

Supporting Font Directory

Third party applications can interact with Font Directory to provide a 'seamless' user interface to users (with respect to Outline Fonts).

- (1) When the fonts made available by Font Directory change, a WIMP message is issued. Applications should react to this message by re-scanning the fontlist and updating any menus, windows, documents, etc.
- (2) Font Directory provides a WIMP message protocol that allows applications to request that any font, stored on the font filing system, be installed in the fontlist.

This means that when an application loads a new document, it can request any fonts that it does not recognise before issuing tiresome 'font not found' errors.

Or an application can ensure that certain 'default' fonts are made available.

- (3) When the user drops a document onto the Font Directory window, it will be scanned and any fonts used will be made available to other desktop applications.

This facility is provided on three levels:

A. Font Directory performs a pattern search for each font name. If the name is found in the document, the font will be made available. This procedure can be slow and can result in fonts being installed when they are not actually used.

B. An extension mechanism is provided. Extension routines can be written that support a particular file type and inform Font Directory of the fonts used in that document. This method improves the performance of this facility, but may be prone to file type revisions.

C. A WIMP message protocol is provided that allows resident applications to scan a document and inform Font Directory of the fonts used by that document.

By applying these three levels in reverse order, Font Directory can support a wide range of file formats whilst securing performance advantages for supported file types.

FontFS Message Protocols

(1) Fonts Changed

This message is issued whenever the fonts made available by FontFS change. If an application receives this message, it should rescan the font list and redraw any documents, menus, etc.

(2) Font Request

This message is aimed at eliminating tiresome ‘font not found’ errors.

When an application loads one of its own documents, it can request any font used in the document that is not already available in the fontlist.

(3) Third Party File Support (or 3pfs)

This message is aimed at eliminating problems of filetype revisions & 3pfs extensions.

When Font Directory wishes to scan a document for the fonts that it contains, it will issue this message first. If an application recognises the file type, it should scan the document and return the fonts used in the document as per the Font Request message.

Events & Actions (summary)

EVENTS	ACTIONS
Fonts Changed message received	Scan FontList

Load Document	Scan Document (1) Load Document
No reply to Font Request	Scan FontList

3pfs Request message received	Scan Document (2)

Chain of events

(1) Fonts Changed

Fonts Changed message received
Application rescans fontlist and updates documents, menus, windows, etc
(Where appropriate, 'font not found' errors are reported)

(2) Font Request

User requests application to load document
Application scans document for fonts used
Application requests any fonts not available via Font Request message
Application loads document without reporting any 'font not found' errors

Either:

Font Directory receives & acknowledges Font Request message
Where possible, requested fonts are made available
Fonts Changed message issued

Or:

Application Font Request not acknowledged
Application rescans fontlist and updates document, menus, windows, etc
(Where appropriate, 'font not found' errors are reported)

(3) 3pfs Request

User requests Font Directory to scan document
Font Directory issues 3pfs Request message

Either:

Application recognises file type & acknowledges 3pfs request
Application scans document for fonts and issues appropriate Font Requests

Or:

Font Directory 3pfs Request not acknowledged
Font Directory looks for 3pfs extensions

If extension is available:

Font Directory calls extension to scan document

If extension does not recognise document version OR if extension not available:

Font Directory performs a pattern search on document

Events

Load Document

When a document is loaded by the application, it should first scan the document and request any unknown fonts and then load the document into memory without reporting any 'font not found' errors.

The application should then wait until it either (i) receives a Fonts Changed message, or (ii) does not receive an acknowledgement of its Font Request message(s).

Fonts Changed (message &82540)

This message is broadcasted by FontFS whenever the fonts made available on the font partition change. The application should scan the fontlist and redraw any documents appropriately.

Font Request (message &82541)

This message is sent when the application scans a document for the fonts (see Scan Document action). If this message is not acknowledged, the application should scan the fontlist and redraw any documents appropriately.

3pfs Request (message &82542)

This message is broadcasted by FontFS when it wishes the fonts in a particular file to be scanned. If the application recognises the file type, it should scan the document and request any fonts used as per the Load Document event.

Actions

Load Document

The application should load the document as usual without reporting any 'font not found' errors.

Scan Document

The application should scan the document for any fonts used and send an appropriate Font Request message. There are two ways of sending this message:

- (1) If scanning a document loaded into the application (recorded message sent).
 - >>> only fonts used in the document that are NOT available in the fontlist should be requested
- (2) If replying to a 3pfs Request message (user message sent).
 - >>> all fonts used in the document should be requested

Scan FontList

If the <Font\$Path> definition has changed, the application should scan the fontlist and then redraw any documents, menus, etc accordingly.

Example Implementation

```
mytype%= application's document filetype
:
fontchanged=&82540
requestfont=&82541
request3pfs=&82542
DIM block% &800
bufferize%=&400 : DIM buffer% bufferize%
oldfontpath$="" : fontpath$=""
:
REPEAT
SYS "Wimp_Poll",,block% TO reason%
CASE reason% OF
WHEN 17,18 : PROCwimp_messagereceived(block%)
ENDCASE
UNTIL FALSE
END

DEFPROCwimp_messagereceived(b%)
CASE b%!16 OF
WHEN fontchanged : PROCscan_fontlist
WHEN request3pfs : IF b%!24=mytype% THEN PROCscan_document3pfs(b%)
WHEN requestfont : IF b%!12=0 AND b%!28=0 THEN PROCscan_fontlist
ENDCASE
ENDPROC

DEFPROCscan_fontlist
fontpath$=FNread_fontpath
IF fontpath$=oldfontpath$ THEN ENDPROC
oldfontpath$=fontpath$
    scan fontlist as usual
    update & redraw menus, windows, documents, etc
    report any 'font not found' errors
ENDPROC

DEFFNread_fontpath
LOCAL length%
SYS "XOS_GSTrans", "<Font$Prefix> <Font$Path>",buffer%,bufferize%-1 TO ,,length%
IF length%>255 THEN length%=255
block% ?length%=13
=$block%
```

```
DEFPROCscan_document(b%)
b%!16=fontrequest : b%!20=0 : b%!24=mytype%
PROCscan_document4fonts(b%,18,0)
ENDPROC
```

```
DEFPROCscan_document3pfs(b%)
LOCAL taskid% : taskid%=b%!4
b%!12=b%!8 : SYS "Wimp_SendMessage",19,b%,taskid%
b%!16=requestfont : PROCscan_document4fonts(b%,17,taskid%)
ENDPROC
```

```
DEFPROCscan_document4fonts(b%,messagetype%,taskid%)
LOCAL requestmade : requestmade=FALSE
  get first font name in document
  WHILE font name found
  IF b%!20>0 OR font is not available in fontlist THEN
  b%!0=28+(LEN(font name)+4) DIV 4)*4
  b%!12=0 : $(b%+28)=font name+CHR$(0)
  SYS "Wimp_SendMessage",messagetype%,b%,taskid%
  requestmade=TRUE
  ENDIF
  get next font name in document
  ENDWHILE
```

```
IF requestmade THEN
b%!0=32 : b%!12=0 : b%!28=0
SYS "Wimp_SendMessage",messagetype%,b%,taskid%
ENDIF
ENDPROC
```

```
DEFPROCload_document
PROCscan_document(block%)
PROCreallyload_document
ENDPROC
```

```
DEFPROCreallyload_document
  load document as usual without reporting 'font not found' errors
ENDPROC
```

3pfs Extensions

An extension file is kept inside the '!FontDir.extensions' directory and is automatically loaded when !FontDir is run.

The file starts with a 4-byte word which should contain the file format that is supported. The extension code should then follow. For example:

```
        EQU  &AFF
.DrawExtension
        MOV  PC,R14
```

On Entry:

- R1 = pointer to parameter block
- R1+0 = file handle
- R1+4 = pointer to 256 byte buffer
- R1+8 = pointer to display list
- R1+12 = item flags to be set
- R1+16 = item flags to be cleared
- R1+20 = -1

All registers should be preserved.

The file to be examined is open for read only input, as per R1+0.

If the document version is recognised, R1+20 should be set to zero OTHERWISE the extension code should be exited immediately.

Each font name (null terminated) used in the document should be stored in the 256 byte buffer pointer to by R1+4 and then the FontFS_SelectFont SWI should be called.

Directory style documents

If an application's files are stored inside a directory (e.g. Impression, Squirrel, etc), an appropriate entry should be made in the '!FontDir.extensions.!dirtypes' file.

This file can be loaded into !Edit and consists of a list of filenames and filetypes:

```
!DocData D87
window DB7
```

Font Directory handles directory style documents by searching inside the directory for each of the files listed in '!dirtypes'.

If a file exists, with the correct file type, then Font Directory will scan that file to locate any fonts used in the document OTHERWISE the document is ignored.

Example DRAW file extension

```
.header
    EQUd &AFF

    STMPD R13!,{r0-r11 ,R14}
    MOV R11,R1                ; keep parameter block in R11

    MOV R0,#0                 ; mark filetype as recognised
    STR R0,[R11,#20]

    LDR R10,[R11,#4]         ; get workspace
    LDR R1,[R11,#0]         ; get file handle

    MOV R4,#40               ; start looking at objects at file offset #40
.object_loop
    BL osgbpb_word          ; get object type
    LDMCSFD R13!,{r0-r11 ,PC} ; finish if eof

    LDR R0,[R10]            ; have we found the font object?
    CMP R0,#0
    BEQ fontobject_found

    BL osgbpb_word          ; get object size
    LDMCSFD R13!,{r0-r11 ,PC} ; finish if eof
    LDR R0,[R10]            ; go onto next object
    SUB R0,R0,#8
    ADD R4,R4,R0
    B object_loop

.fontobject_found
    BL osgbpb_word          ; get object size
    LDMCSFD R13!,{r0-r11 ,PC} ; finish if eof
    LDR R9,[R10]
    SUB R9,R9,#8
    ADD R9,R9,R4

.fontloop
    ADD R4,R4,#1
    CMP R4,R9
    LDMGEFD R13!,{r0-r11 ,PC}

    BL osgbpb_string        ; read font name
    MOV R0,#0 : STRB R0,[R2,#-1]
    MOV R0,R10
    LDRB R0,[R10]           ; check font name
    CMP R0,#0               ; if last font name
    LDMEQFD R13!,{r0-r11 ,PC} ; then exit

    STMPD R13!,{r1 }        ; select font
    MOV R1,R11
    SWI "XFontFS_SelectFont"
    LDMFD R13!,{r1 }
    B fontloop

.osgbpb_word                ; read word from file
    STMPD R13!,{r14 }
    MOV R0,#3
    MOV R2,R10
    MOV R3,#4
    SWI "XOS_GBPB"
    LDMFD R13!,{pc }

.osgbpb_string              ; read string from file
    STMPD R13!,{r14 }
    MOV R2,R10

.getstring_loop
    MOV R0,#3
    MOV R3,#1
    SWI "XOS_GBPB"
    ADDCS R2,R2,#1
    LDMCSFD R13!,{pc }
    LDRB R0,[R2,#-1]
    CMP R0,#32
    BGE getstring_loop
    LDMFD R13!,{pc }
```