libpgf 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 libpgf library provides PSP™-compatible grayscale dot font glyph images to an application program.

By using libpgf, an application program can get information about PSP™-compatible grayscale dot 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.

libpgf also enables an application program to get glyph image data for specific characters included in specific fonts that are part of PSP™-compatible grayscale dot font groups installed on the PlayStation®Vita.

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

libpgf 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 libpgf.

Filename	Description
font/libpgf.h	Header file
libScePgf_stub.a	Stub library file
libScePgf_stub_weak.a	Weak import stub library file

Sample Programs

The following sample programs are provided with libpgf.

samples/sample_code/graphics/api_libpgf/basho

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

samples/sample_code/graphics/api_libpgf/fontcache

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

samples/sample_code/graphics/api_libpgf/simple

Exemplifies the simplest program created using the libpgf library.

2 Usage Procedure

Library Initialization

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

The `sceFontNewLib()` function enables an application program to use multiple instances of the library. However, if libpgf is used as an ordinary library, it is not meaningful to have multiple instances of the library.

Finding a Font

The number of PSP™-compatible grayscale dot fonts that are installed on the PlayStation®Vita can be obtained by using the `sceFontGetNumFontList()` function. Detailed font design information can be obtained by using the `sceFontGetFontList()` function, and a font can be found by using the `sceFontFindOptimumFont()` or `sceFontFindFont()` function.

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

Opening a Font

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

An application program can also open a font by using `sceFontOpenUserFile()` or `sceFontOpenUserMemory()`.

Getting Character Metrics

Character metrics information can be obtained by calling the `sceFontGetCharInfo()` 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 `sceFontGetCharGlyphImage()` 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 `sceFontClose()` function and providing it with the font ID that was obtained from the `sceFontOpen()` function when the font was opened.

Library Termination

A library instance is closed by calling the `sceFontDoneLib()` function and providing it with the library ID that was obtained from the `sceFontNewLib()` 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/libpgf.h>

/* Memory allocation function */
static SceFont_t_pointer cb_Alloc
(
 SceFont_t_pointer pMyData, /* Pointer to user data */
 SceFont_t_u32 size /* Requested size */
 )
{
    return ( malloc (size) );
}

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

/* libpgf test function */
void myTest_libpgf ( void )
{
    SceFont_t_error errorCode;
    SceFont_t_libId libID;
    SceFont_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_Free,    /* Memory free function */
       NULL, /* Open (NULL is OK if sceFontOpenUserFile is not used) */
       NULL, /* Close (NULL is OK if sceFontOpenUserFile is not used) */
       NULL, /* Read (NULL is OK if sceFontOpenUserFile is not used) */
       NULL, /* Seek (NULL is OK if sceFontOpenUserFile is not used) */
       NULL, /* Reserved area */
       NULL, /* Reserved area */
    };
    SceFont_t_int numFontList;
    SceFont_t_fontStyleInfo aFontStyleInfoList [30];
    SceFont_t_fontId fontID;
    SceFont_t_fontInfo fontInfo;
    SceFont_t_charCode charCode=(0x795d);
    SceFont_t_charInfo charInfo;
    SceFont_t_userImageBufferRec myImage;
#define MYIMAGE_WIDTH (128)
```

```
#define MYIMAGE_HEIGHT (128)
    unsigned char *myImageBuffer [(MYIMAGE_WIDTH / 2) * MYIMAGE_HEIGHT];


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

    /* Get number of installed fonts */
    numFontList = sceFontGetNumFontList (libID, &errorCode);
    if ( errorCode != SCE_OK ) ; /* Error handling */
    fprintf (stderr, "numFontList=%d\n", (int)numFontList);

    /* Get font list */
    errorCode = sceFontGetFontList (libID, aFontStyleInfoList, numFontList);
    if ( errorCode != SCE_OK ) ; /* Error handling */

    /* Open font */
    fontID = sceFontOpen (libID, 0, SCE_FONT_FILEBASEDSTREAM, &errorCode);
    if ( errorCode != SCE_OK ) ; /* Error handling */

    /* Get information about opened font */
    errorCode = sceFontGetFontInfo (fontID, &fontInfo);
    if ( errorCode != SCE_OK ) ; /* Error handling */

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

    myImage.pixelFormat = SCE_FONT_USERIMAGE_DIRECT4_L;
    myImage.rect.width = MYIMAGE_WIDTH;
    myImage.rect.height = MYIMAGE_HEIGHT;
    myImage.bytesPerLine = MYIMAGE_WIDTH / 2;
    myImage.reserved = 0;
    myImage.buffer = (SceFont_t_u8 *)myImageBuffer;
    myImage.xPos64 = 20;
    myImage.yPos64 = 40;
    /* Get character glyph images */
    errorCode = sceFontGetCharGlyphImage (fontID, charCode, &myImage);
    if ( errorCode != SCE_OK ) ; /* Error handling */

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

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

    return;
}

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

3 Explanation of Operation

libpgf Internal Processing

sceFontNewLib()

This function accesses the PlayStation®Vita system registry to get information related to installed PSP™-compatible grayscale dot fonts and copies that information to a memory area that is managed by the library instance.

sceFontFindFont()

This function compares search information with information obtained from the system registry to find a desired font and returns its number.

sceFontFindOptimumFont()

This function compares search information with information obtained from the system registry to find a font that most closely matches the requested font attributes and returns its number.

sceFontOpen()

This function opens the font having the specified number and returns its font ID. That font ID will be owned by the library instance specified in the argument.

sceFontClose()

This function closes the font having the specified font ID.

sceFontDoneLib()

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 libpgf

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

Information	Description
Information related to the installed font group	The number of PSP™-compatible grayscale dot fonts installed on the PlayStation®Vita and related information (`SceFont_t_fontStyleInfo`, `SceFont_t_fontInfo`) can be obtained.
Information related to a specific font in the installed font group	Information that is common to all characters included in the specified PSP™-compatible grayscale dot font installed on the PlayStation®Vita (`SceFont_t_iGlyphMetricsInfo`, `SceFont_t_fGlyphMetricsInfo`) can be obtained.
Information related to a specific character of a specific font of the installed font group	Glyph information for a specific character that is part of a specific PSP™-compatible grayscale dot font installed on the PlayStation®Vita (`SceFont_t_charInfo`) can be obtained.

Glyph Metrics Information

Glyph metrics information (`SceFont_t_iGlyphMetricsInfo`, `SceFont_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. A positive value is taken upwards. 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. A positive value is taken upwards. 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.
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 downwards. 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 downwards. 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 `SceFont_t_iGlyphMetricsInfo` whose names end in "64" contain values that are 64 times that of members whose names do not end in "64."

Bitmapped 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 `SceFont_t_userImageBufferRec`. The application copies the glyph image to the buffer by calling the `sceFontGetCharGlyphImage()` 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 SceFont_t_fGlyphMetricsInfo (Breakdown of SceFont_t_iGlyphMetricsInfo)

Figure 1 Glyph metrics for horizontal typesetting

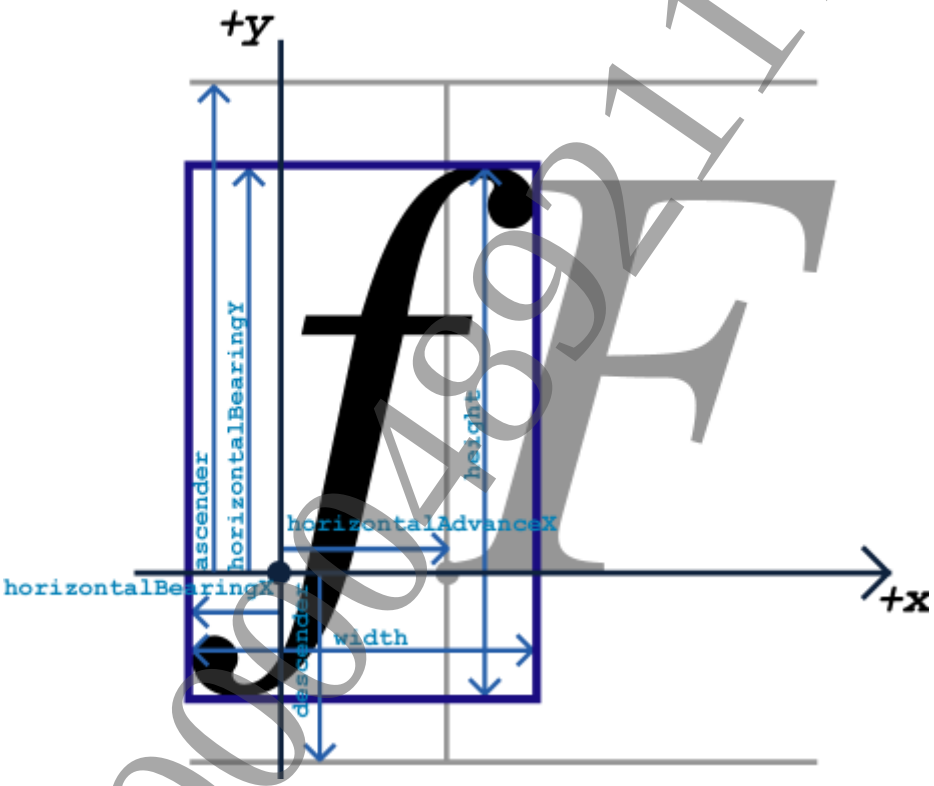


Figure 2 Glyph metrics for vertical typesetting

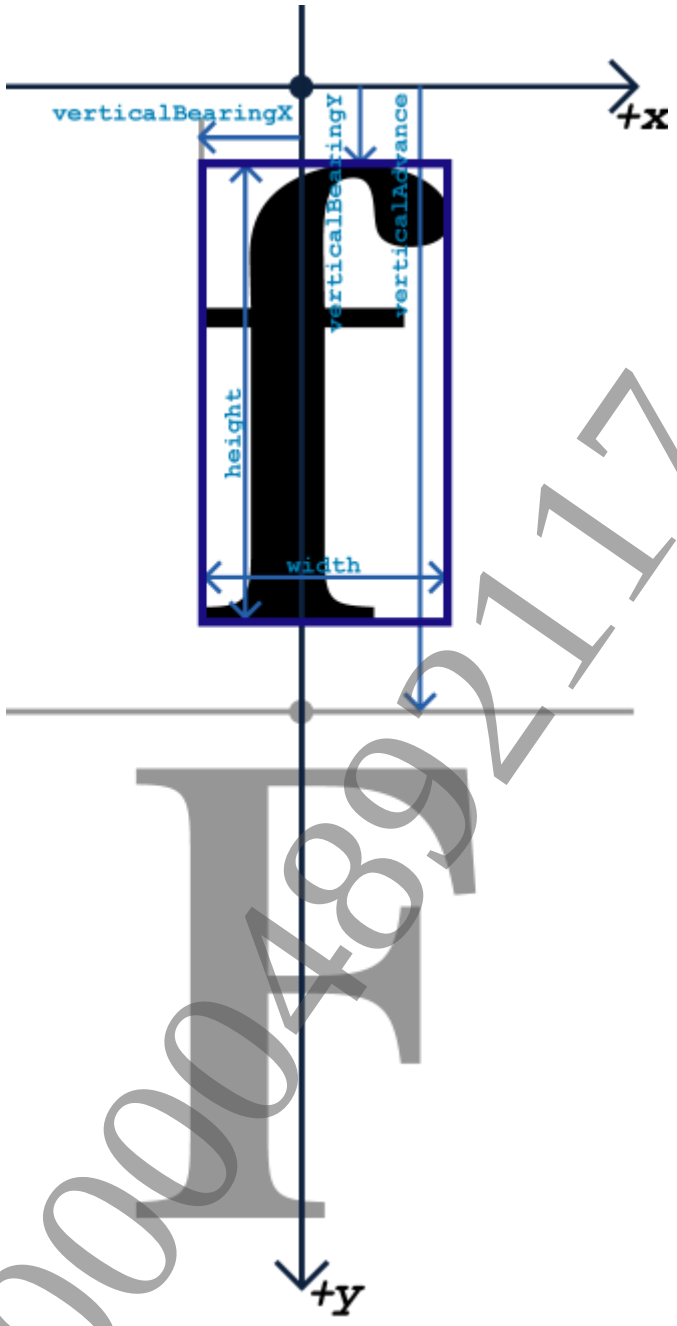
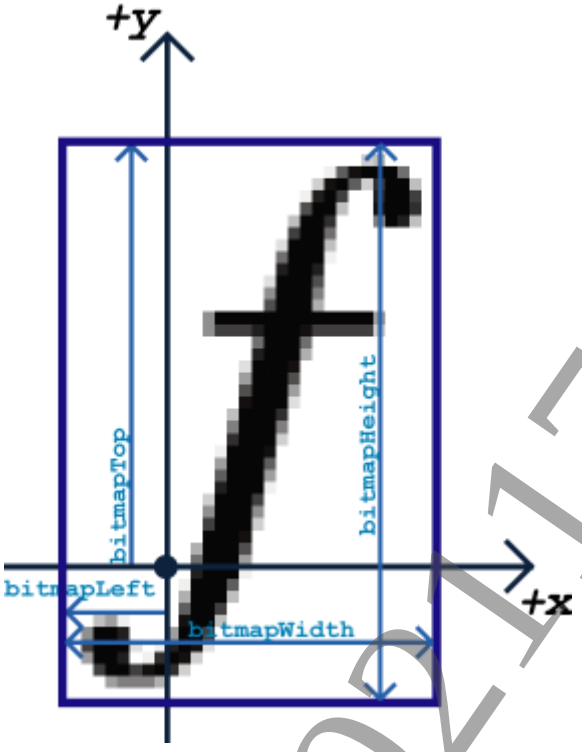


Figure 3 Bitmap information contained in SceFont_t_charInfo



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.
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.

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 `SceFont_t_iGlyphMetricsInfo` or `SceFont_t_fGlyphMetricsInfo` to place bitmapped glyph images.

4 Pre-Installed Fonts

Fonts Pre-Installed on PlayStation®Vita

The PSP™-compatible grayscale dot font listed in the table below are pre-installed on PlayStation®Vita. All of these fonts can be accessed with libpgf.

"Size" in the table refers to the point size with a resolution of 128 dpi.

Index Number	Application Language	Font Type	Style	Size	Note
0	Japanese	Gothic	Regular	10.125 pt	
1	English or other alphabet-based language	Gothic	Regular	10.125 pt	Proportional Cyrillic letters are also recorded
2	English or other alphabet-based language	Roman	Regular	10.125 pt	Proportional Cyrillic letters are also recorded
3	English or other alphabet-based language	Gothic	Oblique	10.125 pt	Proportional Cyrillic letters are also recorded
4	English or other alphabet-based language	Roman	Oblique	10.125 pt	Proportional Cyrillic letters are also recorded
5	English or other alphabet-based language	Gothic	Boldface	10.125 pt	Proportional Cyrillic letters are also recorded
6	English or other alphabet-based language	Roman	Boldface	10.125 pt	Proportional Cyrillic letters are also recorded
7	English or other alphabet-based language	Gothic	Bold oblique	10.125 pt	Proportional Cyrillic letters are also recorded
8	English or other alphabet-based language	Roman	Bold oblique	10.125 pt	Proportional Cyrillic letters are also recorded
9	English or other alphabet-based language	Gothic	Regular	7.0 pt	Proportional Cyrillic letters are also recorded
10	English or other alphabet-based language	Roman	Regular	7.0 pt	Proportional Cyrillic letters are also recorded
11	English or other alphabet-based language	Gothic	Oblique	7.0 pt	Proportional Cyrillic letters are also recorded
12	English or other alphabet-based language	Roman	Oblique	7.0 pt	Proportional Cyrillic letters are also recorded
13	English or other alphabet-based language	Gothic	Boldface	7.0 pt	Proportional Cyrillic letters are also recorded

Index Number	Application Language	Font Type	Style	Size	Note
14	English or other alphabet-based language	Roman	Boldface	7.0 pt	Proportional Cyrillic letters are also recorded
15	English or other alphabet-based language	Gothic	Bold oblique	7.0 pt	Proportional Cyrillic letters are also recorded
16	English or other alphabet-based language	Roman	Bold oblique	7.0 pt	Proportional Cyrillic letters are also recorded
17	Korean	Gothic	Regular	10.125pt	Hangul and the won symbol are recorded.