SUB

COLLABORATORS

	TITLE : SUB		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 29, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

SUB

1.1 Requesters

Requesters

Requesters are TypeSmith windows from which you can select options. Most requesters have Ok and Cancel gadgets to close the requester. The Ok gadget accepts whatever changes you made in the requester. The Cancel gadget ignores the changes you made.

Many of the Settings menu requesters have Save, Use and Cancel gadgets to close the requester because they allow you to customize TypeSmith's preferences. The Save gadget accepts the changes you made in the requester and saves to the TypeSmith tooltypes so that they will be used the next time you use TypeSmith. The Use gadget accepts the changes you made but only for the current session.

TypeSmith requesters are standard Amiga requesters which can be moved around the screen, and many can be resized.

The requesters use standard Amiga gadgets. Refer to your Amiga Workbench manual for more information on standard requesters and gadgets.

The only gadget which is not part of the standard Amiga interface is the popup gadget. This is similar to a cycle gadget except that you must press and hold the mouse button to see a list of the available options. Move the mouse so that the desired option is highlighted, and then release the mouse to select it.

See also:
 Panels

1.2 Panels

Panels

Panels are a special type of requester that can remain open while you use TypeSmith

TypeSmith panels are standard Amiga requesters which can be moved around the screen, but they are asynchronous, which allows them to remain open while you continue to work. The panels in TypeSmith are the Overview link TypeSmith:Help/English/vie/OVER} and Type Preview panels.

See also:
 Requesters

1.3 Recommended Reading

Recommended Reading

For information on subjects related to TypeSmith, the following sources are recommended:

ARexx:

```
An AmigaGuide to ARexx
  Robin Evans, 1993
  An online ARexx reference guide.
  Shareware (available on the Soft-Logik BBS).

Fast Guide to Arexx
  Vidia, 1992.
  A quick reference guide for ARexx.

The ARexx Cookbook
  by Merill Callaway. Whitestone, 1992. ISBN 0-9632773-0-8.
  A good tutorial guide to ARexx.

User’s Guide--ARexx
  by Isabelle Vesey and Robert Stephenson Weird.
  Commodore Electronics Limited, 1992. P/N: 368759-01.
  A good reference to ARexx commands.
```

Font Technology:

```
Adobe Type 1 Font Format, Version 1.1
  by Adobe Systems Inc. Addison-Wesley Publishing Company, Inc., 1990.
  ISBN 0-201-57044-0.
  A technical description of the PostScript Type 1 font format.

PostScript Language Reference Manual, 2nd Edition
  by Adobe Systems Inc. Addison-Wesley Publishing Company, Inc., 1990.
  ISBN 0-201-18127-4.
  A technical description of the PostScript language.

Intellifont Scalable Typeface Format
  Agfa Corporation, 1991.
  A technical description of the Compugraphic Intellifont format.
```

1.4 System Requirements

System Requirements

To use TypeSmith 2.5, you must have at least:

```
Computer:  any Amiga model.

Memory:    2MB total RAM (at least 512K chip RAM).

Drives:    hard drive with 2MB free space.

Workbench: AmigaDOS 2.04 or higher.
```

1.5 Font Hints

Hints improve the appearance of fonts at small sizes and low resolutions. A hint is a special instruction that adjusts how a character outline is displayed. When a character is drawn on the page or the screen, its points rarely coincide with the dots that can be displayed. This causes some strokes and serifs to be thicker or narrower than others. Hints identify the important aspects of a character outline, and distort the outline to make the character more regularly coincide with the displayed dots.

Hints for TrueType fonts are called instructions.

Soft-Logik DMF and PostScript Type 3 formats do not support hinting, so hints will be lost when you save in those formats.

1.6 Kerning

Kerning

Certain character pairs look better if the space between them is reduced. For example, the pair AV looks better when the letters are closer together, in most fonts.

Kerning values can be set for any pair of characters in a font. Thus, a pair of characters which are kerned together is referred to as a kerning pair. You can set a positive or negative kerning value for any pair which will be used any time the two characters are placed next to each other in a program which uses kerning pairs. Kerning is measured in hundredths of character units.

Kerning pairs are created and changed with the Metrics command.

Note: Compugraphic Intellifonts use kerning segments instead of kerning pairs. TypeSmith does not support kerning segments, so kerning values will not be saved when you export Compugraphic Intellifonts, and kerning segments will not be loaded when you open an Intellifont.

1.7 Composition References

Composition References

Composite characters differ from standard characters because they reference other characters in the font. Composite characters save you time and memory space because they are not actual drawings, but references to other characters. For example, an accented character would reference a base character and an accent character. The accented character would not contain the actual character drawing, but only references to its parts.

Composite characters are useful when you are creating complex characters which have repetitive parts. They also save time when you change a font at a later date. Changing the base characters changes all characters which reference the base characters.

You can reference any character in any other character. For example, character 233 could be made up of references to characters 101 and 194. Characters may have up to eight composition references.

Characters can also reference themselves. This is useful for storing parts without wasting a character position.

Composition references are made with the Compositions command.

Accent Positions for Compositions

While compositions can be used for any type of character, their most common use is for accented characters. The position of accents is irrelevant since composite parts may be stored in any character position; however, the standard character positions are recommended for accented characters because the TypeSmith windows for these character numbers are named in these positions.

Soft-Logik and PostScript fonts have one set of accent parts for both upper and lowercase letters. Compugraphic fonts have separate accent parts for upper and lowercase letters. Fonts produced with other font editors may have accent parts in different positions.

Saving a Compugraphic font in Soft-Logik or PostScript format will not change the accent positions because Compugraphic fonts have more accent parts.

Saving a Soft-Logik or PostScript font in Compugraphic format will dissolve all compositions because the composition format is incompatible. This will not affect the appearance of your font, since TypeSmith will compensate by copying the composition references to the applicable characters to preserve the desired combinations.

When you open an existing Soft-Logik font, you may find that the accent positions in it do not match the recommended positions in TypeSmith. This does not create any problems, except that you must remember that the character window name is for the suggested position for the character part.

Old Soft-Logik font accent positions may differ from the TypeSmith positions because Soft-Logik did not recommend positions until the release of TypeSmith. The positions used in TypeSmith were chosen because they match the PostScript accent positions.

Viewing Composite Characters

The default in TypeSmith is to show character compositions, since this is how they will be shown when used in an application. While you are creating the base parts to reference in compositions, you must turn off the display of compositions so that you can draw and edit paths stored in positions used by compositions.

To view the base parts of a font, toggle off the Show Composite Characters command. To view the composite characters instead of the parts, toggle on the Show Composite Characters command from the View menu. Composite characters will be shown when the command has a checkmark next to it in the menu.

You cannot edit a character which has composition references while composite characters are displayed. If you attempt to draw or edit the visible paths in the character window, you will be prompted to edit its component parts. Choosing the Edit Composition Table gadget from this requester is identical to choosing the Compositions command.

When composite characters are shown, titles at the top of the character window ↩ will reflect the character’s name in the TypeSmith character set. For example, ↩ character 194 will be named Acircumflex in the window title bar when ↩ compositions are shown. When composite characters are not shown, the window ↩ title will reflect the name of the character part normally stored in that space ↩ , if any. For example, character 194 will be named acute in the window title ↩ bar when compositions are not shown. (For fonts loaded from DMF and PFB formats ↩ .)

1.8 Path Direction and Character Fills

Character Fills

The paths of all characters in a font are filled automatically when drawn. (They are shown unfilled in the character window unless the Show Filled Characters tool or Show Filled Characters command is selected.)

Simple characters such as T and H are easy to fill since they have only a single path. Some characters may have two paths but are still easy to fill since the paths are completely separate.

If a character has two paths, and one path is completely enclosed by the other, such as the letter O, then the rules for filling it are more complex. PostScript Fonts use a fill routine called the Winding Number Fill, while Soft-Logik Outline fonts and Compugraphic Intellifonts use Even-Odd Fill.

Path Direction

Paths always retain the direction in which they were drawn. Direction can be either clockwise or counterclockwise from the startpoint. Path direction is extremely important for filling PostScript fonts.

While path direction can be ignored for Soft-Logik Outline fonts and Compugraphic Intellifonts, you should design your fonts so that characters fill in the same manner regardless of the font format, because you or someone else may convert your font to another format in the future.

Winding Number Fill (PostScript Fonts)

Winding Number Fill means that a point in a character is outside of the filled area if a line away from that point in any direction crosses exactly as many counterclockwise paths as it crosses clockwise paths. A good rule of thumb is that the outer path should always be drawn clockwise and the inner path counterclockwise.

If the paths are in the same direction, the enclosed area will not be filled, because a line from a point inside the enclosed area does not cross an equal number of clockwise and counterclockwise paths.

It does not matter if the outer path is clockwise or counterclockwise, technically, but it is a good rule to follow. There can also be more than one subpath. A good example of multiple subpaths is a target symbol. Each succeeding pair of paths creates a filled area.

If a path crosses over onto itself, the crossover area will be filled. If two subpaths cross each other, the crossover area will be unfilled if their paths are in different directions, and filled if they are in the same direction.

Even-Odd Fill (Soft-Logik Fonts and Compugraphic Intellifonts)

Even-Odd Fill means that a point in a character is outside of the filled area if a line away from that point in any direction crosses an even number of paths, regardless of their direction. For an O, both paths could be drawn clockwise or counterclockwise, and the fill would still be correct.

If a path crosses over onto itself, the crossover area will be unfilled. If two subpaths cross each other, the crossover area will be unfilled regardless of the directions of their paths.

Changing Path Direction

Each path drawn in a character will be either clockwise or counterclockwise, depending on how the path is drawn. The direction of the path can be changed at any time with the Clockwise and Counterclockwise toggle commands.

To find the direction of a path, select one point in the path and refer to the Path menu. The path direction will have a checkmark next to it in the menu and will be ghosted.

To change the direction of the path, choose the opposite command then refer to the Path menu. The command selected will now be checked and ghosted.

If a character has only one path, it should be clockwise. If a character has two paths, the outer path should be clockwise and the inner path should be counterclockwise.

1.9 Background Grid

Background Grid

The TypeSmith grid is the computer equivalent of graph paper. It allows you to align paths when drawing or moving them. It can be set to a range of sizes and it will not be printed. The grid in TypeSmith has two major advantages over graph paper: points and paths can be snapped to it, and it can be turned off when necessary. You may want to turn the grid display off when the screen is cluttered or when the grid spacing is very small. Points and paths may still be snapped to the grid even when it is not visible.

1.10 ARexx

ARexx

ARexx is an easy programming language which has grown popular on the Amiga and is part of AmigaDOS 2 and higher. ARexx is not required to use TypeSmith it works correctly without it, but ARexx adds increased functionality to the program.

You can use TypeSmith’s ARexx commands as the basic components of more powerful commands. With ARexx, you can add new capabilities to TypeSmith.

The TypeSmith ARexx Port

ARexx macro scripts send instructions to TypeSmith through Amiga message ports. TypeSmith opens a port called TYPESMITH. If a second copy of TypeSmith is launched, it will open a port called TYPESMITH.1. Additional copies of TypeSmith will open subsequently numbered ports. The port name is important because ARexx commands must be sent to the correct port.

All TypeSmith ARexx commands that perform actions in a character window will be executed in the active character window.

To use an ARexx macro with TypeSmith, you must first start ARexx. You can start ARexx before or after launching TypeSmith. TypeSmith allows you to load up to twenty macros assigned to the ten function keys, with and without the Shift key modifier. To assign a macro to a function key, choose the Install ARexx Macros command from the Settings menu.

Included ARexx Scripts

Several ARexx Scripts are included with TypeSmith. Refer to the Release Notes included with this package for a list of these macros.

Script Framework

TypeSmith ARexx scripts must correspond to the ARexx standard. Every ARexx script must start with a comment line. Comments must start with /* and end with */.

The second line in your script should set the port address, so that commands will be addressed to the correct port. The command ADDRESS 'TypeSmith' is used to direct ARexx commands. If your commands must be directed to a second or subsequent copy of TypeSmith, you should adjust the port name as required.

If your script uses any command which returns a value, you must enable results with the OPTIONS command, as is standard in ARexx. Returned values are always stored in the ARexx RESULT variable. i.e.,

```
  /* This is a TypeSmith ARexx Script */
  ADDRESS 'TYPESMITH'
  OPTIONS RESULTS

  'display_alert 'Hello.|Ok'

  /* End of macro */
```

You should enclose commands, text strings and negative numbers in single quotes. i.e.,

```
  'lineto -50'
  'open_dmf work:psfonts/Bodoni.DMF'
```

ARexx commands are not case sensitive.

Errors

When a command is executed, ARexx returns a code to let the script know whether the command was successful. The code is stored in the RC (Return Code) variable. The decimal error codes returned by TypeSmith are:

```
  RC=0    Successful, no error.
  RC=5    Warning only.
  RC=10   Something is wrong.
  RC=20   Fatal error.
```

TypeSmith allows you to draw character paths with the MOVETO, LINETO and CURVETO commands. You can draw paths with invalid coordinates which can create paths outside the visible area. An error code will not be returned when you draw a path with invalid coordinates.

Drawing Paths

TypeSmith character paths are standard paths which correspond to the PostScript path model. When TypeSmith's tools are used for drawing, TypeSmith ensures that paths drawn are valid. It is essential that your ARexx scripts create valid paths only.

A valid character path must begin with a MOVETO command. This moves the pen to the start of the path. Paths may then have any number of LINETO and CURVETO segments up to the maximum allowable (512 points).

Character paths must end with a CLOSEPATH command. If the last segment is a LINETO, you should use the CLOSEPATH command to draw the line or else the CLOSEPATH will be zero length. If the last segment is a CURVETO, you cannot avoid a zero length CLOSEPATH segment.

Characters may have a maximum of 100 paths.

Examples:

```
  /* Bad Path: no closepath command */
  moveto 10 10
  lineto 50 10
  lineto 50 45
  lineto 30 60
  /* End of macro */


  /* Bad Path: no moveto command */
  lineto 50 10
  lineto 50 45
  lineto 30 60
  closepath
  /* End of macro */


  /* Good Path */
  moveto 10 10
  lineto 50 10
  lineto 50 45
  lineto 30 60
  closepath
  /* End of macro */
```