diskfont

COLLABORATORS			
	TITLE : diskfont		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 19, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

diskfont

1.1 diskfont.doc

```
    AvailFonts()
    DisposeFontContents()
    NewFontContents()
    NewScaledDiskFont()
    OpenDiskFont()
```

1.2 diskfont.library/AvailFonts

```
 NAME
AvailFonts -- Inquire available memory & disk fonts.

 SYNOPSIS
error = AvailFonts(buffer, bufBytes, flags);
                   A0      D0        D1

LONG AvailFonts( struct AvailFontsHeader *buffer, LONG bufBytes
  ULONG flags );

 FUNCTION
AvailFonts fills a user supplied buffer with the structure,
described below, that contains information about all the
fonts available in memory and/or on disk.  Those fonts
available on disk need to be loaded into memory and opened
via OpenDiskFont, those already in memory are accessed via
OpenFont.  The TextAttr structure required by the open calls
is part of the information AvailFonts supplies.

When AvailFonts fails, it returns the number of extra bytes
it needed to complete the command.  Add this number to your
current buffer size, allocate a new buffer, and try again.

 INPUTS
buffer - memory to be filled with struct AvailFontsHeader
  followed by an array of AvailFonts elements, which
  contains entries for the available fonts and their
```

```
  names.

bufBytes - the number of bytes in the buffer
flags - AFF_MEMORY is set to search memory for fonts to fill
  the structure, AFF_DISK is set to search the disk for
  fonts to fill the structure.  AFF_SCALED is set to
  not filter out memory fonts that are not designed.
  AFF_BITMAP is set to filter out fonts that are not
  stored in Amiga font format, i.e. to filter out
  outline fonts.  Any combination may be specified.
  AFF_TAGGED is set to fill the buffer with TAvailFonts
  elements instead of AvailFonts elements.

 RESULTS
buffer - filled with struct AvailFontsHeader followed by the
  [T]AvailFonts elements, There will be duplicate entries
  for fonts found both in memory and on disk, differing
  only by type.  The existance of a disk font in the
  buffer indicates that it exists as an entry in a font
  contents file -- the underlying font file has not been
  checked for validity, thus an OpenDiskFont of it may
  fail.
error - if non-zero, this indicates the number of bytes needed
  for AvailFonts in addition to those supplied.  Thus
  structure elements were not returned because of
  insufficient bufBytes.

 EXAMPLE
int afShortage, afSize;
struct AvailFontsHeader *afh;

...

afSize = 400;
do {
    afh = (struct AvailFontsHeader *) AllocMem(afSize, 0);
    if (afh) {
        afShortage = AvailFonts(afh, afSize, AFF_MEMORY|AFF_DISK);
        if (afShortage) {
            FreeMem(afh, afSize);
            afSize += afShortage;
        }
    }
    else {
        fail("AllocMem of AvailFonts buffer afh failed\n");
        break;
    }
}
    while (afShortage);

/*
 * if (afh) non-zero here, then:
 * 1. it points to a valid AvailFontsHeader
 * 2. it must have FreeMem(afh, afSize) called for it after use
 */
```

1.3 diskfont.library/DisposeFontContents

```
 NAME
DisposeFontContents -- Free the result from NewFontContents. (V34)

 SYNOPSIS
DisposeFontContents(fontContentsHeader)
        A1

VOID DisposeFontContents( struct FontContentsHeader * );

 FUNCTION
This function frees the array of FontContents entries
returned by NewFontContents.

 INPUTS
fontContentsHeader - a struct FontContentsHeader pointer
    returned by NewFontContents.

 EXCEPTIONS
This command was first made available as of version 34.

A fontContentsHeader other than one acquired by a call
NewFontContents will crash.

 SEE ALSO
NewFontContents to get structure freed here.
```

1.4 diskfont.library/NewFontContents

```
 NAME
NewFontContents -- Create a FontContents image for a font. (V34)

 SYNOPSIS
fontContentsHeader = NewFontContents(fontsLock,fontName)
     D0                                A0        A1

struct FontContentsHeader *NewFontContents( BPTR, char * );

 FUNCTION
This function creates a new array of FontContents entries
that describe all the fonts associated with the fontName,
specifically, all those in the font directory whose name
is that of the font sans the ".font" suffix.

 INPUTS
fontsLock - a DOS lock on the FONTS: directory (or other
    directory where the font contents file and associated
    font directory resides).
fontName - the font name, with the ".font" suffix, which
    is also the name of the font contents file.

 RESULT
fontContentsHeader - a struct FontContentsHeader pointer.
```

```
 EXCEPTIONS
This command was first made available as of version 34.

D0 is zero if the fontName is does not have a ".font" suffix,
if the fontName is too long, if a DOS error occurred, or if
memory could not be allocated for the fontContentsHeader.

 SEE ALSO
DisposeFontContents to free the structure acquired here.
```

1.5 diskfont.library/NewScaledDiskFont

```
 NAME
NewScaledDiskFont -- Create a DiskFont scaled from another. (V36)

 SYNOPSIS
header = NewScaledDiskFont(srcFont, destTextAttr)
D0                         A0       A1

struct DiskFontHeader *NewScaledDiskFont( struct TextFont *,
  struct TTextAttr * );

 INPUTS
srcFont - the font from which the scaled font is to be
    constructed.
destTextAttr - the desired attributes for the new scaled
    font.  This may be a structure of type TextAttr or
    TTextAttr.

 RESULT
header - a pointer to a DiskFontHeader structure.  This is not
  being managed by the diskfont.library, however.

 NOTES
o   This function may use the blitter.
o   Fonts containing characters that render wholly outside
    the character advance cell are currently not scalable.
o   The font, and memory allocated for the scaled font can
    can be freed by calling StripFont() on the font,
    and then calling UnLoadSeg() on the segment created
    by this function.

    Both the TextFont structure, and segment pointer are contained
    within the DiskFontHeader struct.  The DiskFontHeader structure
    will also be freed as part of the UnLoadSeg() call.
    StripFont() is a new graphics.library call as of V36.
```

1.6 diskfont.library/OpenDiskFont

```
 NAME
     OpenDiskFont - load and get a pointer to a disk font.
```

```
SYNOPSIS
    font = OpenDiskFont(textAttr)
    D0                  A0

FUNCTION
    This function finds the font with the specified textAttr on
    disk, loads it into memory, and returns a pointer to the font
    that can be used in subsequent SetFont and CloseFont calls.
    It is important to match this call with a corresponding
    CloseFont call for effective management of font memory.

    If the font is already in memory, the copy in memory is used.
    The disk copy is not reloaded.

INPUTS
    textAttr - a TextAttr structure that describes the text font
            attributes desired.

RESULTS
    D0 is zero if the desired font cannot be found.

NOTES
    As of V36, OpenDiskFont() will automatically attempt to
    construct a font for you if:

            You have requested a font size which does not exist
            as a designed font, and

            You have not set the DESIGNED bit in the ta_Flags
            field of the TextAttr, or TTextAttr struct.

    Constructed fonts are created by scaling a designed font.
    A designed font is one which typically resides on disk,
    or in ROM (e.g., a font which has been designed by hand
    using a drawing tool).  Designed fonts generally look better
    than fonts constructed by the font scaler, but designed
    fonts also require disk space for each font size.

    Always set the DESIGNED bit if you do not want constructed fonts,
    or use AvailFonts() to find out which font sizes already exist.

    As of V37 the diskfont.library supported built-in outline
    fonts.  Then in V38 the outline font engine was moved to
    a new library, "bullet.library."

BUGS
    This routine will not work well with font names whose file
    name components are longer than the maximum allowed
    (30 characters).
```