NintendoWare for CTR

The Basics of Text Rendering

2010/12/02

Version 1.0.1

PROVISIONAL TRANSLATION

The content of this document is highly confidential and should be handled accordingly.

Table of Contents

1 Introduction 5
1.1 About This Manual 5
1.2 Text Rendering 5
1.2.1 Low-Level Features 5
1.2.2 High-Level Features (Which Use Low-Level Features) 5
1.3 Font Licenses 5

2 Terminology 6
2.1 Character Codes 6
2.2 Glyphs 7
2.3 bcfnt 8
2.4 Other 9

3 Structure of the Text Rendering Library 10
3.1 Text Rendering Parameters 10
3.2 Library Structure 11

4 Font Resources 12
4.1 Data for Each Character 12
4.1.1 Glyph Image 12
4.1.2 Character Parameters 12
4.1.3 Character Code Table 13
4.2 Data Common to All Fonts 14
4.2.1 Sheet Information 14
4.2.2 Substitute Characters 14
4.2.3 Encoding 14
4.2.4 Maximum Character Width 14
4.2.5 Default Character Width Information 14
4.2.6 Text Positional Information 14

5 Archive Fonts 16
5.1 Glyph Groups 16
5.2 Archive Fonts 16

6 Overview of Text Rendering 18
6.1 Rendering Characters 18
6.2 Rendering Text 18

7 Revision History 19

Figures

Figure 3-1 Parameters Used to Render Characters 10
Figure 3-2 Structure of the Text Rendering Library 11
Figure 4-1 A Stored Glyph Image 12
Figure 4-2 Character Parameters 13
Figure 5-1 Graphical Depiction of the Contents of sample.xggp 16
Figure 5-2 Conceptual Diagram of an Archive Font 17

1 Introduction

1.1 About This Manual

This manual provides basic knowledge handling the display of text with NintendoWare for CTR. It provides a foundation for understanding the other manuals that relate to text rendering and the function reference manual. You should read this manual before using NintendoWare for CTR to handle the display of text.

1.2 Text Rendering

This manual refers to the following libraries and related manuals for text rendering:

1.2.1 Low-Level Features

1.2.1.1 Libraries and References

- `Font` class
- `ResFont` class
- `ArchiveFont` class
- `PackedFont` class
- `CharWriter` class
- `TextWriter` class
- `CharStrmReader` class
- `TagProcessor` class

1.2.1.2 Manuals

- `FontConverter_Manual.pdf`
- `FontConverterConsole_Manual.pdf`
- `Font.pdf`
- `Writer.pdf`

1.2.2 High-Level Features (Which Use Low-Level Features)

- Layout Library

1.3 Font Licenses

NintendoWare for CTR can use any of the fonts that are installed on the PC for displaying characters on the CTR system. However, commercial software that uses those fonts must be licensed to use them. It is up to you to obtain the appropriate font licenses for your game software.

NintendoWare for CTR does not come with any font licenses.

2 Terminology

This chapter defines certain terms used in NintendoWare for CTR for text rendering. In some cases, the meaning of these terms differs from general usage. The terms are grouped by category and listed in topical (rather than alphabetical) order.

2.1 Character Codes

- Character

The figure to be rendered. The character includes not only the visible part of the figure but also the surrounding space.

- Character code

The numeric value assigned to a character.

- Character set

A set of character/character code pairs.

- Text encoding

A method for converting and positioning character codes to represent text as a byte string.

- ISO 8859-1

A superset of ASCII with several additional European accented characters and symbols. Also known as Latin-1. Character codes in the 0x00 to 0xFF range are used, but the codes in the ranges 0x00 to 0x1F and 0x80 to 0x9F are for control characters, which cannot be displayed.

- CP1252 (Code Page 1252)

A character set for European languages defined by Microsoft Corporation for Windows. It is the same as ISO 8859-1, except that displayable characters are used in place of control characters in the range 0x80 to 0x9F. All characters that can be displayed with ISO 8859-1 can be displayed by CP1252.

- JIS X 0201

A Japanese-language character set established as a Japanese Industrial Standard (JIS) that includes all double-byte characters, including hiragana, katakana, and kanji characters.

- JIS X 0208

A Japanese-language character set established as a Japanese Industrial Standard (JIS) that includes single-byte ASCII characters and single-byte (half-width) katakana.

- ShiftJIS

The standard text encoding format for the Japanese-language environment. Can be used with both JIS X 0201 and JIS X 0208.

- Unicode

A character coding system designed to encompass all the characters from all the world's languages in a single character set. Unicode defines both character sets and encoding methods.

- UTF16

A text encoding format defined for Unicode. Every character is expressed uniformly using two bytes. Consequently, UTF16, unlike most other encoding formats, is not compatible with the ASCII character set.

- UTF8

A text encoding format defined for Unicode. Different characters in this format have different byte counts, but all ASCII characters use 1-byte codes. Consequently, UTF8 is compatible with other character string encoding formats when the strings consist of ASCII characters only.

2.2 Glyphs

- Glyph

The visibly recognizable shape of a character.

- Glyph image

Same as glyph, but with emphasis on its aspects as an image.

- Glyph width

The width of the rectangle circumscribing the glyph.

- Left space width

The width of the blank space between a glyph and the character immediately to its left. For details, see section 4.1.2 Character Parameters.

- Right space width

The width of the blank space between a glyph and the character immediately to its right. For details, see section 4.1.2 Character Parameters.

- Character width

The width of the character, which is equal to glyph width plus left space width plus right space width. In symbols, character width = glyph width + left space width + right space width. For details, see section 4.1.2 Character Parameters.

- Character width information

Information that relates to the width of the character. There are four widths: left space width, right space width, glyph width, and character width.

- Baseline

The reference line for the positioning of glyphs in the vertical direction when characters are lined up horizontally (i.e., when text is being rendered.)

- Ascender line

The line that defines the top edge of the space the character can occupy. See section 4.2.6 Text Positional Information.

- Descender line

The line that defines the bottom edge of the space the character can occupy. See section 4.2.6 Text Positional Information.

- Ascent

The distance between the baseline and the ascender line. See section 4.2.6 Text Positional Information .

Descent

The distance between the baseline and the descender line. See section 4.2.6 Text Positional Information.

- Font height

The distance between the ascender line and the descender line. See section 4.2.6 Text Positional Information.

- Font width

The character space that serves as the basis for the tab width. It is the same for all characters. See section 4.2.6 Text Positional Information.

2.3 bcfnt

- bcfnt (Binary Ctr FoNT)

Normal font data used by NintendoWare for CTR. This is also the filename extension for files that store this font data.

- bcfna (Binary Ctr FoNt Archived)

Compressed font data used by NintendoWare for CTR. This is also the filename extension for files that store this font data. This data can be used as a font by extracting/uncompressing at the glyph group level.

- Font resources

Normal font data used by NintendoWare for CTR. This term refers to the same thing as bcfnt.

- Archive fonts

Compressed font data used by NintendoWare for CTR. This term refers to the same thing as bcfna.

- FontConverter

A Windows tool for creating bcfnt and bcfna data. This tool runs on Microsoft Windows and is provided with NintendoWare for CTR.

- Glyph group

A glyph group is a collection of glyphs. When using bcfna, fonts can be extracted by executing each glyph group defined in the bcfna.

- Group set

A group set is a collection of glyph groups.

- Substitute character

The character used as a substitute when information is requested for a character that is not included in a font. For details, see section 4.2.2 Substitute Characters.

- Sheet

When using NintendoWare for CTR fonts, glyphs are stored as textures that can be used by the CTR system. Each texture from which a glyph is rendered is called a sheet. For details, see section 4.2.1 Sheet Information.

2.4 Other

- Tagged string

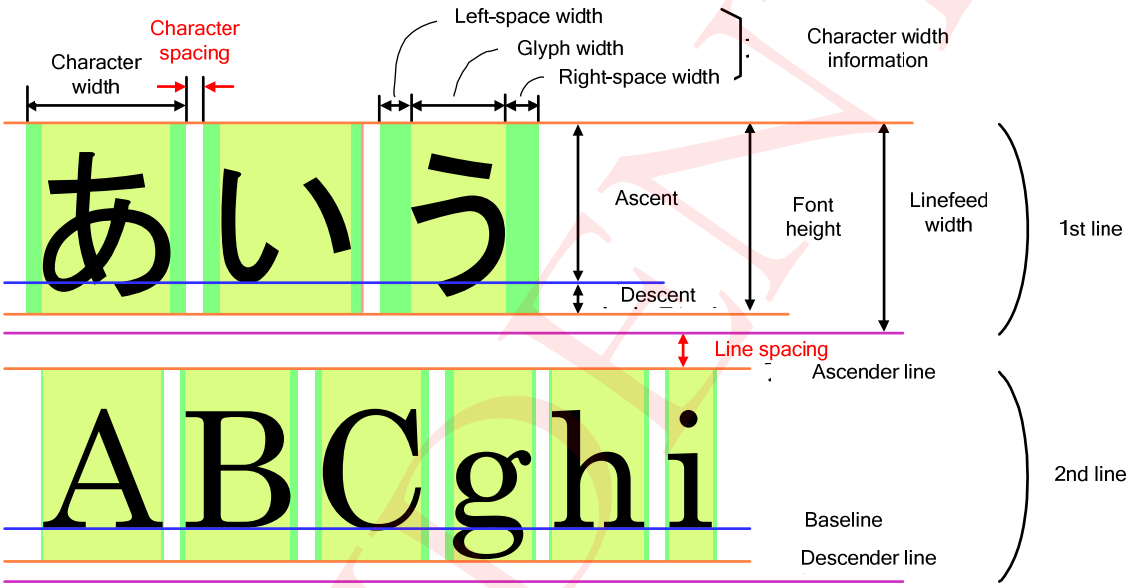
A string that is embedded with commands specific to its own rendering. While NintendoWare for CTR provides a framework for processing a tagged string, the application must prepare the actual processes that will be carried out on the tags embedded in the string.

3 Structure of the Text Rendering Library

3.1 Text Rendering Parameters

Figure 3-1 shows the parameters that are used by the Text Rendering library when it renders characters. Items shown in black are specified by the font resource (bcfnt). Items in red are specified inside the program.

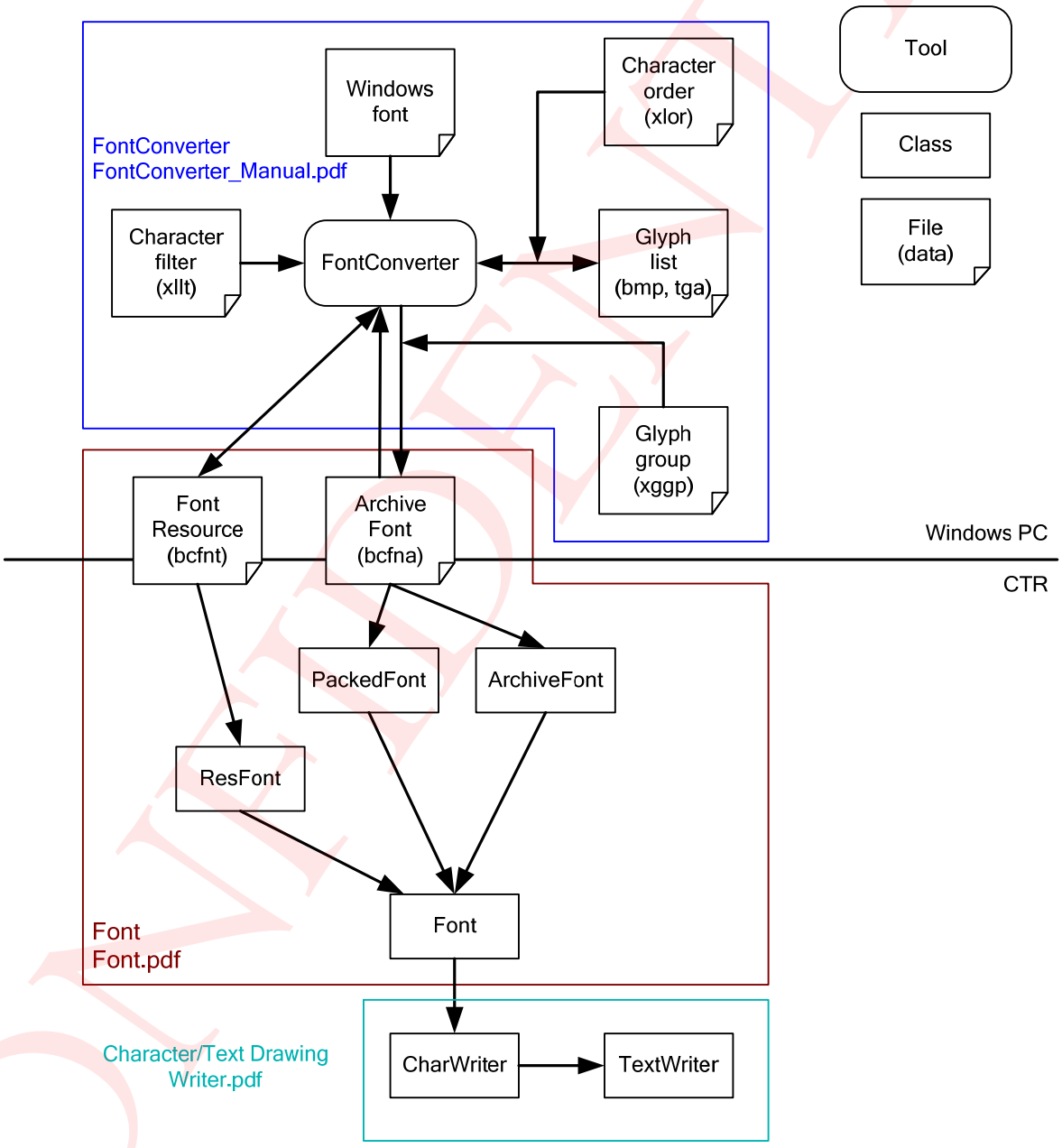
Figure 3-1 Parameters Used to Render Characters



3.2 Library Structure

Figure 3-2 shows an overall schematic of the Text Rendering library. The arrows indicate the direction of data flow.

Figure 3-2 Structure of the Text Rendering Library



On the Windows PC, FontConverter operates at the heart of the process to create the font resource (bcfnt), which is used with the library to render text on the CTR system.

4 Font Resources

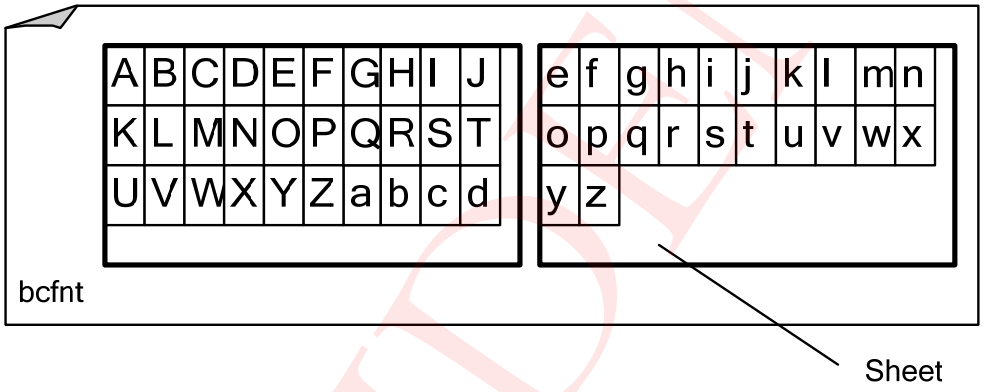
bcfnt (font resource data) is the font data used by NintendoWare for CTR. This chapter provides a simple explanation of the information that is stored in bcfnt. These parameters can be specified by the image file used as input to FontConverter or FontConverterConsole.

4.1 Data for Each Character

4.1.1 Glyph Image

In bcfnt, glyphs are stored in the form of textures that can be used by the CTR system. Each texture is called a sheet. Each sheet is organized as a rectangular grid subdivided into cells. A glyph is rendered in each of these cells.

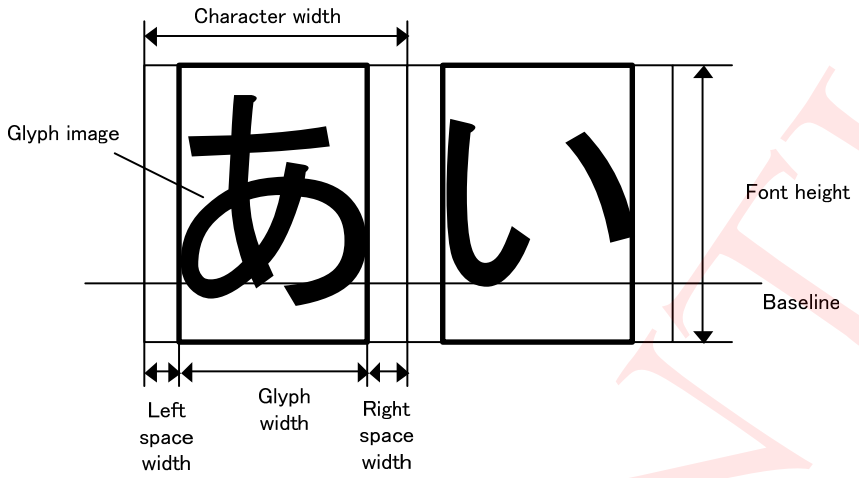
Figure 4-1 A Stored Glyph Image



4.1.2 Character Parameters

Figure 4-2 shows the parameters that are stored in bcfnt for each character.

Figure 4-2 Character Parameters



Each character has four width-related parameters: the left space width, the right space width, the glyph width, and the character width

The glyph width is the width of the rectangle that circumscribes the image. This is determined by the glyph image and cannot be specified.

The left space width and right space width are the widths of the blank space regions to the left and the right of the glyph. Note that these two values can take negative numbers, in which case characters overlap when displayed. As shown in Figure 3-1 Parameters Used to Render Characters, the program can set a parameter to open the "character spacing," which is the space between the right-space width of one character and the left-space width of the next. Normally this value is set to 0, but if a different character spacing is necessary, this parameter can be used to make the change.

Since the character width is the sum of the glyph width and the left and right space widths, it can be smaller than the glyph width and can also assume a negative value. When a string of characters is rendered, the character width defines the distance between one character and the next.

There is no individual “glyph height” value that affects the height of a character. Rather, there is a value called the font height which is shared by the entire font set.

4.1.3 Character Code Table

A subset of the character code table is stored in bcfnt so that the positions of glyph images in the sheet can be obtained from their character codes.

The character code table is created for bcfnt by FontConverter. A different subset can be created for each bcfnt and it can include any of the character codes. However, if a character code is not included in the subset created by FontConverter for a given bcfnt, that bcfnt cannot be used to display that particular character.

4.2 Data Common to All Fonts

4.2.1 Sheet Information

The sheet width and height, the sheet's texture format, the width and height of the rectangle inside the sheet, and the baseline position are all values that are common to all sheets. This information is stored inside bcfnt.

4.2.2 Substitute Characters

A substitute character is registered for use at times when a request is made for a glyph associated with a character not stored in bcfnt. The text rendering functions of NintendoWare for CTR display this substitute character without generating an error.

4.2.3 Encoding

The text rendering features of NintendoWare for CTR support a number of formats for text encoding. However, as explained in section 4.1.3, Character Code Table, each bcfnt stores a subset of the character code table and supports only a certain set of characters. Thus, text will not be displayed correctly if the character set in the character code table cannot be handled by the text encoding format being used by the text rendering feature.

bcfnt stores the text encoding format that the text rendering features use to interpret the text.

4.2.4 Maximum Character Width

The widest character width for all of the characters contained in bcfnt is stored by FontConverter.

4.2.5 Default Character Width Information

bcfnt does not need to store character-width information for every single character. If a request comes for width-related information for a character for which character-width information is not stored, the default character-width information is used instead. One set of default information is stored for the entire font.

If a request comes for width-related information for a character that is not supported by the font, the character-width information for the substitute character is used. If the substitute character does not have its own character-width information, the default information is used instead.

4.2.6 Text Positional Information

bcfnt stores parameters that are used to position each text character. These are the font width, the ascent, the descent, the font height and linefeed width shown in Figure 3-1 Parameters Used to Render Characters

Note that the font height is simply the sum of the ascent and the descent.

The ascent and the descent parameters are used when upper and lower edges of characters are rendered to touch something.

The font height and font width are used as standards for text scaling and for the tab width.

The linefeed width defines the space to the next line after a linefeed. By default, FontConverter assumes the linefeed width is equal to the font height, but this parameter can be changed by explicitly specifying a value. The linefeed width can either be narrowed or widened. The "line space" parameter in Figure 3-1 Parameters Used to Render Characters, is a Text Rendering Library feature by means of which the program can change the space between lines. The spacing between lines is normally defined by the specification for the linefeed width, so the line space parameter is used only in special situations to change the line spacing.

5 Archive Fonts

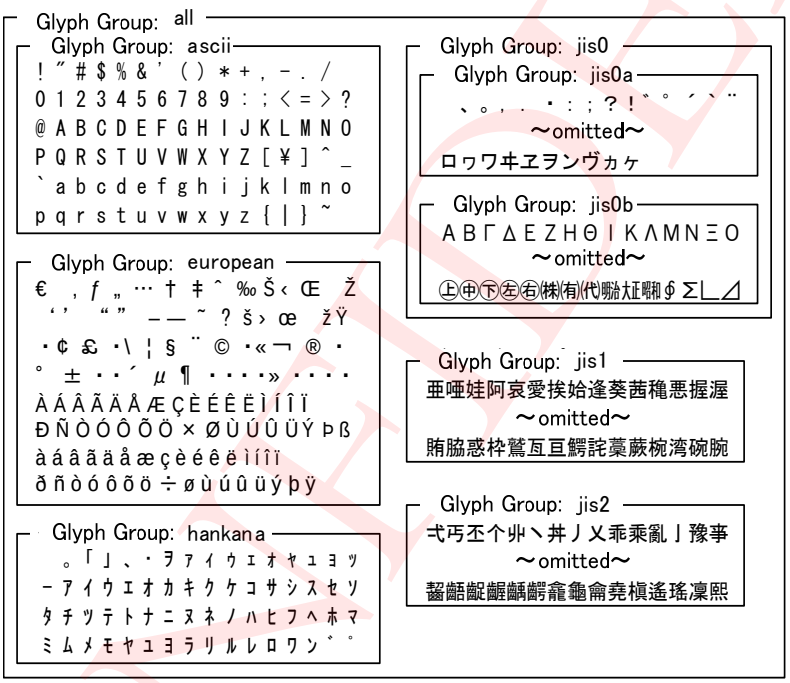
bcfna (archive font data) is one of the font data formats that can be used with NintendoWare for CTR. The archive font contains bcfnt (font resource data) as a base and includes several types of additional information and is used for special applications.

5.1 Glyph Groups

The term “glyph group” is the name given to a collection of glyphs. Figure 5-1 is a graphical depiction of the glyph group defined in the glyph group file, `sample.xggp`, included with FontConverter.

`sample.xggp` contains glyph groups having names like “all, “ascii,” “hankana,” and “jis_level_0.” For example, the glyph group “ascii” is defined as the set of ASCII characters. Furthermore, the glyph group “jis_level_0” is the combination of the glyph group “jis_level_0a” plus “jis_level_0b.” The glyph group “all” represents the sum total of all of the glyph groups.

Figure 5-1 Graphical Depiction of the Contents of sample.xggp



5.2 Archive Fonts

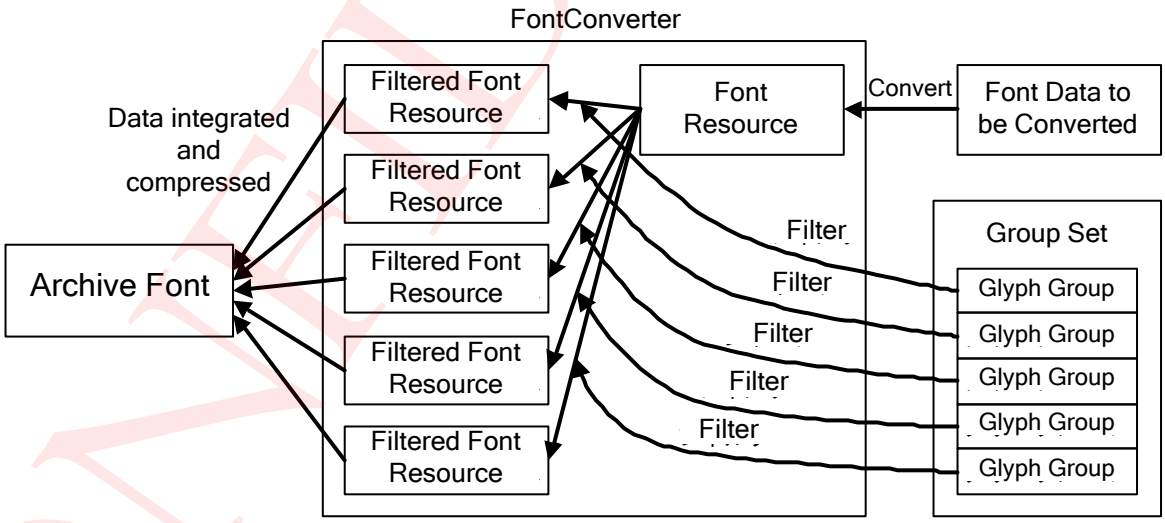
An archive font is compressed font data that includes glyph group information. Although this data cannot be directly used as a font, it can be used as a font when extracted/uncompressed on the CTR system.

Archive fonts can be used to build fonts by extracting the glyphs included in a glyph group that has been specified by name. For example, an archive font possessing the information given in Figure 5-1 for sample.xggp can be created that includes ASCII characters, European characters, and Japanese characters. A font that allows the use of ASCII characters can be built by extracting the glyph group “ascii” from this archive font. This allows a more compact font than would result if all glyphs in the archive were extracted. More than one glyph group can also be extracted to make a single font. For example, if the glyph groups “ascii” and “jis0a” are both specified for extraction from the archive font under consideration, a font can be built that allows the use of ASCII characters, Japanese hiragana, katakana, and symbols.

The main purpose of archive fonts is not compression, but rather the extraction of partial font data in the form of glyph groups. For this reason, if all you want to do is to minimize the size of font resources, better results than using archive fonts are possible by compressing font resources individually. This approach is also probably easier for applications to handle.

Archive fonts are created by specifying to FontConverter the font data to be converted (such as image files) and the group sets to be included (as a collection of glyph group definitions). Conceptually, archive fonts are built according to the flow depicted in Figure 5-2. In other words, each glyph group included in a group set can be viewed as a character filter that is applied to the base font data. The resulting font resource is integrated and compressed. The act of extracting glyph groups from an archive font can be thought of as reproducing each separate filtered font resource.

Figure 5-2 Conceptual Diagram of an Archive Font



6 Overview of Text Rendering

This chapter provides a simple explanation of how NintendoWare for CTR renders characters.

6.1 Rendering Characters

Glyph images are stored in the form of textures in bcfnt. The rendering of characters involves the pasting of glyph image textures on rectangles.

A character is rendered for display on the screen by pasting the glyph texture on a rectangle whose width is the glyph width and whose height is the font height.

Character rendering is performed by the `CharWriter` class.

6.2 Rendering Text

Text can be rendered using the `TextWriter` class. The `TextWriter` class calculates the display position of each character in the string and then uses the `CharWriter` class to render the individual characters to display the text.

Text is displayed by rendering each character in its appropriate position based on the left/right space widths and the leading stored in the bcfnt file.

7 Revision History

Version	Revision Date	Category	Description
1.0.1	2010/12/02	Changed	• Overall Updated document properties.
1.0.0	2010/07/29	Changed	• Overall Revised formatting.
	2010/01/15	Changed	• Overall Revised figure content.
	2009/10/30	-	• Initial version.