libpvf Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Document serial number: 000004892117

Table of Contents

1 Library Overview

Overview

The libpvf library provides glyph images of PlayStation®Vita built-in vector fonts to an application program.

By using libpvf, an application program can get information about vector fonts that are installed on the PlayStation®Vita and from which the application can select arbitrary fonts to be used. The application program can find a matching font from abstract information (such as italic bold Latin) or a font that comes closest to the matching criteria.

libpvf also enables an application program to get glyph image data for specific characters included in specific fonts that are part of vector font groups installed on the PlayStation®Vita.

These glyph images are made up of pixels whose brightness is quantized using two or more values.

libpvf also provides an application program with information it needs to lay out characters. By using this layout information, the application program can arrange proportional fonts attractively.

Related Files

The following files are required to use libpvf.

Filename	Description
font/libpvf.h	Header file
libScePvf_stub.a	Stub library file
libScePvf_stub_weak.a	Weak import stub library file

Sample Programs

The following sample programs are provided with libpvf.

samples/sample_code/graphics/api_libpvf/basho/

Shows the basic method for using libpvf. It creates image files by sequentially opening a Japanese font and multiple Latin fonts.

samples/sample_code/graphics/api_libpvf/fontcache/

Shows how to implement a font cache for external fonts and how to use it with libpvf.

samples/sample_code/graphics/api_libpvf/simple/

Exemplifies the simplest program created using the libpvf library.

2 Usage Procedure

Library Initialization

To use libpvf, a library instance must be created with the `scePvfNewLib()` function. The library is initialized when this instance is created. `scePvfNewLib()` returns a library ID that points to the newly created library instance.

By calling `scePvfNewLib()` multiple times, the application program can hold multiple library instances.

Finding a Font

The number of vector fonts that are installed on the PlayStation®Vita can be obtained by using the `scePvfGetNumFontList()` function. Detailed font design information can be obtained by using the `scePvfGetFontList()` function, and a font can be found by using the `scePvfFindOptimumFont()` or `scePvfFindFont()` function.

Font information can be obtained for a font by using the `scePvfGetFontInfoByIndexNumber()` function and providing it with a font index number. Font information can also be obtained when a font is opened by calling the `scePvfGetFontInfo()` function and providing it with the relevant font ID.

Opening a Font

The `scePvfOpen()` function opens a font having the specified index number and returns its font ID. Two modes are available when opening the font. `SCE_PVF_FILEBASEDSTREAM` performs the required processing while reading font data sequentially from a file. `SCE_PVF_MEMORYBASEDSTREAM` first reads all font data into memory, then performs the required processing. Although `SCE_PVF_FILEBASEDSTREAM` mode requires less memory than `SCE_PVF_MEMORYBASEDSTREAM` mode, it will cause the application program to run slower.

An application program can also open TrueType and OpenType fonts by using the `scePvfOpenUserFile()` or `scePvfOpenUserMemory()` function.

Getting Character Metrics

Character metrics information can be obtained by calling the `scePvfGetCharInfo()` function and providing it with the font ID and character code.

Getting the Glyph Image for a Character

The glyph image for a character can be obtained by calling the `scePvfGetCharGlyphImage()` function and providing it with the font ID and character code. This function stores glyph images in user memory space with a 1/64 pixel precision.

Closing a Font

A font is closed by calling the `scePvfClose()` function and providing it with the font ID that was obtained from the `scePvfOpen()` function when the font was opened.

Library Termination

A library instance is closed by calling the `scePvfDoneLib()` function and providing it with the library ID that was obtained from the `scePvfNewLib()` function. This will close all fonts opened by that library instance, free the memory used by that instance, and finally destroy the instance.

Example of Library Use

A program example is shown below.

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <font/libpvf.h>

/* Memory allocation function */
static ScePvf_t_pointer cb_Alloc
(
 ScePvf_t_pointer pMyData, /* Pointer to user data */
 ScePvf_t_u32 size /* Requested size */
 )
{
   return ( malloc (size) );
}

/* Memory reallocation function */
static ScePvf_t_pointer cb_Realloc
(
 ScePvf_t_pointer pMyData, /* Pointer to user data */
 ScePvf_t_pointer old_p /* Pointer to the original area to be reallocated */
 ScePvf_t_u32 size /* Requested size */
 )
{
   return ( remalloc (old_p, size) );
}

/* Memory release function */
static void cb_Free
(
 ScePvf_t_pointer pMyData, /* Pointer to user data */
 ScePvf_t_pointer p /* Pointer to area to be freed */
 )
{
   free (p);
}

/* libpvf test function */
void myTest_libpvf ( void )
{
   ScePvf_t_error errorCode;
   ScePvf_t_libId libID;
   ScePvf_t_initRec initParams = {
      NULL, / Pointer to user data */
      4, /* Maximum number of fonts to be opened simultaneously */
      NULL, /* Handle for cache instance */
      cb_Alloc, /* Memory allocation function */
      cb_Realloc, /* Memory reallocation function */
      cb_Free,   /* Memory free function */
   };
   ScePvf_t_int numFontList;
   ScePvf_t_fontStyleInfo aFontStyleInfoList [40];
   ScePvf_t_fontId fontID;
   ScePvf_t_charCode charCode=(0x0041);
   ScePvf_t_charInfo charInfo;
   ScePvf_t_userImageBufferRec myImage;
#define MYIMAGE_WIDTH (128)
#define MYIMAGE_HEIGHT (128)
   unsigned char *myImageBuffer [(MYIMAGE_WIDTH / 2) * MYIMAGE_HEIGHT];
```

```
    /* Create font library instance */
    libID = scePvfNewLib (&initParams, &errorCode);
    if ( errorCode != SCE_OK ) ; /* Error processing */

    /* Set expected resolution */
    errorCode = scePvfSetResolution (libID, 128.0f, 128.0f);
    if ( errorCode != SCE_OK ) ; /* Error handling */

    /* Set em square value (process using a value compatible with libpgf) */
    errorCode = scePvfSetEM (libID, 72.0f / (10.125f * 128.0f));
    if ( errorCode != SCE_OK ) ; /* Error handling */

    /* Open default font */
    fontID = scePvfOpen (libID, 0, SCE_PVF_FILEBASEDSTREAM, &errorCode);
    if ( errorCode != SCE_OK ) ; /* Error handling */

    /* Specify character size */
    errorCode = scePvfSetCharSize (fontID, 10.125f, 10.125f);

    /* Get information related to characters having character code charCode */
    /* that are included in the opened font */
    errorCode = scePvfGetCharInfo (fontID, charCode, &charInfo);
    if ( errorCode != SCE_OK ) ; /* Error handling */

    myImage.pixelFormat = SCE_PVF_USERIMAGE_DIRECT4_L;
    myImage.rect.width = MYIMAGE_WIDTH;
    myImage.rect.height = MYIMAGE_HEIGHT;
    myImage.bytesPerLine = MYIMAGE_WIDTH / 2;
    myImage.reserved = 0;
    myImage.buffer = (ScePvf_t_u8 *)myImageBuffer;
    myImage.xPos64 = 20;
    myImage.yPos64 = 40;
    /* Get character glyph images */
    errorCode = scePvfGetCharGlyphImage (fontID, charCode, &myImage);
    if ( errorCode != SCE_OK ) ; /* Error handling */

    /* Close font */
    errorCode = scePvfClose (fontID);
    if ( errorCode != SCE_OK ) ; /* Error handling */

    /* Destroy font library */
    errorCode = scePvfDoneLib (libID);
    if ( errorCode != SCE_OK ) ; /* Error handling */

    return;
}

/* Main function */
int main ( void )
{
    myTest_libpvf ();
    return ( 0 );
}
```

3 Explanation of Operation

libpvf Internal Processing

scePvfNewLib()

This function gets information related to installed vector fonts, allocates the memory area for managing the library instance, copies the information, and then returns the area address as a library instance.

scePvfSetResolution()

This function sets expected resolution values specified with the arguments to the library instance.

scePvfSetEM()

This function sets the em value specified with the argument to the library instance.

The em value is the numerical value used as the point of reference for metrics information values. When expecting 128dpi, the value shown in the metrics information will be compatible with libpgf fonts if 72.0f/(10.125*128.0) is specified as the EM value.

scePvfOpen()

This function opens the font of the number specified with the argument and returns the font ID.

This font ID is managed by the library instance specified in the argument.

scePvfSetCharSize()

This function sets the horizontal and vertical size values specified with the arguments as the rasterized character size of the font ID specified with the argument.

This size will be applied for subsequent rasterizations.

scePvfGetCharInfo()

This function obtains metrics information of the character code specified with the argument of the font ID specified with the argument, and copies this information to the `ScePvf_t_charInfo` area specified with the argument.

scePvfGetCharGlyphImage()

This function obtains the glyph image of the character code specified with the argument of the font ID specified with the argument, and copies the glyph image according to `ScePvf_t_userImageBufferRec` specified with the argument.

scePvfClose()

This function closes the font of the font ID specified with the argument.

scePvfDoneLib()

This function closes all unclosed fonts that are owned by the library instance specified in an argument and frees all memory that was allocated by that instance.

Information That an Application Program Can Obtain from libpvf

An application can use libpvf to get the information shown below from the PlayStation®Vita system.

Information	Description
Information related to the installed vector fonts	The number of vector fonts installed on the PlayStation®Vita and information related to each vector font (`ScePvf_t_fontStyleInfo`, `ScePvf_t_fontInfo`) can be obtained.
Information related to a specific font in the installed vector fonts	Information common to all characters included in the specified vector font installed on the PlayStation®Vita (`ScePvf_t_iGlyphMetricsInfo`, `ScePvf_t_fGlyphMetricsInfo`) can be obtained.
Information related to a specific character of a specific font of the installed vector fonts	Glyph information for a specific character that is part of a specific vector font installed on the PlayStation®Vita (`ScePvf_t_charInfo`) can be obtained.

Glyph Metrics Information

Glyph metrics information (`ScePvf_t_iGlyphMetricsInfo`, `ScePvf_t_fGlyphMetricsInfo`) consists of the elements shown in the figure below.

Name of Value	Meaning
width	Glyph width
height	Glyph height
ascender	Distance from the base line to the virtual top of the character in the design. This is a positive value in the horizontal typesetting example in the figure below.
descender	Distance from the base line to the virtual bottom of the character in the design. This is a negative value in the horizontal typesetting example in the figure below.
horizontalBearingX	Numeric value used when laying out characters horizontally. Distance on the X-axis from the character reference point to the left edge of the character rectangle of that character. A positive value is taken towards the right. This is a negative value in the horizontal typesetting example in the figure below.
horizontalBearingY	Numeric value used when laying out characters horizontally. Distance on the Y-axis from the character reference point to the top edge of the character rectangle of that character. A positive value is taken upwards. This is a negative value in the example in the figure below.
horizontalAdvance	Numeric value used when laying out characters horizontally. Horizontal distance from the character reference point to the next character reference point. A positive value is taken towards the right. This is a positive value in example in the figure below.
verticalBearingX	Numeric value used when laying out characters vertically. Distance on the X-axis from the character reference point to the left edge of the character rectangle of that character. A positive value is taken towards the right. This is a negative value in the vertical typesetting example in the figure below.

Name of Value	Meaning
verticalBearingY	Numeric value used when laying out characters vertically. Distance on the Y-axis from the character reference point to the top edge of the character rectangle of that character. A positive value is taken downward. This is a positive value in the example in the figure below.
verticalAdvance	Numeric value used when laying out characters vertically. Vertical distance from the character reference point to the next character reference point. A positive value is taken downward. This is a positive value in the example in the figure below.

The units for these numeric values are pixels. Pixels are different from points, which are used as the units of character size. Members of `ScePvf_t_iGlyphMetricsInfo` whose names end in "64" contain values that are 64 times that of members whose names do not end in "64."

Rasterized Glyph Images

An application program can get glyph images whose brightness is quantized as two or more values.

An application program can specify the buffer memory to be used for output, and the position at which glyph images are to be output in a structure of type `ScePvf_t_userImageBufferRec`. The application copies the glyph image to the buffer by calling the `scePvfGetCharGlyphImage()` function and passing it the desired character code.

The precision of the position is 1/64 pixel. This is related to the fact that the *horizontalAdvance* value of a character is in units of 1/64 pixel. The brightness of pixels of a glyph that is placed spanning the pixel boundary of the buffer is divided internally among the neighboring pixels.

Breakdown of ScePvf_t_fGlyphMetricsInfo (Breakdown of ScePvf_t_iGlyphMetricsInfo)

Figure 1 Glyph Metrics for Horizontal Typesetting

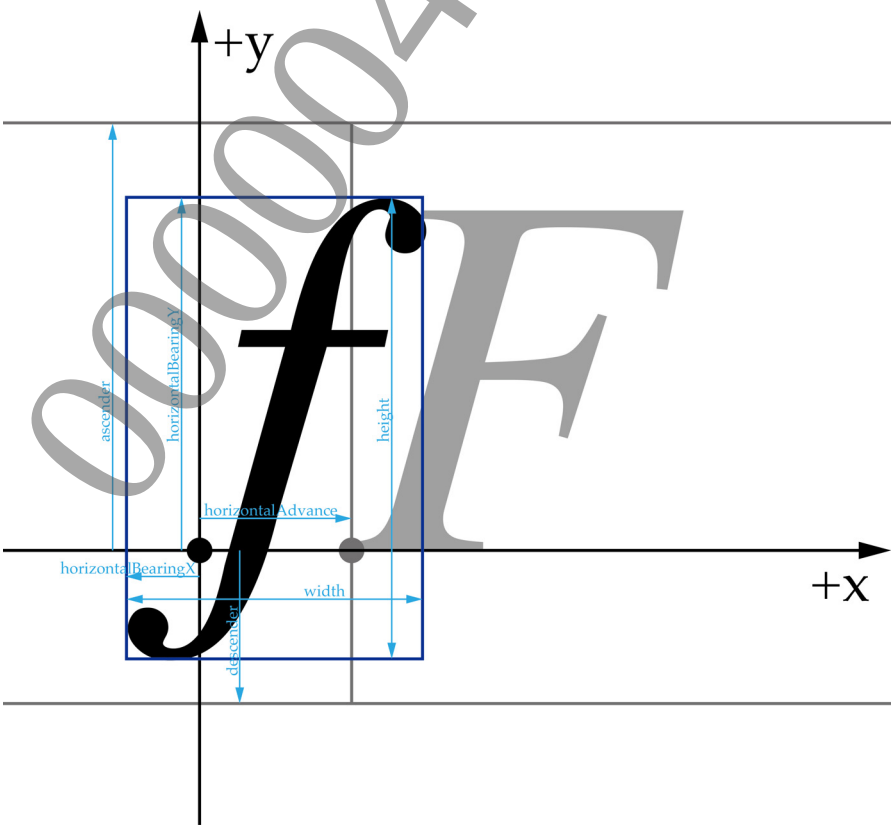


Figure 2 Glyph Metrics for Vertical Typesetting

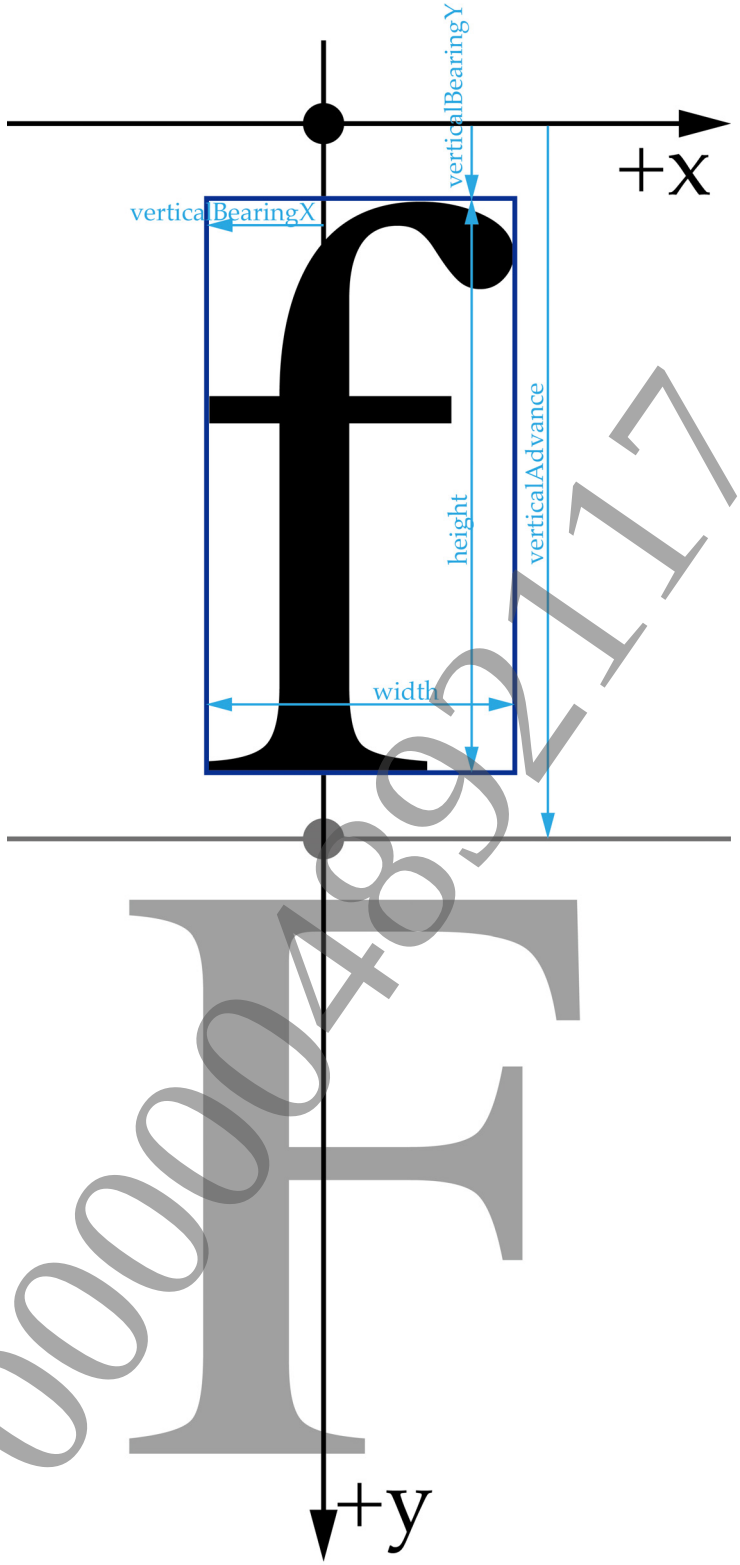


Figure 3 Metrics Details for Horizontal Typesetting and Vertical Typesetting

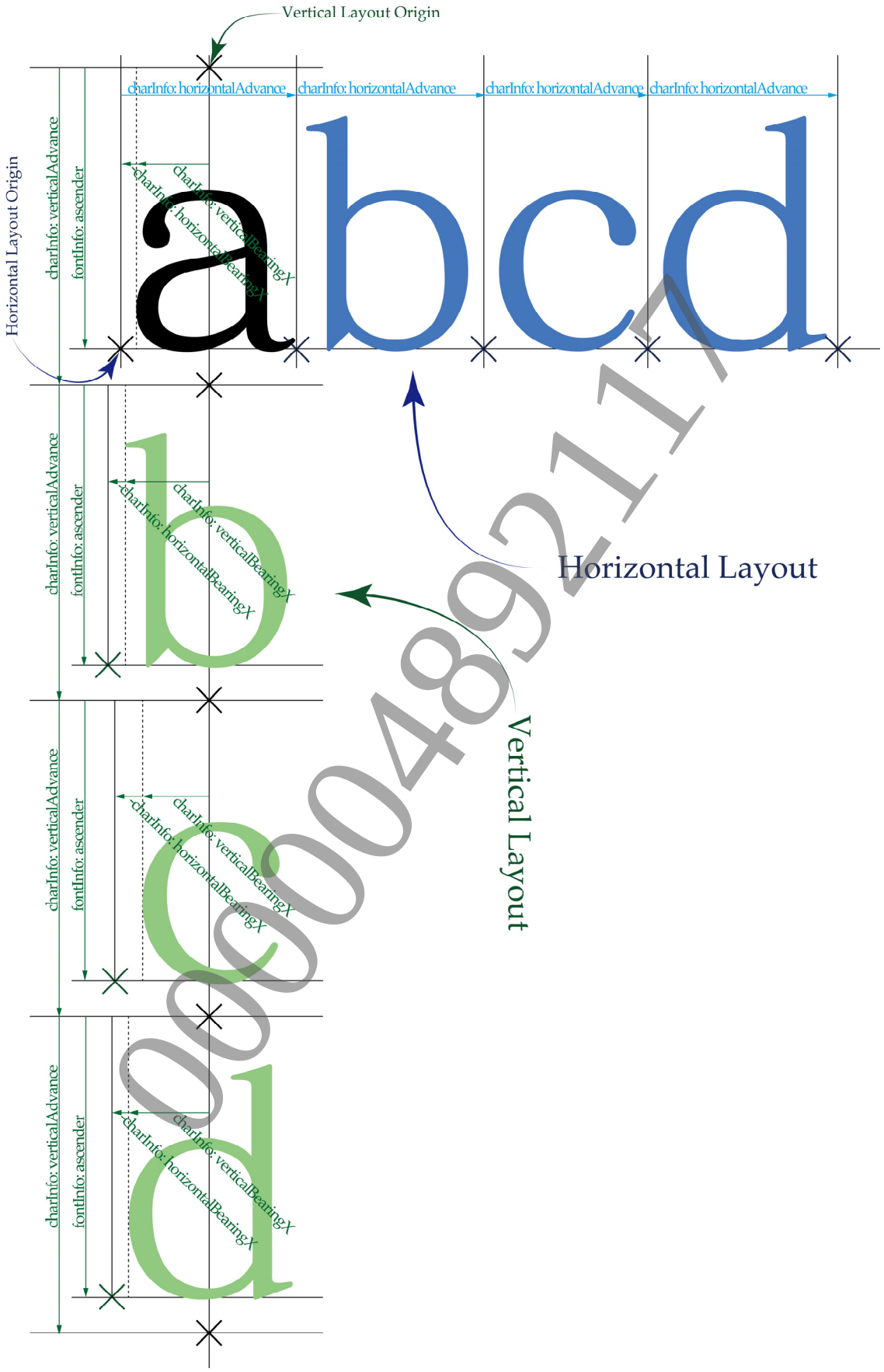
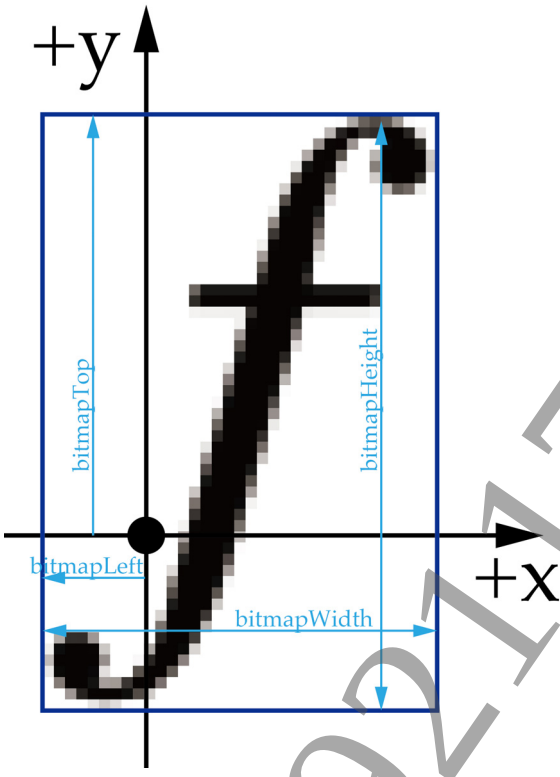


Figure 4 Bitmap Information Contained in ScePvf_t_charInfo



Name of Value	Meaning
bitmapWidth	Integer value that indicates the horizontal size of bitmap data in pixel units
bitmapHeight	Integer value that indicates the vertical size of bitmap data in pixel units
bitmapLeft	Integer value indicating the distance in pixel units from the character reference point to the left edge of the bitmap. This value indicates the position of the character's origin on the bitmapped glyph image along the X-axis. This value is processed within libpvf. The application program does not need to handle it.
bitmapTop	Integer value indicating the distance in pixel units from the character reference point to the top edge of the bitmap. This value indicates the position of the character's origin on the bitmapped glyph image along the Y-axis. This value is processed within libpvf. The application program does not need to handle it.

Notes

The numeric value indicating the position of the character's origin for vertical typesetting layout of a bitmapped glyph image is not saved.

When characters are composed vertically, do not expect to place subpixels of bitmapped glyph images precisely. Instead, use the *verticalBearingX*, *verticalBearingY*, and *verticalAdvance* values of `ScePvf_t_iGlyphMetricsInfo` or `ScePvf_t_fGlyphMetricsInfo` to place bitmapped glyph images.

The Japanese fonts and Chinese fonts in the PlayStation®Vita system internal vector fonts, have vertical layout information.

On the other hand, the Latin fonts and Korean fonts do not have vertical layout information. The characters will therefore not be placed properly when laid out vertically using the aforementioned method.

Figure 5　Examples of Japanese/Latin/Korean/Chinese Fonts Laid Out Vertically

4 Transition from libpgf

Differences between libpgf and libpvf

libpgf is the library designed to handle grayscale dot fonts, whereas libpvf is designed to handle vector fonts. Although vector fonts have very different format from grayscale dot fonts, libpvf contains an interface that is compatible with libpgf.

To ensure a smooth transition from libpgf, structure names, function names, and constant names of libpvf are defined based on systematic renaming rules, as shown in the following table.

libpgf	libpvf
`SceFont*`	`ScePvf*`
`sceFont*`	`scePvf*`
`SCE_FONT*`	`SCE_PVF*`

Differences in Main Structures

There are differences in some of the structure members between libpgf and libpvf. The following tables show how members of the main libpvf structures (`ScePvf*`) differ from those of the equivalent libpgf structures (`SceFont*`).

Refer to the "libpvf Reference" document for further information on the added members.

ScePvf_t_initRec (libpgf: SceFont_t_initRec)

Member Name	Difference
openFunc	Deleted
closeFunc	Deleted
readFunc	Deleted
seekFunc	Deleted
onErrorFunc	Deleted
whenDoneReadFunc	Deleted
reallocFunc	**Added**

ScePvf_t_cacheSystemInterface (libpgf: SceFont_t_cacheSystemInterface)

Member Name	Difference
write2ToCacheFunc	Deleted
write3ToCacheFunc	Deleted
read2FromCacheFunc	Deleted
read3FromCacheFunc	Deleted

ScePvfCacheKey (libpgf: SceFontCacheKey)

Member Name	Difference
keyValue4	**Added**
keyValue5	**Added**
keyValue6	**Added**
keyValue7	**Added**
keyValue8	**Added**

ScePvf_t_fontStyleInfo (libpgf: SceFont_t_fontStyleInfo)

Member Name	Difference
hSize	Deleted
vSize	Deleted
hResolution	Deleted
vResolution	Deleted
styleName	**Added**

ScePvf_t_fontInfo (libpgf: SceFont_t_fontInfo)

Member Name	Difference
maxGlyphBitmapWidth	Deleted
maxGlyphBitmapHeight	Deleted
numSubChars	Deleted
pixelDepth	Deleted

Functions Added to libpvf

Functions added to libpvf are shown below.

Added Functions	Description
scePvfSetEM()	Sets em square value
scePvfSetCharSize()	Sets rasterized character size
scePvfSetEmboldenRate()	Sets embolden (unembolden) rate
scePvfSetSkewValue()	Sets skew value
scePvfIsElement()	Checks whether or not the glyph of a specific character code within a specific font exists

Refer to the "libpvf Reference" document for further information on the added functions.

Notes

The following table outlines other points to note regarding implementation.

Issue	Difference
Handling of *bitmapLeft* and *bitmapTop* of SceFont_t_charInfo/ ScePvf_t_charInfo when setting the *xPos64* and *yPos64* values of SceFont_t_userImageBufferRec/ ScePvf_t_userImageBufferRec	With libpgf, the application program must explicitly add *bitmapLeft* and *-bitmapTop* values multiplied by 64 to the *xPos64* and *yPos64* values respectively upon setting. With libpvf, this process is performed within the library, so the application program does not need to take *bitmapLeft* and *bitmapTop* into consideration.

Refer to sample programs of each library for examples.

5 Pre-Installed Fonts

Fonts Pre-Installed on PlayStation®Vita

The vector fonts listed in the table below are pre-installed on PlayStation®Vita. All of these fonts can be accessed with libpvf.

Index Number	Application Language	Font Type	Style	Note
0	Japanese	Gothic	Regular	
1	Japanese	Ming	Regular	
2	Japanese	Gothic	Boldface	
3	Japanese	Ming	Boldface	
4	English or other alphabet-based language	Gothic	Regular	Proportional Cyrillic letters are also recorded
5	English or other alphabet-based language	Roman	Regular	Proportional Cyrillic letters are also recorded
6	English or other alphabet-based language	Gothic	Oblique	Proportional Cyrillic letters are also recorded
7	English or other alphabet-based language	Roman	Oblique	Proportional Cyrillic letters are also recorded
8	English or other alphabet-based language	Gothic	Boldface	Proportional Cyrillic letters are also recorded
9	English or other alphabet-based language	Roman	Boldface	Proportional Cyrillic letters are also recorded
10	English or other alphabet-based language	Gothic	Bold oblique	Proportional Cyrillic letters are also recorded
11	English or other alphabet-based language	Roman	Bold oblique	Proportional Cyrillic letters are also recorded
12	Korean	Gothic	Regular	Hangul and the won symbol are recorded
13	Korean	Ming	Regular	Hangul and the won symbol are recorded
14	Korean	Gothic	Boldface	Hangul and the won symbol are recorded
15	Korean	Ming	Boldface	Hangul and the won symbol are recorded
16	Chinese	Gothic	Regular	Traditional and simplified characters are recorded.
17	Chinese	Ming	Regular	Traditional and simplified characters are recorded.

6 Notes

Handling of Numerical Value Information and Bitmap Information Obtained from libpvf

Use Other than on the PlayStation®Vita

Use of the numerical value information and of the bitmap information resulting from rasterization processing obtained using libpvf APIs is not allowed for purposes other than use in application programs running on the PlayStation®Vita.

This restriction does not apply, i.e. use is possible in the case of indirect use, such as for screenshot printing media relating to said application programs.

Saving Within Save Data and Reusing

With regard to Japanese fonts, Latin fonts and Korean fonts, saving within save data and reuse at a later time are both allowed for the numerical value information and the bitmap information resulting from rasterization processing obtained using libpvf APIs.

Neither saving within save data nor reuse at a later time is allowed for Chinese fonts.

Non-Runtime Use

In the development process of an application program, it is not allowed to reuse static data/a part of the application program, which has been generated by the development tool through the obtainment of numerical value information and bitmap information resulting from rasterization processing by using libpvf APIs.

	Use Other than on PlayStation®Vita	Saving within Save Data and Reuse	Non-runtime Use
Japanese fonts	No	Yes	No
Latin fonts	No	Yes	No
Korean fonts	No	Yes	No
Chinese fonts	No	No	No