Dig out those broken fonts!
  Eliminate some needless memory copying in font loading.
  New mapping test window available on the Options window.
  Encoding selection will choose Win Unicode over Apple; fixes
    quite a few screwy fonts with bad Apple Unicode tables.
  Preview window no longer limited to 640x400.

Changes in v0.7.9

  add support for shear and rotation

Changes in v0.7.8

  fix buggy yMin gadget
  fix crash when specifying a random file as codepage definition
  add
           bit map
            creation utility

Changes in v0.7.7

  adds options to select various or custom metrics to determine
    the height factor used to scale fonts to the proper size.
    In particular, this should make the M$ euro fonts usable
    (again). See
           Options
            and
           FAQ
            for more information.
  variety of small bug fixes

Changes in v0.7.5

  eliminate potential crash when a font file could not be accessed
  kerning support enabled
    note that only one level of kerning is supported, and bullet
    api calls specifying either 'Design' or 'Text' kerning will
    result in the same kerning value.
  corrected width list calculation (apparently broken since 0.7)
  ttfmanager adds some keyboard support, especially in preview

Changes in v0.7.3

  should now support infamous win symbol encoding
  added 'stretch' option for larger glyphs at small sizes
  applied the most current freetype patches

Changes in v0.7.2

  plugged a nasty memory leak in the library
  font specific codepage support enabled

Changes in v0.7.1

  ttfmanager – several cosmetic changes
  ttflist – now reports some kerning table info
  ttf.library
   – fixed a problem with baseline shift on some characters
    at smaller sizes (especially with diacritical marks)
   – some changes in advance width calculations

Changes in v0.7

  Reinstall Warning – some important things have changed in
  the font installation, and all fonts installed with any

previous version must be re-installed. (Sorry, but this
should be the last time.)

ttfmanager

 – preview, font information and options windows added
 – several changes in the .otag files produced by installing,
   adding a complete set of font attribute mappings, fixing
   the space width calculation.

ttf.library

 – More speed! Typically 1/3 faster on a 040/25MHz.

 – The library now handles internally the correction between
   Amiga font size and actual em size, instead of relying on
   diskfont.library to do the job. This opens the way for
   several future enhancements.

Changes in v0.6.3

  ttf.library now supports a system wide alternate codepage

  Added ttfcp, a codepage installation tool.

Changes in v0.6.2

  Only the ttf.library has changed in this version.

  The quest for perfection continues – the character advance
  calculations are virtually identical to that common M$ OS
  engine. An issue with non-glyph widths (i.e. the space
  character will be fixed in the next installer version.

  Source of mystery crashes (of other tasks) found and fixed.

Changes in v0.6

  Re-examined the character advance calculations to improve
  spacing at smaller sizes.

  Modified the font install routines to calculate the extremes
  of the ascent and descent by default. Problem fonts (baseline
  shifts or clipped glyphs) should be re-installed with the
  new ttfmanager or ttfinstall version.

  Added a fake version number to ttf.library for WordWorth.

Changes in v0.5

  Added ttfmanager and Install-ttflib.
  Re-examined linkages, often resulting in smaller file sizes.

Changes in v0.4

  WidthLists are now supported.

Subtle changes to returns of non-existent glyphs to be more
compatible with bullet, and make smaller (memory wise) fonts
when called by diskfont.library.

ttfinstall - correct advance width problem with monospaced fonts

Changes in v0.3

Sets many more tags in otag file to make some programs much
happier (FinalWriter, to name one.) As a result, ALL FONTS
installed with v0.2 or earlier MUST BE RE-INSTALLED. Sorry,
but the results are usually worth it.

Mimic bullet.library in handling of glyphless codes, such as
a space.

Widths are calculated more appropriately, resulting in better
display under many conditions.

Several other peculiar, but un-enumerated, bugs were squashed.

1.25 The future of ttf.library

Future

Still many improvements in the works for ttfmanager.

Add family support, allowing different font files to be used
for bold and italic requests automatically.

Add library support for algorithmic emboldening

Examining possibilities of using the forthcoming
FreeType 2 engine.

1.26 Frequently Asked Question

FAQ

Can/will it get faster?

Well, you should have seen it in the early days!

Actually, I do keep looking at ways to speed things up,
but TrueType rendering is a computationally intensive
task, and it places heavy demands on slower processors.
Faster processors and plenty of memory help a lot. I'll
keep at it if you'll upgrade ;)

Why are the fonts so small or have a so much space underneath?

  Microsoft's core fonts, pan-european versions, have a very
  unusual characteristic which has made the display less than
  pleasing under ttf.library version 0.7 and above. First,
  thanks go to Pavel Cizek for having the patience to help
  me see and understand the problem.

  The fonts have a run of line drawing glyphs which distort
  the global bounding box (a virtual box which can surround
  ANY glyph in the font.) Since version 0.7, this size has
  been the basis for translating the font into Amiga sizes.
  It should be noted, that the decision to use this metric
  for the basis was made for several reasons, one being that
  it is the most reliable metric above all others in a wide
  variety of fonts examined in great detail. This unusual
  feature escaped the testing. Metropol, which I used for
  many Latin-2 tests, does not have any such problem. Sorry.

  To allow these fonts, or any others which may have similar
  problems, there are now several new options available. In
  fact, this is the main reason for V0.7.7.

  A quick guide to installing one of these fonts:

  Select the desired font. Open the options window. Select a
  metric source of 'Custom'. Make sure the correct code page
  is set (either as the system default, or for this font.)
  Click calculate, and two new numbers should be displayed
  for yMax and yMin. Install.

  Repeat these steps, including the 'Calculate' step, for
  each font which displays similar symptoms.

Can we make bitmaps, please?

  A very frequently requested enhancement is for bitmap font
  generation. A version of Georg Steger's
                Bitline
                  is now
  included.

How about a PPC version?

  A PPC version creates some additional concerns, but it is
  being considered.

Why are the fonts so tiny? 8 point soandso.font looks
bad. On a PC I can use 7 point foboz, but it is just
little squiggles on my amiga. Etc, etc.

  A quick lesson of fonts and terms is in order. In more
  traditional typography, a point is a standard unit of

measure, approximately 1/72 of an inch. A point size
refers to the size of an imaginary box surrounding the
letter 'M', know as the em square. The size of this box
in 1/72 inch units is the point size. When Windoze defined
its version of point size, it invented a new concept,
the virtual display inch (I'm not kidding!) which is
"approximately 30 to 40 percent larger" If you consider
that the declared resolution of a typical Win display is
96DPI square, you'll find that the math works out to the
em square point measurement is approximately the pixel
height count of the letter M.

The Amiga however uses 'points' to specify the entire
height of the font, not just the letter M. There is no
provision, through diskfont.library for a glyph to fall
outside of a 'point' high bitmap. As a consequence, a
box which will hold the 'M' won't hold a 'Ç' or an 'È'
or any other characters. The needed room must come from
somewhere, and this means that the sizes must be scaled
to fit. Since ttf.library is at the bottom of the chain,
the apparently inflated point size must be specified. As
a result, requests for 8 point glyphs will usually result
in total garbage, as 3-4 'points' get used for any needed
ascenders and descenders, leaving little to reflect the
characters. 10 is sometimes recognizeable, and 12 starts
to be useful for most fonts.

To match an Amiga point size exactly to those of another
platform, examine the highest ascender, lowest descender
and the letter 'M' on the other platform. The letter 'M'
should be approximately the point size pixels tall. Add
the additional pixel heights of the high and low extremes
to that point height, and you should have a close value
for a corresponding Amiga size.

What can I do to make my browser look like a PC?

  You can match the font sizes used by default in MSIE
  or Mozilla on a Winbox. There will be differences in
  the layout by the browser, but the letters in the words
  could be made to match in many ways. The biggest problem
  is in finding the correct sizes to specify, approximate
  values of which are presented below. You will need to
  obtain the TimeNewRoman and CourierNew font families.
  I leave all legal, ethical and tactical aspects of this
  to the individual. (These are excellent fonts, and they
  are available without cost, but I have yet to find an
  Amiga only way to obtain some versions of them, and
  remain entirely within the EULA. The win3.1 versions
  can be extracted with unzip, but not all faces, nor
  the latest versions are available in this form.)

  Approximate matching fonts/sizes are as follows:

      H1   TimesNewRomanBold/36
      H2   TimesNewRomanBold/28

```
      H3          TimesNewRomanBold/20
      H4          TimesNewRomanBold/18
      H5          TimesNewRomanBold/14
      H6          TimesNewRomanBold/12

      FONT SIZE=1 TimesNewRoman/12
      FONT SIZE=2 TimesNewRoman/15
      FONT SIZE=3 TimesNewRoman/18
      FONT SIZE=4 TimesNewRoman/20
      FONT SIZE=5 TimesNewRoman/27
      FONT SIZE=6 TimesNewRoman/35
      FONT SIZE=7 TimesNewRoman/53

      Preformated CourierNew/15
```

  Another potentially useful browser trick is to set a codepage
  to eliminate the dreaded no such character box that appears
  often where quote marks and apostrophes should appear.
  Although I think that any web author that allows such rot to
  appear on a page (Including Amiga, Inc!) should at least be
  publicly chastised, the fact remains that these artifacts are
  widespread. It seems that M$ 'improved' (in its opinion) the
  ISO 8859-1 standard to create the 1252 codepage. (For more
  info, see ftp.unicode.org. for mapping tables.) It seems that
  range from 0x80 to 0x9F, which is non-printing in 8859-1 (a
  REAL standard) is heavily and arbitrarily populated in CP1252,
  adding left and right single and double quotes, the new Euro
  currency symbol, and other things.

What is this strange version number on ttf.library?
  WordWorth wants to open Version 2, and ttf.library is not
  even to version 1.0 yet. This is a bug in WordWorth. It
  has no reason to specify a library version at all when
  opening the library. This is likely a holdover from a very
  early bullet.library, but it is still WRONG! For the time
  being, I have faked the version in the library header to
  pass this test. Version 0.7 appears as version 10.7 to the
  system. Any future versions will continue to be offset by
  10, as long as required.