

## GRRLIB

### 4.4.1

Generated by Doxygen 1.8.17



<b>1 GRRLIB Documentation</b>	<b>1</b>
1.1 Introduction	1
1.2 Links	1
1.3 Credits	1
1.4 Licence	1
<b>2 LICENCE</b>	<b>3</b>
<b>3 Changelog</b>	<b>5</b>
<b>4 Module Index</b>	<b>9</b>
4.1 Modules	9
<b>5 Data Structure Index</b>	<b>11</b>
5.1 Data Structures	11
<b>6 File Index</b>	<b>13</b>
6.1 File List	13
<b>7 Module Documentation</b>	<b>15</b>
7.1 Everything in GRRLIB	15
7.1.1 Detailed Description	20
7.1.2 Macro Definition Documentation	20
7.1.2.1 RGBA	21
7.1.3 Enumeration Type Documentation	21
7.1.3.1 GRRLIB_blendMode	21
7.1.4 Function Documentation	21
7.1.4.1 GRRLIB_3dMode()	21
7.1.4.2 GRRLIB_BMFX_Blur()	22
7.1.4.3 GRRLIB_BMFX_FlipH()	22
7.1.4.4 GRRLIB_BMFX_FlipV()	23
7.1.4.5 GRRLIB_BMFX_Grayscale()	23
7.1.4.6 GRRLIB_BMFX_Invert()	23
7.1.4.7 GRRLIB_BMFX_Pixelate()	24
7.1.4.8 GRRLIB_BMFX_Scatter()	24
7.1.4.9 GRRLIB_BMFX_Sepia()	25
7.1.4.10 GRRLIB_Camera3dSettings()	25
7.1.4.11 GRRLIB_Circle()	26
7.1.4.12 GRRLIB_ClearTex()	26
7.1.4.13 GRRLIB_ClipDrawing()	27
7.1.4.14 GRRLIB_CompEnd()	27
7.1.4.15 GRRLIB_CompStart()	27
7.1.4.16 GRRLIB_CreateEmptyTexture()	28
7.1.4.17 GRRLIB_DrawCone()	28

---

7.1.4.18	GRRLIB_DrawCube()	28
7.1.4.19	GRRLIB_DrawCylinder()	29
7.1.4.20	GRRLIB_DrawImg()	29
7.1.4.21	GRRLIB_DrawImgQuad()	30
7.1.4.22	GRRLIB_DrawPart()	30
7.1.4.23	GRRLIB_DrawSphere()	31
7.1.4.24	GRRLIB_DrawTessPanel()	31
7.1.4.25	GRRLIB_DrawTile()	32
7.1.4.26	GRRLIB_DrawTileQuad()	32
7.1.4.27	GRRLIB_DrawTorus()	33
7.1.4.28	GRRLIB_Exit()	33
7.1.4.29	GRRLIB_FillScreen()	33
7.1.4.30	GRRLIB_FlushTex()	34
7.1.4.31	GRRLIB_FreeBMF()	34
7.1.4.32	GRRLIB_FreeTexture()	34
7.1.4.33	GRRLIB_FreeTTF()	35
7.1.4.34	GRRLIB_GeckoInit()	35
7.1.4.35	GRRLIB_GeckoPrintf()	35
7.1.4.36	GRRLIB_GetAntiAliasing()	36
7.1.4.37	GRRLIB_GetBlend()	36
7.1.4.38	GRRLIB_GetPixelFromFB()	36
7.1.4.39	GRRLIB_GetPixelFromtexImg()	37
7.1.4.40	GRRLIB_GXEngine()	37
7.1.4.41	GRRLIB_Init()	37
7.1.4.42	GRRLIB_InitTileSet()	38
7.1.4.43	GRRLIB_Line()	38
7.1.4.44	GRRLIB_LoadBMF()	39
7.1.4.45	GRRLIB_LoadFile()	39
7.1.4.46	GRRLIB_LoadTexture()	40
7.1.4.47	GRRLIB_LoadTextureBMP()	40
7.1.4.48	GRRLIB_LoadTextureFromFile()	40
7.1.4.49	GRRLIB_LoadTextureJPG()	41
7.1.4.50	GRRLIB_LoadTextureJPGEx()	41
7.1.4.51	GRRLIB_LoadTexturePNG()	42
7.1.4.52	GRRLIB_LoadTTF()	42
7.1.4.53	GRRLIB_NGone()	43
7.1.4.54	GRRLIB_NGoneFilled()	43
7.1.4.55	GRRLIB_NPlot()	43
7.1.4.56	GRRLIB_ObjectView()	44
7.1.4.57	GRRLIB_ObjectViewInv()	44
7.1.4.58	GRRLIB_ObjectViewRotate()	45
7.1.4.59	GRRLIB_ObjectViewScale()	45

7.1.4.60 GRRLIB_ObjectViewTrans()	45
7.1.4.61 GRRLIB_Plot()	46
7.1.4.62 GRRLIB_PrintBMF()	46
7.1.4.63 GRRLIB_Printf()	47
7.1.4.64 GRRLIB_PrintfTTF()	47
7.1.4.65 GRRLIB_PrintfTTFW()	48
7.1.4.66 GRRLIB_PtInRect()	48
7.1.4.67 GRRLIB_Rectangle()	49
7.1.4.68 GRRLIB_RectInRect()	49
7.1.4.69 GRRLIB_RectOnRect()	50
7.1.4.70 GRRLIB_Screen2Texture()	50
7.1.4.71 GRRLIB_ScrShot()	51
7.1.4.72 GRRLIB_SetAntiAliasing()	51
7.1.4.73 GRRLIB_SetBackgroundColour()	51
7.1.4.74 GRRLIB_SetBlend()	52
7.1.4.75 GRRLIB_SetHandle()	52
7.1.4.76 GRRLIB_SetLightAmbient()	52
7.1.4.77 GRRLIB_SetLightDiff()	53
7.1.4.78 GRRLIB_SetLightSpec()	53
7.1.4.79 GRRLIB_SetLightSpot()	53
7.1.4.80 GRRLIB_SetMidHandle()	54
7.1.4.81 GRRLIB_SetPixelToFB()	54
7.1.4.82 GRRLIB_SetPixelToTexImg()	55
7.1.4.83 GRRLIB_SetTexture()	55
7.1.4.84 GRRLIB_WidthTTF()	56
7.1.4.85 GRRLIB_WidthTTFW()	56
<b>8 Data Structure Documentation</b>	<b>57</b>
8.1 GRRLIB_bytemapChar Struct Reference	57
8.1.1 Detailed Description	57
8.1.2 Field Documentation	57
8.1.2.1 data	58
8.1.2.2 height	58
8.1.2.3 relx	58
8.1.2.4 rely	58
8.1.2.5 width	58
8.2 GRRLIB_bytemapFont Struct Reference	59
8.2.1 Detailed Description	59
8.2.2 Field Documentation	59
8.2.2.1 name	60
8.2.2.2 nbChar	60
8.2.2.3 palette	60

---

8.2.2.4 version	60
8.3 GRRLIB_drawSettings Struct Reference	60
8.3.1 Detailed Description	61
8.3.2 Field Documentation	61
8.3.2.1 blend	61
8.3.2.2 lights	61
8.4 GRRLIB_Font Struct Reference	61
8.4.1 Detailed Description	62
8.5 GRRLIB_texImg Struct Reference	62
8.5.1 Detailed Description	63
8.5.2 Field Documentation	63
8.5.2.1 nbtileh	63
8.5.2.2 nbtilew	63
8.5.2.3 tiledtex	63
8.5.2.4 tileh	63
8.5.2.5 tilestart	64
8.5.2.6 tilew	64
8.5.2.7 w	64
<b>9 File Documentation</b>	<b>65</b>
9.1 grrlib.h File Reference	65
9.1.1 Detailed Description	66
9.2 GRRLIB__inline.h File Reference	67
9.2.1 Detailed Description	68
9.3 GRRLIB__lib.h File Reference	68
9.3.1 Detailed Description	71
<b>10 Example Documentation</b>	<b>73</b>
10.1 template/source/main.c	73
<b>Index</b>	<b>75</b>

# Chapter 1

## GRRLIB Documentation

Welcome to the GRRLIB documentation. A complete list of functions is available from this [page](#).

### 1.1 Introduction

GRRLIB is a C/C++ 2D/3D graphics library for Wii application developers. It is essentially a wrapper which presents a friendly interface to the Nintendo GX core.

### 1.2 Links

Code: <https://github.com/GRRLIB/GRRLIB>  
Discussions: <http://grrlib.santo.fr/forum>  
Chat: [#GRRLIB](#) on EFnet

### 1.3 Credits

Project Leader : NoNameNo  
Documentation : Crayon, BlueChip  
Lead Coder : NoNameNo  
Support Coders : Crayon, Xane, DragonMinded, BlueChip  
Advisors : RedShade, Jespa

### 1.4 Licence

See the [LICENCE](#) file for licence rights and limitations (MIT).





## Chapter 2

# LICENCE

Copyright (c) 2009-2021 The GRRLIB Team

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.



## Chapter 3

# Changelog

All notable changes to this project will be documented in this file.

### Unreleased

- Fixed compatibility issues with devkitPPC release 39.

### 4.4.1 - 2021-03-05

- Patched widescreen on Wii U.

### 4.4.0 - 2020-01-12

- Removed libpng, zlib, libfreetype and libjpeg from the project. These libraries should now be installed in devkitPro with pacman.
- GRRLIB and pngu are now installed into the *portlibs* folder instead of the *libogc* folder.
- Removed `uint` declaration. The `u32` type should be used instead.
- The `GRRLIB_CreateEmptyTexture()` function is not inline anymore.

### 4.3.2 - 2012-08-12

- libpng was updated to version 1.5.12
- zlib was updated to version 1.2.7
- libjpeg was updated to version 8d
- FreeType updated to version 2.4.10
- `GRRLIB_ClampVar8()` was removed from GRRLIB
- Removed warnings from `GRRLIB_LoadBMF()`

### 4.3.1 - 2010-10-22

- libpng was updated to version 1.4.4
- FreeType updated to version 2.4.3
- GRRLIB is compatible with libogc 1.8.4

### 4.3.0 - 2010-06-28

- libpng was updated to version 1.4.2
- libjpeg was updated to version 8b
- zlib was updated to version 1.2.5
- FreeType 2.3.12 support (first support)
- 3D primitive drawing: torus, sphere, cube, cylinder and cone
- Light functions (diffuse, specular, spot) + ambient
- Split functions for rotation/translation/scaling 3d object.
- Here is the list a new added samples demo code:
  - 3d\_light1 -> Simple Diffuse light sample code
  - 3d\_light2 -> Simple Lights and `GRRLIB_ObjectViewInv()` sample
  - 3d\_light3 -> A little Specular light sample code
  - 3d\_light4 -> Spot Light Sample Code
  - 3D\_sample5 -> Simple demo of rotation/translation/scaling 3d object.
  - ttf -> TrueType Font demo

### 4.2.0 - 2009-12-16

- First support to 3D functions
- `GRRLIB_CompoStart()` and `GRRLIB_CompoEnd()` for real GX compositing with transparency support
- `GRRLIB_Screen2Texture()` is now fully optimized
- USB\_Gecko output facilities
- `GRRLIB_Compose()` was deleted since it was not fully using GX
- `GRRLIB_GetColor()` was deleted, the RGBA macro should be used instead
- Lot of new sample code:
  - 3D\_CubedTileDemo (How to use dynamic texturing)
  - 3D\_sample1 (A simple rotating flat cube)
  - 3D\_sample2 (A simple rotating textured cube)
  - 3D\_sample3 (A textured cube and compositing)
  - 3D\_sample4 (A complex object rotating)
  - basic\_drawing (How to use some basic GRRLIB functions)
  - bitmap\_fx (Effects ShowRoom)

- 
- blending (How to use blending mode)
  - compositing (A simple compositing how to)
  - funsin (A gradient sinusoid dancing)
  - particle (A nice particle sample code)
  - template (Use this as a basis for your project)
  - TileDemo (This will show you how to use tiles/tileset and map)
  - unlimited2d (A faky technic for unlimited sprites)
  - unlimited3d (Same as above but with 3D)

### 4.1.1 - 2009-11-24

- Fully compatible with devkitPro release 18 and 19 (code and examples)
- libpng was updated to version 1.2.40
- Support for MS-Windows Bitmap format uncompressed (1-bit, 4-bit, 8-bit, 24-bit and 32-bit)
- New function called `GRRLIB_DrawPart()` to draw a specific part of a texture
- Extra parameters to `GRRLIB_Screen2Texture()`
- Video is now initialized even without a SD card
- Fixed a problem with scaling images
- Documentation improvement

### 4.1.0 - 2009-10-05

- Completely new file structure with sub-folders
- Installer for vendor libraries (jpeg, png, pngu)
- Makefile for GRRLIB
- Many (all suitable) functions are now inlined
- Support for the recent changes to libogc
- Alpha compositor function added
- Correct use of pointers (no more struct passing)
- Speed improvement, bug fixing and more...

### 4.0.0 - 2009-03-05

- Color format changed to RGBA for ALL GRRLib functions to fit to `GXColor` format and use `GX_Color1u32`
- `GRRLIB_LoadTexture()` now auto detect PNG or JPEG
- GRRLib introduce a new texture structure (easier to handle texture width, height, etc ...)
- Add `GRRLIB_InitTileSet()` to initialize a tile set
- `GRRLIB_DrawImg()` recoded for simpler use

- `GRRLIB_DrawTile()` recoded for simpler use
- `InitVideo()` and `GRRLIB_Start()` merge into `GRRLIB_Init()`.
- Add `GRRLIB_PtInRect()`, `GRRLIB_RectInRect()` and `GRRLIB_RectOnRect()` to detect hot zone
- `GRRLIB_GetPixelFromtexImg()` and `GRRLIB_SetPixelTotexImg()` to directly read/write in texture
- `GRRLIB_CreateEmptyTexture()` and `GRRLIB_FlushTex()`
- New Bitmap FX
- Add `GRRLIB_Exit()` to free the memory allocated by GRRLIB

# Chapter 4

## Module Index

### 4.1 Modules

Here is a list of all modules:

Everything in GRRLIB . . . . . 15





# Chapter 5

## Data Structure Index

### 5.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">GRRLIB_bytemapChar</a>	Structure to hold the bytemap character information . . . . .	57
<a href="#">GRRLIB_bytemapFont</a>	Structure to hold the bytemap font information . . . . .	59
<a href="#">GRRLIB_drawSettings</a>	Structure to hold the current drawing settings . . . . .	60
<a href="#">GRRLIB_Font</a>	Structure to hold the TTF information . . . . .	61
<a href="#">GRRLIB_texImg</a>	Structure to hold the texture information . . . . .	62



# Chapter 6

## File Index

### 6.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">grrlib.h</a> . . . . .	65
<a href="#">GRRLIB__inline.h</a> . . . . .	67
<a href="#">GRRLIB__lib.h</a> . . . . .	68



# Chapter 7

## Module Documentation

### 7.1 Everything in GRRLIB

This is the complete list of functions, structures, defines, typedefs, enumerations and variables you may want to used to make your homebrew with GRRLIB. You simply need to include [grrlib.h](#) in your project to have access to all of these.

#### Data Structures

- struct [GRRLIB\\_drawSettings](#)  
*Structure to hold the current drawing settings.*
- struct [GRRLIB\\_texImg](#)  
*Structure to hold the texture information.*
- struct [GRRLIB\\_bytemapChar](#)  
*Structure to hold the bytemap character information.*
- struct [GRRLIB\\_bytemapFont](#)  
*Structure to hold the bytemap font information.*
- struct [GRRLIB\\_Font](#)  
*Structure to hold the TTF information.*

#### Macros

- #define [GRRLIB\\_VER\\_STRING](#) "4.4.1"  
*Version information for GRRLIB.*
- #define [R\(c\)](#) (((c) >> 24) & 0xFF)  
*Extract red component of colour.*
- #define [G\(c\)](#) (((c) >> 16) & 0xFF)  
*Extract green component of colour.*
- #define [B\(c\)](#) (((c) >> 8) & 0xFF)  
*Extract blue component of colour.*
- #define [A\(c\)](#) ( (c) & 0xFF)  
*Extract alpha component of colour.*
- #define [RGBA](#)(r, g, b, a)  
*Build an RGB pixel from components.*

- #define [GRRLIB\\_BLEND\\_NONE](#) ([GRRLIB\\_BLEND\\_ALPHA](#))  
*Alias for [GRRLIB\\_BLEND\\_ALPHA](#).*
- #define [GRRLIB\\_BLEND\\_LIGHT](#) ([GRRLIB\\_BLEND\\_ADD](#))  
*Alias for [GRRLIB\\_BLEND\\_ADD](#).*
- #define [GRRLIB\\_BLEND\\_SHADE](#) ([GRRLIB\\_BLEND\\_MULTI](#))  
*Alias for [GRRLIB\\_BLEND\\_MULTI](#).*
- #define **GRR\_EXTERN** extern
- #define **GRR\_INIT**(v)
- #define **GRR\_INITS**(...)
- #define **INLINE** static inline

## Typedefs

- typedef enum [GRRLIB\\_blendMode](#) [GRRLIB\\_blendMode](#)  
*[GRRLIB](#) Blending Modes.*
- typedef struct [GRRLIB\\_drawSettings](#) [GRRLIB\\_drawSettings](#)  
*Structure to hold the current drawing settings.*
- typedef struct [GRRLIB\\_texImg](#) [GRRLIB\\_texImg](#)  
*Structure to hold the texture information.*
- typedef struct [GRRLIB\\_bytemapChar](#) [GRRLIB\\_bytemapChar](#)  
*Structure to hold the bytemap character information.*
- typedef struct [GRRLIB\\_bytemapFont](#) [GRRLIB\\_bytemapFont](#)  
*Structure to hold the bytemap font information.*
- typedef struct [GRRLIB\\_Font](#) [GRRLIB\\_ttfFont](#)  
*Structure to hold the TTF information.*

## Enumerations

- enum [GRRLIB\\_blendMode](#) {  
[GRRLIB\\_BLEND\\_ALPHA](#) = 0, [GRRLIB\\_BLEND\\_ADD](#) = 1, [GRRLIB\\_BLEND\\_SCREEN](#) = 2, [GRRLIB\\_BLEND\\_MULTI](#)  
= 3,  
[GRRLIB\\_BLEND\\_INV](#) = 4 }
- [GRRLIB](#) Blending Modes.*

## Functions

- **GRR\_EXTERN** void \*xfb[2] **GRR\_INITS** (NULL, NULL)
- **GRR\_EXTERN** u32 fb **GRR\_INIT** (0)
- **INLINE** void [GRRLIB\\_ClipReset](#) (void)  
*Reset the clipping to normal.*
- **INLINE** void [GRRLIB\\_ClipDrawing](#) (const int x, const int y, const int width, const int height)  
*Clip the drawing area to an rectangle.*
- **INLINE** bool [GRRLIB\\_PtInRect](#) (const int hotx, const int hoty, const int hotw, const int hoth, const int wpadx, const int wpady)  
*Determine whether the specified point lies within the specified rectangle.*
- **INLINE** bool [GRRLIB\\_RectInRect](#) (const int rect1x, const int rect1y, const int rect1w, const int rect1h, const int rect2x, const int rect2y, const int rect2w, const int rect2h)  
*Determine whether a specified rectangle lies within another rectangle.*
- **INLINE** bool [GRRLIB\\_RectOnRect](#) (const int rect1x, const int rect1y, const int rect1w, const int rect1h, const int rect2x, const int rect2y, const int rect2w, const int rect2h)

- Determine whether a part of a specified rectangle lies on another rectangle.*

  - `INLINE void GRRLIB_NPlot (const guVector v[], const u32 color[], const long n)`

*Draw an array of points.*
- `INLINE void GRRLIB_NGone (const guVector v[], const u32 color[], const long n)`

*Draw a polygon.*
- `INLINE void GRRLIB_NGoneFilled (const guVector v[], const u32 color[], const long n)`

*Draw a filled polygon.*
- `INLINE void GRRLIB_GXEngine (const guVector v[], const u32 color[], const long n, const u8 fmt)`

*Draws a vector.*
- `INLINE void GRRLIB_FillScreen (const u32 color)`

*Clear screen with a specific color.*
- `INLINE void GRRLIB_Plot (const f32 x, const f32 y, const u32 color)`

*Draw a dot.*
- `INLINE void GRRLIB_Line (const f32 x1, const f32 y1, const f32 x2, const f32 y2, const u32 color)`

*Draw a line.*
- `INLINE void GRRLIB_Rectangle (const f32 x, const f32 y, const f32 width, const f32 height, const u32 color, const bool filled)`

*Draw a rectangle.*
- `INLINE void GRRLIB_SetHandle (GRRLIB_texImg *tex, const int x, const int y)`

*Set a texture's X and Y handles.*
- `INLINE void GRRLIB_SetMidHandle (GRRLIB_texImg *tex, const bool enabled)`

*Center a texture's handles.*
- `INLINE u32 GRRLIB_GetPixelFromtexImg (const int x, const int y, const GRRLIB_texImg *tex)`

*Return the color value of a pixel from a GRRLIB\_texImg.*
- `INLINE void GRRLIB_SetPixelTotexImg (const int x, const int y, GRRLIB_texImg *tex, const u32 color)`

*Set the color value of a pixel to a GRRLIB\_texImg.*
- `INLINE u32 GRRLIB_GetPixelFromFB (int x, int y)`

*Reads a pixel directly from the FrontBuffer.*
- `INLINE void GRRLIB_SetPixelToFB (int x, int y, u32 pokeColor)`

*Writes a pixel directly from the FrontBuffer.*
- `INLINE void GRRLIB_SetBlend (const GRRLIB_blendMode blendmode)`

*Set a blending mode.*
- `INLINE GRRLIB_blendMode GRRLIB_GetBlend (void)`

*Get the current blending mode.*
- `INLINE void GRRLIB_SetAntiAliasing (const bool aa)`

*Turn anti-aliasing on/off.*
- `INLINE bool GRRLIB_GetAntiAliasing (void)`

*Get current anti-aliasing setting.*
- `INLINE void GRRLIB_ClearTex (GRRLIB_texImg *tex)`

*Clear a texture to transparent black.*
- `INLINE void GRRLIB_FlushTex (GRRLIB_texImg *tex)`

*Write the contents of a texture in the data cache down to main memory.*
- `INLINE void GRRLIB_FreeTexture (GRRLIB_texImg *tex)`

*Free memory allocated for texture.*
- `GRRLIB_bytemapFont * GRRLIB_LoadBMF (const u8 my_bmf[])`

*Load a ByteMap font structure from a buffer.*
- `void GRRLIB_FreeBMF (GRRLIB_bytemapFont *bmf)`

*Free memory allocated by ByteMap fonts.*
- `void GRRLIB_InitTileSet (GRRLIB_texImg *tex, const u32 tilew, const u32 tileh, const u32 tilestart)`

*Initialize a tile set.*
- `void GRRLIB_BMFX_FlipH (const GRRLIB_texImg *texsrc, GRRLIB_texImg *texdest)`

- Flip texture horizontal.*

  - void [GRRLIB\\_BMFX\\_FlipV](#) (const [GRRLIB\\_texImg](#) \*texsrc, [GRRLIB\\_texImg](#) \*texdest)
- Flip texture vertical.*

  - void [GRRLIB\\_BMFX\\_Grayscale](#) (const [GRRLIB\\_texImg](#) \*texsrc, [GRRLIB\\_texImg](#) \*texdest)
- Change a texture to gray scale.*

  - void [GRRLIB\\_BMFX\\_Sepia](#) (const [GRRLIB\\_texImg](#) \*texsrc, [GRRLIB\\_texImg](#) \*texdest)
- Change a texture to sepia (old photo style).*

  - void [GRRLIB\\_BMFX\\_Invert](#) (const [GRRLIB\\_texImg](#) \*texsrc, [GRRLIB\\_texImg](#) \*texdest)
- Invert colors of the texture.*

  - void [GRRLIB\\_BMFX\\_Blur](#) (const [GRRLIB\\_texImg](#) \*texsrc, [GRRLIB\\_texImg](#) \*texdest, const u32 factor)
- A texture effect (Blur).*

  - void [GRRLIB\\_BMFX\\_Scatter](#) (const [GRRLIB\\_texImg](#) \*texsrc, [GRRLIB\\_texImg](#) \*texdest, const u32 factor)
- A texture effect (Scatter).*

  - void [GRRLIB\\_BMFX\\_Pixelate](#) (const [GRRLIB\\_texImg](#) \*texsrc, [GRRLIB\\_texImg](#) \*texdest, const u32 factor)
- A texture effect (Pixelate).*

  - int [GRRLIB\\_Init](#) (void)
- Initialize GRRLIB.*

  - void [GRRLIB\\_Exit](#) (void)
- Call this before exiting your application.*

  - void [GRRLIB\\_Circle](#) (const f32 x, const f32 y, const f32 radius, const u32 color, const u8 filled)
- Draw a circle.*

  - int [GRRLIB\\_LoadFile](#) (const char \*filename, u8 \*\*data)
- Load a file to memory.*

  - [GRRLIB\\_texImg](#) \* [GRRLIB\\_LoadTextureFromFile](#) (const char \*filename)
- Load a texture from a file.*

  - bool [GRRLIB\\_ScrShot](#) (const char \*filename)
- Make a PNG screenshot.*

  - void [GRRLIB\\_Printf](#) (const f32 xpos, const f32 ypos, const [GRRLIB\\_texImg](#) \*tex, const u32 color, const f32 zoom, const char \*text,...)
- Print formatted output.*

  - void [GRRLIB\\_PrintBMF](#) (const f32 xpos, const f32 ypos, const [GRRLIB\\_bytemapFont](#) \*bmf, const char \*text,...)
- Print formatted output with a ByteMap font.*

  - void [GRRLIB\\_DrawImg](#) (const f32 xpos, const f32 ypos, const [GRRLIB\\_texImg](#) \*tex, const f32 degrees, const f32 scaleX, const f32 scaleY, const u32 color)
- Draw a texture.*

  - void [GRRLIB\\_DrawImgQuad](#) (const [guVector](#) pos[4], [GRRLIB\\_texImg](#) \*tex, const u32 color)
- Draw a textured quad.*

  - void [GRRLIB\\_DrawTile](#) (const f32 xpos, const f32 ypos, const [GRRLIB\\_texImg](#) \*tex, const f32 degrees, const f32 scaleX, const f32 scaleY, const u32 color, const int frame)
- Draw a tile.*

  - void [GRRLIB\\_DrawPart](#) (const f32 xpos, const f32 ypos, const f32 partx, const f32 party, const f32 partw, const f32 parth, const [GRRLIB\\_texImg](#) \*tex, const f32 degrees, const f32 scaleX, const f32 scaleY, const u32 color)
- Draw a part of a texture.*

  - void [GRRLIB\\_DrawTileQuad](#) (const [guVector](#) pos[4], [GRRLIB\\_texImg](#) \*tex, const u32 color, const int frame)
- Draw a tile in a quad.*

  - void [GRRLIB\\_Render](#) (void)
- Call this function after drawing.*

  - void [GRRLIB\\_Screen2Texture](#) (int posx, int posy, [GRRLIB\\_texImg](#) \*tex, bool clear)
- Make a snapshot of the screen in a texture WITHOUT ALPHA LAYER.*



- void `GRRLIB_CompStart` (void)  
*Start GX compositing process.*
- void `GRRLIB_CompEnd` (int posx, int posy, `GRRLIB_texImg *tex`)  
*End GX compositing process (Make a snapshot of the screen in a texture WITH ALPHA LAYER).*
- `GRRLIB_texImg * GRRLIB_CreateEmptyTexture` (const u32 w, const u32 h)  
*Create an empty texture.*
- `GRRLIB_texImg * GRRLIB_LoadTexture` (const u8 \*my\_img)  
*Load a texture from a buffer.*
- `GRRLIB_texImg * GRRLIB_LoadTexturePNG` (const u8 \*my\_png)  
*Load a texture from a buffer.*
- `GRRLIB_texImg * GRRLIB_LoadTextureJPG` (const u8 \*my\_jpg)  
*Load a texture from a buffer.*
- `GRRLIB_texImg * GRRLIB_LoadTextureJPGEx` (const u8 \*my\_jpg, const int)  
*Load a texture from a buffer.*
- `GRRLIB_texImg * GRRLIB_LoadTextureBMP` (const u8 \*my\_bmp)  
*Load a texture from a buffer.*
- bool `GRRLIB_GeckoInit` ()  
*Initialize USB Gecko.*
- void `GRRLIB_GeckoPrintf` (const char \*text,...)  
*Print Gecko.*
- void `GRRLIB_SetBackgroundColour` (u8 r, u8 g, u8 b, u8 a)  
*Set the background parameter when screen is cleared.*
- void `GRRLIB_Camera3dSettings` (f32 posx, f32 posy, f32 posz, f32 upx, f32 upy, f32 upz, f32 lookx, f32 looky, f32 lookz)  
*Set the camera parameter (contributed my chris\_c aka DaShAmAn).*
- void `GRRLIB_3dMode` (f32 minDist, f32 maxDist, f32 fov, bool texturemode, bool normalmode)  
*Set up the position matrix (contributed by chris\_c aka DaShAmAn).*
- void `GRRLIB_2dMode` ()  
*Go back to 2D mode (contributed by chris\_c aka DaShAmAn).*
- void `GRRLIB_ObjectViewBegin` (void)  
*Init the object matrix to draw object.*
- void `GRRLIB_ObjectViewScale` (f32 scalx, f32 scaly, f32 scalz)  
*Scale the object matrix to draw object.*
- void `GRRLIB_ObjectViewRotate` (f32 angx, f32 angy, f32 angz)  
*Rotate the object matrix to draw object .*
- void `GRRLIB_ObjectViewTrans` (f32 posx, f32 posy, f32 posz)  
*Translate the object matrix to draw object.*
- void `GRRLIB_ObjectViewEnd` (void)  
*Concat the object and the view matrix and calculate the inverse normal matrix.*
- void `GRRLIB_ObjectView` (f32 posx, f32 posy, f32 posz, f32 angx, f32 angy, f32 angz, f32 scalx, f32 scaly, f32 scalz)  
*Set the view matrix to draw object (in this order scale, rotate AND trans).*
- void `GRRLIB_ObjectViewInv` (f32 posx, f32 posy, f32 posz, f32 angx, f32 angy, f32 angz, f32 scalx, f32 scaly, f32 scalz)  
*Set the view matrix to draw object (in this order scale, trans AND rotate).*
- void `GRRLIB_SetTexture` (`GRRLIB_texImg *tex`, bool rep)  
*Set the texture to an object (contributed by chris\_c aka DaShAmAn).*
- void `GRRLIB_DrawTorus` (f32 r, f32 R, int nsides, int rings, bool filled, u32 col)  
*Draw a torus (with normal).*
- void `GRRLIB_DrawSphere` (f32 r, int lats, int longs, bool filled, u32 col)  
*Draw a sphere (with normal).*

- void [GRRLIB\\_DrawCube](#) (f32 size, bool filled, u32 col)  
*Draw a cube (with normal).*
- void [GRRLIB\\_DrawCylinder](#) (f32 r, f32 h, int d, bool filled, u32 col)  
*Draw a cylinder (with normal).*
- void [GRRLIB\\_DrawCone](#) (f32 r, f32 h, int d, bool filled, u32 col)  
*Draw a cone (with normal).*
- void [GRRLIB\\_DrawTessPanel](#) (f32 w, f32 wstep, f32 h, f32 hstep, bool filled, u32 col)  
*Draw a Tesselated panel (with normal).*
- void [GRRLIB\\_SetLightAmbient](#) (u32 ambientcolor)  
*Set ambient color.*
- void [GRRLIB\\_SetLightDiff](#) (u8 num, guVector pos, f32 distattn, f32 brightness, u32 lightcolor)  
*Set diffuse light parameters.*
- void [GRRLIB\\_SetLightSpec](#) (u8 num, guVector dir, f32 shy, u32 lightcolor, u32 speccolor)  
*Set specular light parameters.*
- void [GRRLIB\\_SetLightSpot](#) (u8 num, guVector pos, guVector lookat, f32 angAttn0, f32 angAttn1, f32 angAttn2, f32 distAttn0, f32 distAttn1, f32 distAttn2, u32 lightcolor)  
*Set Spot light parameters.*
- void [GRRLIB\\_SetLightOff](#) (void)  
*Set all lights off, like at init.*
- [GRRLIB\\_ttfFont](#) \* [GRRLIB\\_LoadTTF](#) (const u8 \*file\_base, s32 file\_size)  
*Load a TTF from a buffer.*
- void [GRRLIB\\_FreeTTF](#) ([GRRLIB\\_ttfFont](#) \*myFont)  
*Free memory allocated by TTF fonts.*
- void [GRRLIB\\_PrintTTF](#) (int x, int y, [GRRLIB\\_ttfFont](#) \*myFont, const char \*string, unsigned int fontSize, const u32 color)  
*Print function for TTF font.*
- void [GRRLIB\\_PrintTTFW](#) (int x, int y, [GRRLIB\\_ttfFont](#) \*myFont, const wchar\_t \*string, unsigned int fontSize, const u32 color)  
*Print function for TTF font.*
- u32 [GRRLIB\\_WidthTTF](#) ([GRRLIB\\_ttfFont](#) \*myFont, const char \*, unsigned int)  
*Get the width of a text in pixel.*
- u32 [GRRLIB\\_WidthTTFW](#) ([GRRLIB\\_ttfFont](#) \*myFont, const wchar\_t \*, unsigned int)  
*Get the width of a text in pixel.*

## Variables

- `GRR_EXTERN GXRModeObj` \* [rmode](#)  
*Video mode.*

### 7.1.1 Detailed Description

This is the complete list of functions, structures, defines, typedefs, enumerations and variables you may want to used to make your homebrew with GRRLIB. You simply need to include [grrlib.h](#) in your project to have access to all of these.

### 7.1.2 Macro Definition Documentation

### 7.1.2.1 RGBA

```
#define RGBA(
    r,
    g,
    b,
    a )
```

**Value:**

```
( (u32) ( ( (u32) (r)          <<24) | \
(((u32) (g)) &0xFF) <<16) | \
(((u32) (b)) &0xFF) << 8) | \
( (u32) (a) &0xFF          ) ) )
```

Build an RGB pixel from components.

**Parameters**

<i>r</i>	Red component.
<i>g</i>	Green component.
<i>b</i>	Blue component.
<i>a</i>	Alpha component.

## 7.1.3 Enumeration Type Documentation

### 7.1.3.1 GRRLIB\_blendMode

```
enum GRRLIB_blendMode
```

GRRLIB Blending Modes.

**Enumerator**

GRRLIB_BLEND_ALPHA	Alpha Blending.
GRRLIB_BLEND_ADD	Additive Blending.
GRRLIB_BLEND_SCREEN	Alpha Light Blending.
GRRLIB_BLEND_MULTI	Multiply Blending.
GRRLIB_BLEND_INV	Invert Color Blending.

## 7.1.4 Function Documentation

### 7.1.4.1 GRRLIB\_3dMode()

```
void GRRLIB_3dMode (
    f32 minDist,
```

```

    f32 maxDist,
    f32 fov,
    bool texturemode,
    bool normalmode )

```

Set up the position matrix (contributed by chris\_c aka DaShAmAn).

#### Parameters

<i>minDist</i>	Minimal distance for the camera.
<i>maxDist</i>	Maximal distance for the camera.
<i>fov</i>	Field of view for the camera.
<i>texturemode</i>	False, GX won't need texture coordinate, True, GX will need texture coordinate.
<i>normalmode</i>	False, GX won't need normal coordinate, True, GX will need normal coordinate.

#### 7.1.4.2 GRRLIB\_BMFX\_Blur()

```

void GRRLIB_BMFX_Blur (
    const GRRLIB_texImg * texsrc,
    GRRLIB_texImg * texdest,
    const u32 factor )

```

A texture effect (Blur).

#### See also

[GRRLIB\\_FlushTex](#)

#### Parameters

<i>texsrc</i>	The texture source.
<i>texdest</i>	The texture destination.
<i>factor</i>	The blur factor.

#### 7.1.4.3 GRRLIB\_BMFX\_FlipH()

```

void GRRLIB_BMFX_FlipH (
    const GRRLIB_texImg * texsrc,
    GRRLIB_texImg * texdest )

```

Flip texture horizontal.

#### See also

[GRRLIB\\_FlushTex](#)

## Parameters

<i>texsrc</i>	The texture source.
<i>texdest</i>	The texture destination.

**7.1.4.4 GRRLIB\_BMFX\_FlipV()**

```
void GRRLIB_BMFX_FlipV (
    const GRRLIB_texImg * texsrc,
    GRRLIB_texImg * texdest )
```

Flip texture vertical.

## See also

[GRRLIB\\_FlushTex](#)

## Parameters

<i>texsrc</i>	The texture source.
<i>texdest</i>	The texture destination.

**7.1.4.5 GRRLIB\_BMFX\_Grayscale()**

```
void GRRLIB_BMFX_Grayscale (
    const GRRLIB_texImg * texsrc,
    GRRLIB_texImg * texdest )
```

Change a texture to gray scale.

## See also

[GRRLIB\\_FlushTex](#)

## Parameters

<i>texsrc</i>	The texture source.
<i>texdest</i>	The texture grayscaled destination.

**7.1.4.6 GRRLIB\_BMFX\_Invert()**

```
void GRRLIB_BMFX_Invert (
```

```
const GRRLIB_texImg * texsrc,
GRRLIB_texImg * texdest )
```

Invert colors of the texture.

See also

[GRRLIB\\_FlushTex](#)

Parameters

<i>texsrc</i>	The texture source.
<i>texdest</i>	The texture destination.

#### 7.1.4.7 GRRLIB\_BMFX\_Pixelate()

```
void GRRLIB_BMFX_Pixelate (
    const GRRLIB_texImg * texsrc,
    GRRLIB_texImg * texdest,
    const u32 factor )
```

A texture effect (Pixelate).

See also

[GRRLIB\\_FlushTex](#)

Parameters

<i>texsrc</i>	The texture source.
<i>texdest</i>	The texture destination.
<i>factor</i>	The factor level of the effect.

#### 7.1.4.8 GRRLIB\_BMFX\_Scatter()

```
void GRRLIB_BMFX_Scatter (
    const GRRLIB_texImg * texsrc,
    GRRLIB_texImg * texdest,
    const u32 factor )
```

A texture effect (Scatter).

See also

[GRRLIB\\_FlushTex](#)

## Parameters

<i>texsrc</i>	The texture source.
<i>texdest</i>	The texture destination.
<i>factor</i>	The factor level of the effect.

**7.1.4.9 GRRLIB\_BMFX\_Sepia()**

```
void GRRLIB_BMFX_Sepia (
    const GRRLIB_texImg * texsrc,
    GRRLIB_texImg * texdest )
```

Change a texture to sepia (old photo style).

## See also

[GRRLIB\\_FlushTex](#)

## Parameters

<i>texsrc</i>	The texture source.
<i>texdest</i>	The texture destination.

## Author

elisherer

**7.1.4.10 GRRLIB\_Camera3dSettings()**

```
void GRRLIB_Camera3dSettings (
    f32 posx,
    f32 posy,
    f32 posz,
    f32 upx,
    f32 upy,
    f32 upz,
    f32 lookx,
    f32 looky,
    f32 lookz )
```

Set the camera parameter (contributed my chris\_c aka DaShAmAn).

## Parameters

<i>posx</i>	x position of the camera.
<i>posy</i>	y position of the camera.

## Parameters

<i>posz</i>	z position of the camera.
<i>upx</i>	Alpha component.
<i>upy</i>	Alpha component.
<i>upz</i>	Alpha component.
<i>lookx</i>	x up position of the camera.
<i>looky</i>	y up position of the camera.
<i>lookz</i>	z up position of the camera.

7.1.4.11 **GRRLIB\_Circle()**

```
void GRRLIB_Circle (
    const f32 x,
    const f32 y,
    const f32 radius,
    const u32 color,
    const u8 filled )
```

Draw a circle.

## Author

Dark\_Link

## Parameters

<i>x</i>	Specifies the x-coordinate of the circle.
<i>y</i>	Specifies the y-coordinate of the circle.
<i>radius</i>	The radius of the circle.
<i>color</i>	The color of the circle in RGBA format.
<i>filled</i>	Set to true to fill the circle.

7.1.4.12 **GRRLIB\_ClearTex()**

```
INLINE void GRRLIB_ClearTex (
    GRRLIB_texImg * tex )
```

Clear a texture to transparent black.

## Parameters

<i>tex</i>	Texture to clear.
------------	-------------------



**7.1.4.13 GRRLIB\_ClipDrawing()**

```

INLINE void GRRLIB_ClipDrawing (
    const int x,
    const int y,
    const int width,
    const int height )

```

Clip the drawing area to an rectangle.

**Parameters**

<i>x</i>	The x-coordinate of the rectangle.
<i>y</i>	The y-coordinate of the rectangle.
<i>width</i>	The width of the rectangle.
<i>height</i>	The height of the rectangle.

**7.1.4.14 GRRLIB\_CompEnd()**

```

void GRRLIB_CompEnd (
    int posx,
    int posy,
    GRRLIB_texImg * tex )

```

End GX compositing process (Make a snapshot of the screen in a texture WITH ALPHA LAYER).

EFB is cleared after this function.

**See also**

[GRRLIB\\_CompStart](#)

**Parameters**

<i>posx</i>	top left corner of the grabbed part.
<i>posy</i>	top left corner of the grabbed part.
<i>tex</i>	A pointer to a texture representing the screen or NULL if an error occurs.

**7.1.4.15 GRRLIB\_CompStart()**

```

void GRRLIB_CompStart (
    void )

```

Start GX compositing process.

See also

[GRRLIB\\_CompoEnd](#)

#### 7.1.4.16 GRRLIB\_CreateEmptyTexture()

```
GRRLIB_texImg* GRRLIB_CreateEmptyTexture (
    const u32 w,
    const u32 h )
```

Create an empty texture.

Parameters

<i>w</i>	Width of the new texture to create.
<i>h</i>	Height of the new texture to create.

Returns

A [GRRLIB\\_texImg](#) structure newly created.

#### 7.1.4.17 GRRLIB\_DrawCone()

```
void GRRLIB_DrawCone (
    f32 r,
    f32 h,
    int d,
    bool filled,
    u32 col )
```

Draw a cone (with normal).

Parameters

<i>r</i>	Radius of the cone.
<i>h</i>	High of the cone.
<i>d</i>	Density of slice.
<i>filled</i>	Wired or not.
<i>col</i>	Color of the cone.

#### 7.1.4.18 GRRLIB\_DrawCube()

```
void GRRLIB_DrawCube (
    f32 size,
```

```

    bool filled,
    u32 col )

```

Draw a cube (with normal).

#### Parameters

<i>size</i>	Size of the cube edge.
<i>filled</i>	Wired or not.
<i>col</i>	Color of the cube.

#### 7.1.4.19 GRRLIB\_DrawCylinder()

```

void GRRLIB_DrawCylinder (
    f32 r,
    f32 h,
    int d,
    bool filled,
    u32 col )

```

Draw a cylinder (with normal).

#### Parameters

<i>r</i>	Radius of the cylinder.
<i>h</i>	High of the cylinder.
<i>d</i>	Density of slice.
<i>filled</i>	Wired or not.
<i>col</i>	Color of the cylinder.

#### 7.1.4.20 GRRLIB\_DrawImg()

```

void GRRLIB_DrawImg (
    const f32 xpos,
    const f32 ypos,
    const GRRLIB_texImg * tex,
    const f32 degrees,
    const f32 scaleX,
    const f32 scaleY,
    const u32 color )

```

Draw a texture.

#### Parameters

<i>xpos</i>	Specifies the x-coordinate of the upper-left corner.
<i>ypos</i>	Specifies the y-coordinate of the upper-left corner.

## Parameters

<i>tex</i>	The texture to draw.
<i>degrees</i>	Angle of rotation.
<i>scaleX</i>	Specifies the x-coordinate scale. -1 could be used for flipping the texture horizontally.
<i>scaleY</i>	Specifies the y-coordinate scale. -1 could be used for flipping the texture vertically.
<i>color</i>	Color in RGBA format.

**7.1.4.21 GRRLIB\_DrawImgQuad()**

```
void GRRLIB_DrawImgQuad (
    const guVector pos[4],
    GRRLIB_texImg * tex,
    const u32 color )
```

Draw a textured quad.

## Parameters

<i>pos</i>	Vector array of the 4 points.
<i>tex</i>	The texture to draw.
<i>color</i>	Color in RGBA format.

**7.1.4.22 GRRLIB\_DrawPart()**

```
void GRRLIB_DrawPart (
    const f32 xpos,
    const f32 ypos,
    const f32 partx,
    const f32 party,
    const f32 partw,
    const f32 parth,
    const GRRLIB_texImg * tex,
    const f32 degrees,
    const f32 scaleX,
    const f32 scaleY,
    const u32 color )
```

Draw a part of a texture.

## Parameters

<i>xpos</i>	Specifies the x-coordinate of the upper-left corner.
<i>ypos</i>	Specifies the y-coordinate of the upper-left corner.
<i>partx</i>	Specifies the x-coordinate of the upper-left corner in the texture.
<i>party</i>	Specifies the y-coordinate of the upper-left corner in the texture.

## Parameters

<i>partw</i>	Specifies the width in the texture.
<i>parth</i>	Specifies the height in the texture.
<i>tex</i>	The texture containing the tile to draw.
<i>degrees</i>	Angle of rotation.
<i>scaleX</i>	Specifies the x-coordinate scale. -1 could be used for flipping the texture horizontally.
<i>scaleY</i>	Specifies the y-coordinate scale. -1 could be used for flipping the texture vertically.
<i>color</i>	Color in RGBA format.

## 7.1.4.23 GRRLIB\_DrawSphere()

```
void GRRLIB_DrawSphere (
    f32 r,
    int lats,
    int longs,
    bool filled,
    u32 col )
```

Draw a sphere (with normal).

## Parameters

<i>r</i>	Radius of the sphere.
<i>lats</i>	Number of latitudes.
<i>longs</i>	Number of longitudes.
<i>filled</i>	Wired or not.
<i>col</i>	Color of the sphere.

## 7.1.4.24 GRRLIB\_DrawTessPanel()

```
void GRRLIB_DrawTessPanel (
    f32 w,
    f32 wstep,
    f32 h,
    f32 hstep,
    bool filled,
    u32 col )
```

Draw a Tesselated panel (with normal).

## Parameters

<i>w</i>	Width of the panel.
<i>wstep</i>	Size of width slices.
<i>h</i>	Height of the panel.

## Parameters

<i>hstep</i>	Size the de height slices.
<i>filled</i>	Wired or not.
<i>col</i>	Color in RGBA format.

**7.1.4.25 GRRLIB\_DrawTile()**

```
void GRRLIB_DrawTile (
    const f32 xpos,
    const f32 ypos,
    const GRRLIB_texImg * tex,
    const f32 degrees,
    const f32 scaleX,
    const f32 scaleY,
    const u32 color,
    const int frame )
```

Draw a tile.

## Parameters

<i>xpos</i>	Specifies the x-coordinate of the upper-left corner.
<i>ypos</i>	Specifies the y-coordinate of the upper-left corner.
<i>tex</i>	The texture containing the tile to draw.
<i>degrees</i>	Angle of rotation.
<i>scaleX</i>	Specifies the x-coordinate scale. -1 could be used for flipping the texture horizontally.
<i>scaleY</i>	Specifies the y-coordinate scale. -1 could be used for flipping the texture vertically.
<i>color</i>	Color in RGBA format.
<i>frame</i>	Specifies the frame to draw.

**7.1.4.26 GRRLIB\_DrawTileQuad()**

```
void GRRLIB_DrawTileQuad (
    const guVector pos[4],
    GRRLIB_texImg * tex,
    const u32 color,
    const int frame )
```

Draw a tile in a quad.

## Parameters

<i>pos</i>	Vector array of the 4 points.
<i>tex</i>	The texture to draw.
<i>color</i>	Color in RGBA format.
<i>frame</i>	Specifies the frame to draw.

#### 7.1.4.27 GRRLIB\_DrawTorus()

```
void GRRLIB_DrawTorus (
    f32 r,
    f32 R,
    int nsides,
    int rings,
    bool filled,
    u32 col )
```

Draw a torus (with normal).

##### Parameters

<i>r</i>	Radius of the ring.
<i>R</i>	Radius of the torus.
<i>nsides</i>	Number of faces per ring.
<i>rings</i>	Number of rings.
<i>filled</i>	Wired or not.
<i>col</i>	Color of the torus.

#### 7.1.4.28 GRRLIB\_Exit()

```
void GRRLIB_Exit (
    void )
```

Call this before exiting your application.

Ensure this function is only ever called once and only if the setup function has been called.

#### 7.1.4.29 GRRLIB\_FillScreen()

```
INLINE void GRRLIB_FillScreen (
    const u32 color )
```

Clear screen with a specific color.

##### Parameters

<i>color</i>	The color to use to fill the screen.
--------------	--------------------------------------

#### 7.1.4.30 GRRLIB\_FlushTex()

```
INLINE void GRRLIB_FlushTex (
    GRRLIB_texImg * tex )
```

Write the contents of a texture in the data cache down to main memory.

For performance the CPU holds a data cache where modifications are stored before they get written down to main memory.

##### Parameters

<i>tex</i>	The texture to flush.
------------	-----------------------

#### 7.1.4.31 GRRLIB\_FreeBMF()

```
void GRRLIB_FreeBMF (
    GRRLIB_bytemapFont * bmf )
```

Free memory allocated by ByteMap fonts.

If *bmf* is a null pointer, the function does nothing.

##### Note

This function does not change the value of *bmf* itself, hence it still points to the same (now invalid) location.

##### Parameters

<i>bmf</i>	A <a href="#">GRRLIB_bytemapFont</a> structure.
------------	---

#### 7.1.4.32 GRRLIB\_FreeTexture()

```
INLINE void GRRLIB_FreeTexture (
    GRRLIB_texImg * tex )
```

Free memory allocated for texture.

If *tex* is a null pointer, the function does nothing.

##### Note

This function does not change the value of *tex* itself, hence it still points to the same (now invalid) location.



## Parameters

<i>tex</i>	A <a href="#">GRRLIB_texImg</a> structure.
------------	--

**7.1.4.33 GRRLIB\_FreeTTF()**

```
void GRRLIB_FreeTTF (
    GRRLIB_ttfFont * myFont )
```

Free memory allocated by TTF fonts.

If *myFont* is a null pointer, the function does nothing.

## Note

This function does not change the value of *myFont* itself, hence it still points to the same (now invalid) location.

## Parameters

<i>myFont</i>	A TTF.
---------------	--------

**7.1.4.34 GRRLIB\_GeckoInit()**

```
bool GRRLIB_GeckoInit ( )
```

Initialize USB Gecko.

## Returns

bool true=everything worked, false=problems occurred.

**7.1.4.35 GRRLIB\_GeckoPrintf()**

```
void GRRLIB_GeckoPrintf (
    const char * text,
    ... )
```

Print Gecko.

## Parameters

<i>text</i>	Text to print.
...	Optional arguments.

#### 7.1.4.36 GRRLIB\_GetAntiAliasing()

```
INLINE bool GRRLIB_GetAntiAliasing (
    void )
```

Get current anti-aliasing setting.

##### Returns

True if anti-aliasing is enabled.

#### 7.1.4.37 GRRLIB\_GetBlend()

```
INLINE GRRLIB_blendMode GRRLIB_GetBlend (
    void )
```

Get the current blending mode.

##### Returns

The current blending mode.

#### 7.1.4.38 GRRLIB\_GetPixelFromFB()

```
INLINE u32 GRRLIB_GetPixelFromFB (
    int x,
    int y )
```

Reads a pixel directly from the FrontBuffer.

##### Parameters

<i>x</i>	The x-coordinate within the FB.
<i>y</i>	The y-coordinate within the FB.

##### Returns

The color of a pixel in RGBA format.

**7.1.4.39 GRRLIB\_GetPixelFromtexImg()**

```

INLINE u32 GRRLIB_GetPixelFromtexImg (
    const int x,
    const int y,
    const GRRLIB_texImg * tex )

```

Return the color value of a pixel from a [GRRLIB\\_texImg](#).

**Parameters**

<i>x</i>	Specifies the x-coordinate of the pixel in the texture.
<i>y</i>	Specifies the y-coordinate of the pixel in the texture.
<i>tex</i>	The texture to get the color from.

**Returns**

The color of a pixel in RGBA format.

**7.1.4.40 GRRLIB\_GXEngine()**

```

INLINE void GRRLIB_GXEngine (
    const guVector v[],
    const u32 color[],
    const long n,
    const u8 fmt )

```

Draws a vector.

**Parameters**

<i>v</i>	The vector to draw.
<i>color</i>	The color of the vector in RGBA format.
<i>n</i>	Number of points in the vector.
<i>fmt</i>	Type of primitive.

**7.1.4.41 GRRLIB\_Init()**

```

int GRRLIB_Init (
    void )

```

Initialize GRRLIB.

Call this once at the beginning your code.

**Returns**

A integer representing a code:

- 0 : The operation completed successfully.
- -1 : Not enough memory is available to initialize GRRLIB.
- -2 : Failed to add the fat device driver to the devoptab.
- -3 : Failed to initialize the font engine.

**See also**

[GRRLIB\\_Exit](#)

**7.1.4.42 GRRLIB\_InitTileSet()**

```
void GRRLIB_InitTileSet (
    GRRLIB_texImg * tex,
    const u32 tilew,
    const u32 tileh,
    const u32 tilestart )
```

Initialize a tile set.

**Parameters**

<i>tex</i>	The texture to initialize.
<i>tilew</i>	Width of the tile.
<i>tileh</i>	Height of the tile.
<i>tilestart</i>	Offset for starting position (Used in fonts).

**7.1.4.43 GRRLIB\_Line()**

```
INLINE void GRRLIB_Line (
    const f32 x1,
    const f32 y1,
    const f32 x2,
    const f32 y2,
    const u32 color )
```

Draw a line.

**Parameters**

<i>x1</i>	Starting point for line for the x coordinate.
<i>y1</i>	Starting point for line for the y coordinate.
<i>x2</i>	Ending point for line for the x coordinate.
<i>y2</i>	Ending point for line for the x coordinate.
<i>color</i>	Line color in RGBA format.

Author

JESPA

#### 7.1.4.44 GRRLIB\_LoadBMF()

```
GRRLIB_bytemapFont* GRRLIB_LoadBMF (
    const u8 my_bmf [ ] )
```

Load a ByteMap font structure from a buffer.

File format version 1.1 is used, more information could be found at <https://bmf.php5.cz/?page=format>

Parameters

<i>my_bmf</i>	The ByteMap font buffer to load.
---------------	----------------------------------

Returns

A [GRRLIB\\_bytemapFont](#) structure filled with BMF information.

See also

[GRRLIB\\_FreeBMF](#)

#### 7.1.4.45 GRRLIB\_LoadFile()

```
int GRRLIB_LoadFile (
    const char * filename,
    u8 ** data )
```

Load a file to memory.

Parameters

<i>filename</i>	Name of the file to be loaded.
<i>data</i>	Pointer-to-your-pointer. Ie. { u8 *data; GRRLIB_LoadFile("file", &data); }. It is your responsibility to free the memory allocated by this function.

Returns

A integer representing a code:

- 0 : EmptyFile.
- -1 : FileNotFound.

- -2 : OutOfMemory.
- -3 : FileReadError.
- >0 : FileLength.

#### 7.1.4.46 GRRLIB\_LoadTexture()

```
GRRLIB_texImg* GRRLIB_LoadTexture (
    const u8 * my_img )
```

Load a texture from a buffer.

##### Parameters

<i>my_img</i>	The JPEG, PNG or Bitmap buffer to load.
---------------	---

##### Returns

A [GRRLIB\\_texImg](#) structure filled with image information.

#### 7.1.4.47 GRRLIB\_LoadTextureBMP()

```
GRRLIB_texImg* GRRLIB_LoadTextureBMP (
    const u8 * my_bmp )
```

Load a texture from a buffer.

It only works for the MS-Windows standard format uncompressed (1-bit, 4-bit, 8-bit, 24-bit and 32-bit).

##### Parameters

<i>my_bmp</i>	the Bitmap buffer to load.
---------------	----------------------------

##### Returns

A [GRRLIB\\_texImg](#) structure filled with image information.

#### 7.1.4.48 GRRLIB\_LoadTextureFromFile()

```
GRRLIB_texImg* GRRLIB_LoadTextureFromFile (
    const char * filename )
```

Load a texture from a file.

## Parameters

<i>filename</i>	The JPEG, PNG or Bitmap filename to load.
-----------------	---

## Returns

A [GRRLIB\\_texImg](#) structure filled with image information. If an error occurs NULL will be returned.

#### 7.1.4.49 GRRLIB\_LoadTextureJPG()

```
GRRLIB_texImg* GRRLIB_LoadTextureJPG (
    const u8 * my_jpg )
```

Load a texture from a buffer.

Take care to have the JPG finish with 0xFF 0xD9!

## Parameters

<i>my_jpg</i>	The JPEG buffer to load.
---------------	--------------------------

## Returns

A [GRRLIB\\_texImg](#) structure filled with image information.

#### 7.1.4.50 GRRLIB\_LoadTextureJPGEx()

```
GRRLIB_texImg* GRRLIB_LoadTextureJPGEx (
    const u8 * my_jpg,
    const int my_size )
```

Load a texture from a buffer.

## Author

DrTwox

## Parameters

<i>my_jpg</i>	The JPEG buffer to load.
<i>my_size</i>	Size of the JPEG buffer to load.

**Returns**

A [GRRLIB\\_texImg](#) structure filled with image information.

**7.1.4.51 GRRLIB\_LoadTexturePNG()**

```
GRRLIB_texImg* GRRLIB_LoadTexturePNG (
    const u8 * my_png )
```

Load a texture from a buffer.

**Parameters**

<i>my_png</i>	the PNG buffer to load.
---------------	-------------------------

**Returns**

A [GRRLIB\\_texImg](#) structure filled with image information. If image size is not correct, the texture will be completely transparent.

**7.1.4.52 GRRLIB\_LoadTTF()**

```
GRRLIB_ttfFont* GRRLIB_LoadTTF (
    const u8 * file_base,
    s32 file_size )
```

Load a TTF from a buffer.

**Parameters**

<i>file_base</i>	Buffer with TTF data. You must not deallocate the memory before calling <a href="#">GRRLIB_FreeTTF</a> .
<i>file_size</i>	Size of the TTF buffer.

**Returns**

A handle to a given TTF font object or NULL if it fails to load the font.

**See also**

[GRRLIB\\_FreeTTF](#)



**7.1.4.53 GRRLIB\_NGone()**

```

INLINE void GRRLIB_NGone (
    const guVector v[],
    const u32 color[],
    const long n )

```

Draw a polygon.

**Parameters**

<i>v</i>	The vector containing the coordinates of the polygon.
<i>color</i>	The color of the filled polygon in RGBA format.
<i>n</i>	Number of points in the vector.

**7.1.4.54 GRRLIB\_NGoneFilled()**

```

INLINE void GRRLIB_NGoneFilled (
    const guVector v[],
    const u32 color[],
    const long n )

```

Draw a filled polygon.

**Parameters**

<i>v</i>	The vector containing the coordinates of the polygon.
<i>color</i>	The color of the filled polygon in RGBA format.
<i>n</i>	Number of points in the vector.

**7.1.4.55 GRRLIB\_NPlot()**

```

INLINE void GRRLIB_NPlot (
    const guVector v[],
    const u32 color[],
    const long n )

```

Draw an array of points.

**Parameters**

<i>v</i>	Array containing the points.
<i>color</i>	The color of the points in RGBA format.
<i>n</i>	Number of points in the vector array.

#### 7.1.4.56 GRRLIB\_ObjectView()

```
void GRRLIB_ObjectView (
    f32 posx,
    f32 posy,
    f32 posz,
    f32 angx,
    f32 angy,
    f32 angz,
    f32 scalx,
    f32 scaly,
    f32 scalz )
```

Set the view matrix to draw object (in this order scale, rotate AND trans).

##### Parameters

<i>posx</i>	x position of the object.
<i>posy</i>	y position of the object.
<i>posz</i>	z position of the object.
<i>angx</i>	x rotation angle of the object.
<i>angy</i>	y rotation angle of the object.
<i>angz</i>	z rotation angle of the object.
<i>scalx</i>	x scale of the object.
<i>scaly</i>	y scale of the object.
<i>scalz</i>	z scale of the object.

#### 7.1.4.57 GRRLIB\_ObjectViewInv()

```
void GRRLIB_ObjectViewInv (
    f32 posx,
    f32 posy,
    f32 posz,
    f32 angx,
    f32 angy,
    f32 angz,
    f32 scalx,
    f32 scaly,
    f32 scalz )
```

Set the view matrix to draw object (in this order scale, trans AND rotate).

##### Parameters

<i>posx</i>	x position of the object.
<i>posy</i>	y position of the object.
<i>posz</i>	z position of the object.

## Parameters

<i>angx</i>	x rotation angle of the object.
<i>angy</i>	y rotation angle of the object.
<i>angz</i>	z rotation angle of the object.
<i>scalx</i>	x scale of the object.
<i>scaly</i>	y scale of the object.
<i>scalz</i>	z scale of the object.

**7.1.4.58 GRRLIB\_ObjectViewRotate()**

```
void GRRLIB_ObjectViewRotate (
    f32 angx,
    f32 angy,
    f32 angz )
```

Rotate the object matrix to draw object .

## Parameters

<i>angx</i>	x rotation angle of the object.
<i>angy</i>	y rotation angle of the object.
<i>angz</i>	z rotation angle of the object.

**7.1.4.59 GRRLIB\_ObjectViewScale()**

```
void GRRLIB_ObjectViewScale (
    f32 scalx,
    f32 scaly,
    f32 scalz )
```

Scale the object matrix to draw object.

## Parameters

<i>scalx</i>	x scale of the object.
<i>scaly</i>	y scale of the object.
<i>scalz</i>	z scale of the object.

**7.1.4.60 GRRLIB\_ObjectViewTrans()**

```
void GRRLIB_ObjectViewTrans (
```

```

    f32 posx,
    f32 posy,
    f32 posz )

```

Translate the object matrix to draw object.

#### Parameters

<i>posx</i>	x position of the object.
<i>posy</i>	y position of the object.
<i>posz</i>	z position of the object.

#### 7.1.4.61 GRRLIB\_Plot()

```

INLINE void GRRLIB_Plot (
    const f32 x,
    const f32 y,
    const u32 color )

```

Draw a dot.

#### Parameters

<i>x</i>	Specifies the x-coordinate of the dot.
<i>y</i>	Specifies the y-coordinate of the dot.
<i>color</i>	The color of the dot in RGBA format.

#### Author

Jespa

#### 7.1.4.62 GRRLIB\_PrintBMF()

```

void GRRLIB_PrintBMF (
    const f32 xpos,
    const f32 ypos,
    const GRRLIB_bytemapFont * bmf,
    const char * text,
    ... )

```

Print formatted output with a ByteMap font.

This function could be slow, it should be used with GRRLIB\_CompoStart and GRRLIB\_CompoEnd.

#### Parameters

<i>xpos</i>	Specifies the x-coordinate of the upper-left corner of the text.
<i>ypos</i>	Specifies the y-coordinate of the upper-left corner of the text.
<i>bmf</i>	The ByteMap font to use.
<i>text</i>	Text to draw.
...	Optional arguments.

### 7.1.4.63 GRRLIB\_Printf()

```
void GRRLIB_Printf (
    const f32 xpos,
    const f32 ypos,
    const GRRLIB_texImg * tex,
    const u32 color,
    const f32 zoom,
    const char * text,
    ... )
```

Print formatted output.

#### Parameters

<i>xpos</i>	Specifies the x-coordinate of the upper-left corner of the text.
<i>ypos</i>	Specifies the y-coordinate of the upper-left corner of the text.
<i>tex</i>	The texture containing the character set.
<i>color</i>	Text color in RGBA format. The alpha channel is used to change the opacity of the text.
<i>zoom</i>	This is a factor by which the text size will be increase or decrease.
<i>text</i>	Text to draw.
...	Optional arguments.

### 7.1.4.64 GRRLIB\_PrintfTTF()

```
void GRRLIB_PrintfTTF (
    int x,
    int y,
    GRRLIB_ttfFont * myFont,
    const char * string,
    unsigned int fontSize,
    const u32 color )
```

Print function for TTF font.

#### Parameters

<i>x</i>	Specifies the x-coordinate of the upper-left corner of the text.
<i>y</i>	Specifies the y-coordinate of the upper-left corner of the text.
<i>myFont</i>	A TTF.
<i>string</i>	Text to draw.
<i>fontSize</i>	Size of the font.
<i>color</i>	Text color in RGBA format.

#### 7.1.4.65 GRRLIB\_PrintfTTFW()

```
void GRRLIB_PrintfTTFW (
    int x,
    int y,
    GRRLIB_ttfFont * myFont,
    const wchar_t * utf32,
    unsigned int fontSize,
    const u32 color )
```

Print function for TTF font.

#### Author

wplaat and DrTwox

#### Parameters

<i>x</i>	Specifies the x-coordinate of the upper-left corner of the text.
<i>y</i>	Specifies the y-coordinate of the upper-left corner of the text.
<i>myFont</i>	A TTF.
<i>utf32</i>	Text to draw.
<i>fontSize</i>	Size of the font.
<i>color</i>	Text color in RGBA format.

#### 7.1.4.66 GRRLIB\_PtInRect()

```
INLINE bool GRRLIB_PtInRect (
    const int hotx,
    const int hoty,
    const int hotw,
    const int hoth,
    const int wpadx,
    const int wpady )
```

Determine whether the specified point lies within the specified rectangle.

#### Parameters

<i>hotx</i>	Specifies the x-coordinate of the upper-left corner of the rectangle.
<i>hoty</i>	Specifies the y-coordinate of the upper-left corner of the rectangle.
<i>hotw</i>	The width of the rectangle.
<i>hoth</i>	The height of the rectangle.
<i>wpadx</i>	Specifies the x-coordinate of the point.
<i>wpady</i>	Specifies the y-coordinate of the point.

**Returns**

If the specified point lies within the rectangle, the return value is true otherwise it's false.

**7.1.4.67 GRRLIB\_Rectangle()**

```

INLINE void GRRLIB_Rectangle (
    const f32 x,
    const f32 y,
    const f32 width,
    const f32 height,
    const u32 color,
    const bool filled )

```

Draw a rectangle.

**Parameters**

<i>x</i>	Specifies the x-coordinate of the upper-left corner of the rectangle.
<i>y</i>	Specifies the y-coordinate of the upper-left corner of the rectangle.
<i>width</i>	The width of the rectangle.
<i>height</i>	The height of the rectangle.
<i>color</i>	The color of the rectangle in RGBA format.
<i>filled</i>	Set to true to fill the rectangle.

**7.1.4.68 GRRLIB\_RectInRect()**

```

INLINE bool GRRLIB_RectInRect (
    const int rect1x,
    const int rect1y,
    const int rect1w,
    const int rect1h,
    const int rect2x,
    const int rect2y,
    const int rect2w,
    const int rect2h )

```

Determine whether a specified rectangle lies within another rectangle.

**Parameters**

<i>rect1x</i>	Specifies the x-coordinate of the upper-left corner of the rectangle.
<i>rect1y</i>	Specifies the y-coordinate of the upper-left corner of the rectangle.
<i>rect1w</i>	Specifies the width of the rectangle.
<i>rect1h</i>	Specifies the height of the rectangle.
<i>rect2x</i>	Specifies the x-coordinate of the upper-left corner of the rectangle.
<i>rect2y</i>	Specifies the y-coordinate of the upper-left corner of the rectangle.
<i>rect2w</i>	Specifies the width of the rectangle.
<i>rect2h</i>	Specifies the height of the rectangle.

**Returns**

If the specified rectangle lies within the other rectangle, the return value is true otherwise it's false.

**7.1.4.69 GRRLIB\_RectOnRect()**

```

INLINE bool GRRLIB_RectOnRect (
    const int rect1x,
    const int rect1y,
    const int rect1w,
    const int rect1h,
    const int rect2x,
    const int rect2y,
    const int rect2w,
    const int rect2h )

```

Determine whether a part of a specified rectangle lies on another rectangle.

**Parameters**

<i>rect1x</i>	Specifies the x-coordinate of the upper-left corner of the first rectangle.
<i>rect1y</i>	Specifies the y-coordinate of the upper-left corner of the first rectangle.
<i>rect1w</i>	Specifies the width of the first rectangle.
<i>rect1h</i>	Specifies the height of the first rectangle.
<i>rect2x</i>	Specifies the x-coordinate of the upper-left corner of the second rectangle.
<i>rect2y</i>	Specifies the y-coordinate of the upper-left corner of the second rectangle.
<i>rect2w</i>	Specifies the width of the second rectangle.
<i>rect2h</i>	Specifies the height of the second rectangle.

**Returns**

If the specified rectangle lies on the other rectangle, the return value is true otherwise it's false.

**7.1.4.70 GRRLIB\_Screen2Texture()**

```

void GRRLIB_Screen2Texture (
    int posx,
    int posy,
    GRRLIB_texImg * tex,
    bool clear )

```

Make a snapshot of the screen in a texture WITHOUT ALPHA LAYER.

**Parameters**

<i>posx</i>	top left corner of the grabbed part.
<i>posy</i>	top left corner of the grabbed part.
<i>tex</i>	A pointer to a texture representing the screen or NULL if an error occurs.
<i>clear</i>	When this flag is set to true, the screen is cleared after copy.



**7.1.4.71 GRRLIB\_ScrShot()**

```
bool GRRLIB_ScrShot (
    const char * filename )
```

Make a PNG screenshot.

It should be called after drawing stuff on the screen, but before GRRLIB\_Render. libfat is required to use the function.

**Parameters**

<i>filename</i>	Name of the file to write.
-----------------	----------------------------

**Returns**

bool true=everything worked, false=problems occurred.

**7.1.4.72 GRRLIB\_SetAntiAliasing()**

```
INLINE void GRRLIB_SetAntiAliasing (
    const bool aa )
```

Turn anti-aliasing on/off.

**Parameters**

<i>aa</i>	Set to true to enable anti-aliasing (Default: Enabled).
-----------	---

**7.1.4.73 GRRLIB\_SetBackgroundColour()**

```
void GRRLIB_SetBackgroundColour (
    u8 r,
    u8 g,
    u8 b,
    u8 a )
```

Set the background parameter when screen is cleared.

**Parameters**

<i>r</i>	Red component.
<i>g</i>	Green component.
<i>b</i>	Blue component.
<i>a</i>	Alpha component.

#### 7.1.4.74 GRRLIB\_SetBlend()

```

INLINE void GRRLIB_SetBlend (
    const GRRLIB_blendMode blendmode )

```

Set a blending mode.

##### Parameters

<i>blendmode</i>	The blending mode to use (Default: GRRLIB_BLEND_ALPHA).
------------------	---

#### 7.1.4.75 GRRLIB\_SetHandle()

```

INLINE void GRRLIB_SetHandle (
    GRRLIB_texImg * tex,
    const int x,
    const int y )

```

Set a texture's X and Y handles.

For example, it could be used for the rotation of a texture.

##### Parameters

<i>tex</i>	The texture to set the handle on.
<i>x</i>	The x-coordinate of the handle.
<i>y</i>	The y-coordinate of the handle.

#### 7.1.4.76 GRRLIB\_SetLightAmbient()

```

void GRRLIB_SetLightAmbient (
    u32 ambientcolor )

```

Set ambient color.

When no diffuse light is shining on a object, the color is equal to ambient color.

##### Parameters

<i>ambientcolor</i>	Ambient color in RGBA format.
---------------------	-------------------------------

**7.1.4.77 GRRLIB\_SetLightDiff()**

```
void GRRLIB_SetLightDiff (
    u8 num,
    guVector pos,
    f32 distattn,
    f32 brightness,
    u32 lightcolor )
```

Set diffuse light parameters.

**Parameters**

<i>num</i>	Number of the light. It's a number from 0 to 7.
<i>pos</i>	Position of the diffuse light (x/y/z).
<i>distattn</i>	Distance attenuation.
<i>brightness</i>	Brightness of the light. The value should be between 0 and 1.
<i>lightcolor</i>	Color of the light in RGBA format.

**7.1.4.78 GRRLIB\_SetLightSpec()**

```
void GRRLIB_SetLightSpec (
    u8 num,
    guVector dir,
    f32 shy,
    u32 lightcolor,
    u32 speccolor )
```

Set specular light parameters.

**Parameters**

<i>num</i>	Number of the light. It's a number from 0 to 7.
<i>dir</i>	Direction of the specular ray (x/y/z).
<i>shy</i>	Shyniness of the specular. ( between 4 and 254)
<i>lightcolor</i>	Color of the light in RGBA format.
<i>speccolor</i>	Specular color in RGBA format..

**7.1.4.79 GRRLIB\_SetLightSpot()**

```
void GRRLIB_SetLightSpot (
    u8 num,
    guVector pos,
    guVector lookat,
    f32 angAttn0,
```

```

    f32 angAttn1,
    f32 angAttn2,
    f32 distAttn0,
    f32 distAttn1,
    f32 distAttn2,
    u32 lightcolor )

```

Set Spot light parameters.

#### Parameters

<i>num</i>	Number of the light. It's a number from 0 to 7.
<i>pos</i>	Position of the spot light (x/y/z).
<i>lookat</i>	Where spot light look at (x/y/z).
<i>angAttn0</i>	cone attenuation factor 0.
<i>angAttn1</i>	cone attenuation factor 1.
<i>angAttn2</i>	cone attenuation factor 2.
<i>distAttn0</i>	Distance attenuation factor 0.
<i>distAttn1</i>	Distance attenuation factor 1.
<i>distAttn2</i>	Distance attenuation factor 2.
<i>lightcolor</i>	Color of the light in RGBA format.

#### 7.1.4.80 GRRLIB\_SetMidHandle()

```

inline void GRRLIB_SetMidHandle (
    GRRLIB_texImg * tex,
    const bool enabled )

```

Center a texture's handles.

For example, it could be used for the rotation of a texture.

#### Parameters

<i>tex</i>	The texture to center.
<i>enabled</i>	

#### 7.1.4.81 GRRLIB\_SetPixelToFB()

```

inline void GRRLIB_SetPixelToFB (
    int x,
    int y,
    u32 pokeColor )

```

Writes a pixel directly from the FrontBuffer.

## Parameters

<i>x</i>	The x-coordinate within the FB.
<i>y</i>	The y-coordinate within the FB.
<i>pokeColor</i>	The color of the pixel in RGBA format.

## 7.1.4.82 GRRLIB\_SetPixelTotexImg()

```

INLINE void GRRLIB_SetPixelTotexImg (
    const int x,
    const int y,
    GRRLIB_texImg * tex,
    const u32 color )

```

Set the color value of a pixel to a [GRRLIB\\_texImg](#).

## See also

[GRRLIB\\_FlushTex](#)

## Parameters

<i>x</i>	Specifies the x-coordinate of the pixel in the texture.
<i>y</i>	Specifies the y-coordinate of the pixel in the texture.
<i>tex</i>	The texture to set the color to.
<i>color</i>	The color of the pixel in RGBA format.

## 7.1.4.83 GRRLIB\_SetTexture()

```

void GRRLIB_SetTexture (
    GRRLIB_texImg * tex,
    bool rep )

```

Set the texture to an object (contributed by [chris\\_c](#) aka DaShAmAn).

## Parameters

<i>tex</i>	Pointer to an image texture ( <a href="#">GRRLIB_texImg</a> format).
<i>rep</i>	Texture Repeat Mode, True will repeat it, False won't.

#### 7.1.4.84 GRRLIB\_WidthTTF()

```
u32 GRRLIB_WidthTTF (
    GRRLIB_ttfFont * myFont,
    const char * string,
    unsigned int fontSize )
```

Get the width of a text in pixel.

##### Parameters

<i>myFont</i>	A TTF.
<i>string</i>	The text to check.
<i>fontSize</i>	The size of the font.

##### Returns

The width of a text in pixel.

#### 7.1.4.85 GRRLIB\_WidthTTFW()

```
u32 GRRLIB_WidthTTFW (
    GRRLIB_ttfFont * myFont,
    const wchar_t * utf32,
    unsigned int fontSize )
```

Get the width of a text in pixel.

##### Parameters

<i>myFont</i>	A TTF.
<i>utf32</i>	The text to check.
<i>fontSize</i>	The size of the font.

##### Returns

The width of a text in pixel.

## Chapter 8

# Data Structure Documentation

### 8.1 GRRLIB\_bytemapChar Struct Reference

Structure to hold the bytemap character information.

```
#include <grplib.h>
```

#### Data Fields

- u8 [width](#)  
*Character width.*
- u8 [height](#)  
*Character height.*
- s8 [relx](#)  
*Horizontal offset relative to cursor (-128 to 127).*
- s8 [rely](#)  
*Vertical offset relative to cursor (-128 to 127).*
- u8 [kerning](#)  
*Kerning (Horizontal cursor shift after drawing the character).*
- u8 \* [data](#)  
*Character data (uncompressed, 8 bits per pixel).*

#### 8.1.1 Detailed Description

Structure to hold the bytemap character information.

#### 8.1.2 Field Documentation

### 8.1.2.1 data

u8\* data

Character data (uncompressed, 8 bits per pixel).

### 8.1.2.2 height

u8 height

Character height.

### 8.1.2.3 relx

s8 relx

Horizontal offset relative to cursor (-128 to 127).

### 8.1.2.4 rely

s8 rely

Vertical offset relative to cursor (-128 to 127).

### 8.1.2.5 width

u8 width

Character width.

The documentation for this struct was generated from the following file:

- [grrlib.h](#)

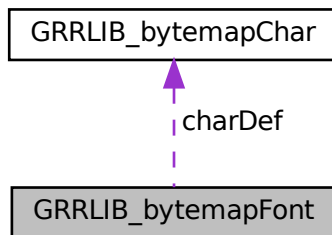


## 8.2 GRRLIB\_bytemapFont Struct Reference

Structure to hold the bytemap font information.

```
#include <grrlib.h>
```

Collaboration diagram for GRRLIB\_bytemapFont:



### Data Fields

- `char * name`  
*Font name.*
- `u32 * palette`  
*Font palette.*
- `u16 nbChar`  
*Number of characters in font.*
- `u8 version`  
*Version.*
- `s8 tracking`  
*Tracking (Add-space after each char) (-128 to 127).*
- `GRRLIB_bytemapChar charDef [256]`  
*Array of bitmap characters.*

### 8.2.1 Detailed Description

Structure to hold the bytemap font information.

### 8.2.2 Field Documentation

### 8.2.2.1 name

`char* name`

Font name.

### 8.2.2.2 nbChar

`u16 nbChar`

Number of characters in font.

### 8.2.2.3 palette

`u32* palette`

Font palette.

### 8.2.2.4 version

`u8 version`

Version.

The documentation for this struct was generated from the following file:

- [grrlib.h](#)

## 8.3 GRRLIB\_drawSettings Struct Reference

Structure to hold the current drawing settings.

```
#include <grrlib.h>
```

## Data Fields

- bool [antialias](#)  
*AntiAlias is enabled when set to true.*
- [GRRLIB\\_blendMode](#) [blend](#)  
*Blending Mode.*
- int [lights](#)  
*Active lights.*

### 8.3.1 Detailed Description

Structure to hold the current drawing settings.

### 8.3.2 Field Documentation

#### 8.3.2.1 [blend](#)

[GRRLIB\\_blendMode](#) [blend](#)

Blending Mode.

#### 8.3.2.2 [lights](#)

`int lights`

Active lights.

The documentation for this struct was generated from the following file:

- [grrlib.h](#)

## 8.4 GRRLIB\_Font Struct Reference

Structure to hold the TTF information.

```
#include <grrlib.h>
```

## Data Fields

- void \* [face](#)  
*A TTF face object.*
- bool [kerning](#)  
*true whenever a face object contains kerning data that can be accessed with FT\_Get\_Kerning.*

### 8.4.1 Detailed Description

Structure to hold the TTF information.

The documentation for this struct was generated from the following file:

- [grrlib.h](#)

## 8.5 GRRLIB\_texImg Struct Reference

Structure to hold the texture information.

```
#include <grrlib.h>
```

## Data Fields

- u32 [w](#)  
*The width of the texture in pixels.*
- u32 [h](#)  
*The height of the texture in pixels.*
- int [handlex](#)  
*Texture handle x.*
- int [handley](#)  
*Texture handle y.*
- int [offsetx](#)  
*Texture offset x.*
- int [offsety](#)  
*Texture offset y.*
- bool [tiledtex](#)  
*Texture is tiled if set to true.*
- u32 [tilew](#)  
*The width of one tile in pixels.*
- u32 [tileh](#)  
*The height of one tile in pixels.*
- u32 [nbtilew](#)  
*Number of tiles for the x axis.*
- u32 [nbtileh](#)  
*Number of tiles for the y axis.*
- u32 [tilestart](#)  
*Offset to tile starting position.*
- f32 [ofnormaltexx](#)  
*Offset of normalized texture on x.*
- f32 [ofnormaltexy](#)  
*Offset of normalized texture on y.*
- void \* [data](#)  
*Pointer to the texture data.*

## 8.5.1 Detailed Description

Structure to hold the texture information.

## 8.5.2 Field Documentation

### 8.5.2.1 nbtileh

```
u32 nbtileh
```

Number of tiles for the y axis.

### 8.5.2.2 nbtilew

```
u32 nbtilew
```

Number of tiles for the x axis.

### 8.5.2.3 tiledtex

```
bool tiledtex
```

Texture is tiled if set to true.

### 8.5.2.4 tileh

```
u32 tileh
```

The height of one tile in pixels.

#### 8.5.2.5 tilestart

u32 tilestart

Offset to tile starting position.

#### 8.5.2.6 tilew

u32 tilew

The width of one tile in pixels.

#### 8.5.2.7 w

u32 w

The width of the texture in pixels.

The documentation for this struct was generated from the following file:

- [grrlib.h](#)

# Chapter 9

## File Documentation

### 9.1 grrlib.h File Reference

```
#include <gccore.h>
#include "grrlib/GRRLIB__lib.h"
#include "grrlib/GRRLIB__inline.h"
```

#### Data Structures

- struct [GRRLIB\\_drawSettings](#)  
*Structure to hold the current drawing settings.*
- struct [GRRLIB\\_texImg](#)  
*Structure to hold the texture information.*
- struct [GRRLIB\\_bytemapChar](#)  
*Structure to hold the bytemap character information.*
- struct [GRRLIB\\_bytemapFont](#)  
*Structure to hold the bytemap font information.*
- struct [GRRLIB\\_Font](#)  
*Structure to hold the TTF information.*

#### Macros

- #define [GRRLIB\\_VER\\_STRING](#) "4.4.1"  
*Version information for GRRLIB.*
- #define [R\(c\)](#) (((c) >> 24) & 0xFF)  
*Extract red component of colour.*
- #define [G\(c\)](#) (((c) >> 16) & 0xFF)  
*Extract green component of colour.*
- #define [B\(c\)](#) (((c) >> 8) & 0xFF)  
*Extract blue component of colour.*
- #define [A\(c\)](#) ( (c) & 0xFF)  
*Extract alpha component of colour.*
- #define [RGBA](#)(r, g, b, a)

*Build an RGB pixel from components.*

- #define [GRRLIB\\_BLEND\\_NONE](#) ([GRRLIB\\_BLEND\\_ALPHA](#))  
*Alias for GRRLIB\_BLEND\_ALPHA.*
- #define [GRRLIB\\_BLEND\\_LIGHT](#) ([GRRLIB\\_BLEND\\_ADD](#))  
*Alias for GRRLIB\_BLEND\_ADD.*
- #define [GRRLIB\\_BLEND\\_SHADE](#) ([GRRLIB\\_BLEND\\_MULTI](#))  
*Alias for GRRLIB\_BLEND\_MULTI.*
- #define **GRR\_EXTERN** extern
- #define **GRR\_INIT**(v)
- #define **GRR\_INITS**(...)
- #define **INLINE** static inline

## Typedefs

- typedef enum [GRRLIB\\_blendMode](#) [GRRLIB\\_blendMode](#)  
*GRRLIB Blending Modes.*
- typedef struct [GRRLIB\\_drawSettings](#) [GRRLIB\\_drawSettings](#)  
*Structure to hold the current drawing settings.*
- typedef struct [GRRLIB\\_texImg](#) [GRRLIB\\_texImg](#)  
*Structure to hold the texture information.*
- typedef struct [GRRLIB\\_bytemapChar](#) [GRRLIB\\_bytemapChar](#)  
*Structure to hold the bytemap character information.*
- typedef struct [GRRLIB\\_bytemapFont](#) [GRRLIB\\_bytemapFont](#)  
*Structure to hold the bytemap font information.*
- typedef struct [GRRLIB\\_Font](#) [GRRLIB\\_ttfFont](#)  
*Structure to hold the TTF information.*

## Enumerations

- enum [GRRLIB\\_blendMode](#) {  
[GRRLIB\\_BLEND\\_ALPHA](#) = 0, [GRRLIB\\_BLEND\\_ADD](#) = 1, [GRRLIB\\_BLEND\\_SCREEN](#) = 2, [GRRLIB\\_BLEND\\_MULTI](#)  
= 3,  
[GRRLIB\\_BLEND\\_INV](#) = 4 }
- GRRLIB Blending Modes.*

## Functions

- **GRR\_EXTERN** void \*xfb[2] **GRR\_INITS** (NULL, NULL)
- **GRR\_EXTERN** u32 fb **GRR\_INIT** (0)

## Variables

- **GRR\_EXTERN** GXRModeObj \* [rmode](#)  
*Video mode.*

### 9.1.1 Detailed Description

GRRLIB user include file.



## 9.2 GRRLIB\_\_inline.h File Reference

```
#include <grplib/GRRLIB_clipping.h>
#include <grplib/GRRLIB_collision.h>
#include <grplib/GRRLIB_fbComplex.h>
#include <grplib/GRRLIB_fbGX.h>
#include <grplib/GRRLIB_fbSimple.h>
#include <grplib/GRRLIB_handle.h>
#include <grplib/GRRLIB_pixel.h>
#include <grplib/GRRLIB_settings.h>
#include <grplib/GRRLIB_texSetup.h>
```

### Functions

- **INLINE void GRRLIB\_ClipReset** (void)  
*Reset the clipping to normal.*
- **INLINE void GRRLIB\_ClipDrawing** (const int x, const int y, const int width, const int height)  
*Clip the drawing area to an rectangle.*
- **INLINE bool GRRLIB\_PtInRect** (const int hotx, const int hoty, const int hotw, const int hoth, const int wpadx, const int wpady)  
*Determine whether the specified point lies within the specified rectangle.*
- **INLINE bool GRRLIB\_RectInRect** (const int rect1x, const int rect1y, const int rect1w, const int rect1h, const int rect2x, const int rect2y, const int rect2w, const int rect2h)  
*Determine whether a specified rectangle lies within another rectangle.*
- **INLINE bool GRRLIB\_RectOnRect** (const int rect1x, const int rect1y, const int rect1w, const int rect1h, const int rect2x, const int rect2y, const int rect2w, const int rect2h)  
*Determine whether a part of a specified rectangle lies on another rectangle.*
- **INLINE void GRRLIB\_NPlot** (const guVector v[], const u32 color[], const long n)  
*Draw an array of points.*
- **INLINE void GRRLIB\_NGone** (const guVector v[], const u32 color[], const long n)  
*Draw a polygon.*
- **INLINE void GRRLIB\_NGoneFilled** (const guVector v[], const u32 color[], const long n)  
*Draw a filled polygon.*
- **INLINE void GRRLIB\_GXEngine** (const guVector v[], const u32 color[], const long n, const u8 fmt)  
*Draws a vector.*
- **INLINE void GRRLIB\_FillScreen** (const u32 color)  
*Clear screen with a specific color.*
- **INLINE void GRRLIB\_Plot** (const f32 x, const f32 y, const u32 color)  
*Draw a dot.*
- **INLINE void GRRLIB\_Line** (const f32 x1, const f32 y1, const f32 x2, const f32 y2, const u32 color)  
*Draw a line.*
- **INLINE void GRRLIB\_Rectangle** (const f32 x, const f32 y, const f32 width, const f32 height, const u32 color, const bool filled)  
*Draw a rectangle.*
- **INLINE void GRRLIB\_SetHandle** (GRRLIB\_texImg \*tex, const int x, const int y)  
*Set a texture's X and Y handles.*
- **INLINE void GRRLIB\_SetMidHandle** (GRRLIB\_texImg \*tex, const bool enabled)  
*Center a texture's handles.*
- **INLINE u32 GRRLIB\_GetPixelFromtexImg** (const int x, const int y, const GRRLIB\_texImg \*tex)  
*Return the color value of a pixel from a GRRLIB\_texImg.*

- `INLINE void GRRLIB_SetPixelTotexImg (const int x, const int y, GRRLIB_texImg *tex, const u32 color)`  
*Set the color value of a pixel to a GRRLIB\_texImg.*
- `INLINE u32 GRRLIB_GetPixelFromFB (int x, int y)`  
*Reads a pixel directly from the FrontBuffer.*
- `INLINE void GRRLIB_SetPixelToFB (int x, int y, u32 pokeColor)`  
*Writes a pixel directly from the FrontBuffer.*
- `INLINE void GRRLIB_SetBlend (const GRRLIB_blendMode blendmode)`  
*Set a blending mode.*
- `INLINE GRRLIB_blendMode GRRLIB_GetBlend (void)`  
*Get the current blending mode.*
- `INLINE void GRRLIB_SetAntiAliasing (const bool aa)`  
*Turn anti-aliasing on/off.*
- `INLINE bool GRRLIB_GetAntiAliasing (void)`  
*Get current anti-aliasing setting.*
- `INLINE void GRRLIB_ClearTex (GRRLIB_texImg *tex)`  
*Clear a texture to transparent black.*
- `INLINE void GRRLIB_FlushTex (GRRLIB_texImg *tex)`  
*Write the contents of a texture in the data cache down to main memory.*
- `INLINE void GRRLIB_FreeTexture (GRRLIB_texImg *tex)`  
*Free memory allocated for texture.*

### 9.2.1 Detailed Description

GRRLIB inline function prototypes. Do not include `GRRLIB__inline.h` directly, include only `GRRLIB.h`.

## 9.3 GRRLIB\_\_lib.h File Reference

### Functions

- `GRRLIB_bytemapFont * GRRLIB_LoadBMF (const u8 my_bmf[])`  
*Load a ByteMap font structure from a buffer.*
- `void GRRLIB_FreeBMF (GRRLIB_bytemapFont *bmf)`  
*Free memory allocated by ByteMap fonts.*
- `void GRRLIB_InitTileSet (GRRLIB_texImg *tex, const u32 tilew, const u32 tileh, const u32 tilestart)`  
*Initialize a tile set.*
- `void GRRLIB_BMFX_FlipH (const GRRLIB_texImg *texsrc, GRRLIB_texImg *texdest)`  
*Flip texture horizontal.*
- `void GRRLIB_BMFX_FlipV (const GRRLIB_texImg *texsrc, GRRLIB_texImg *texdest)`  
*Flip texture vertical.*
- `void GRRLIB_BMFX_Grayscale (const GRRLIB_texImg *texsrc, GRRLIB_texImg *texdest)`  
*Change a texture to gray scale.*
- `void GRRLIB_BMFX_Sepia (const GRRLIB_texImg *texsrc, GRRLIB_texImg *texdest)`  
*Change a texture to sepia (old photo style).*
- `void GRRLIB_BMFX_Invert (const GRRLIB_texImg *texsrc, GRRLIB_texImg *texdest)`  
*Invert colors of the texture.*
- `void GRRLIB_BMFX_Blur (const GRRLIB_texImg *texsrc, GRRLIB_texImg *texdest, const u32 factor)`  
*A texture effect (Blur).*
- `void GRRLIB_BMFX_Scatter (const GRRLIB_texImg *texsrc, GRRLIB_texImg *texdest, const u32 factor)`

- A texture effect (Scatter).*

  - void [GRRLIB\\_BMFX\\_Pixelate](#) (const [GRRLIB\\_texImg](#) \*texsrc, [GRRLIB\\_texImg](#) \*texdest, const u32 factor)
- A texture effect (Pixelate).*

  - int [GRRLIB\\_Init](#) (void)
- Initialize GRRLIB.*

  - void [GRRLIB\\_Exit](#) (void)
- Call this before exiting your application.*

  - void [GRRLIB\\_Circle](#) (const f32 x, const f32 y, const f32 radius, const u32 color, const u8 filled)
- Draw a circle.*

  - int [GRRLIB\\_LoadFile](#) (const char \*filename, u8 \*\*data)
- Load a file to memory.*

  - [GRRLIB\\_texImg](#) \* [GRRLIB\\_LoadTextureFromFile](#) (const char \*filename)
- Load a texture from a file.*

  - bool [GRRLIB\\_ScrShot](#) (const char \*filename)
- Make a PNG screenshot.*

  - void [GRRLIB\\_Printf](#) (const f32 xpos, const f32 ypos, const [GRRLIB\\_texImg](#) \*tex, const u32 color, const f32 zoom, const char \*text,...)
- Print formatted output.*

  - void [GRRLIB\\_PrintBMF](#) (const f32 xpos, const f32 ypos, const [GRRLIB\\_bytemapFont](#) \*bmf, const char \*text,...)
- Print formatted output with a ByteMap font.*

  - void [GRRLIB\\_DrawImg](#) (const f32 xpos, const f32 ypos, const [GRRLIB\\_texImg](#) \*tex, const f32 degrees, const f32 scaleX, const f32 scaleY, const u32 color)
- Draw a texture.*

  - void [GRRLIB\\_DrawImgQuad](#) (const [guVector](#) pos[4], [GRRLIB\\_texImg](#) \*tex, const u32 color)
- Draw a textured quad.*

  - void [GRRLIB\\_DrawTile](#) (const f32 xpos, const f32 ypos, const [GRRLIB\\_texImg](#) \*tex, const f32 degrees, const f32 scaleX, const f32 scaleY, const u32 color, const int frame)
- Draw a tile.*

  - void [GRRLIB\\_DrawPart](#) (const f32 xpos, const f32 ypos, const f32 partx, const f32 party, const f32 partw, const f32 parth, const [GRRLIB\\_texImg](#) \*tex, const f32 degrees, const f32 scaleX, const f32 scaleY, const u32 color)
- Draw a part of a texture.*

  - void [GRRLIB\\_DrawTileQuad](#) (const [guVector](#) pos[4], [GRRLIB\\_texImg](#) \*tex, const u32 color, const int frame)
- Draw a tile in a quad.*

  - void [GRRLIB\\_Render](#) (void)
- Call this function after drawing.*

  - void [GRRLIB\\_Screen2Texture](#) (int posx, int posy, [GRRLIB\\_texImg](#) \*tex, bool clear)
- Make a snapshot of the screen in a texture WITHOUT ALPHA LAYER.*

  - void [GRRLIB\\_CompStart](#) (void)
- Start GX compositing process.*

  - void [GRRLIB\\_CompEnd](#) (int posx, int posy, [GRRLIB\\_texImg](#) \*tex)
- End GX compositing process (Make a snapshot of the screen in a texture WITH ALPHA LAYER).*

  - [GRRLIB\\_texImg](#) \* [GRRLIB\\_CreateEmptyTexture](#) (const u32 w, const u32 h)
- Create an empty texture.*

  - [GRRLIB\\_texImg](#) \* [GRRLIB\\_LoadTexture](#) (const u8 \*my\_img)
- Load a texture from a buffer.*

  - [GRRLIB\\_texImg](#) \* [GRRLIB\\_LoadTexturePNG](#) (const u8 \*my\_png)
- Load a texture from a buffer.*

  - [GRRLIB\\_texImg](#) \* [GRRLIB\\_LoadTextureJPG](#) (const u8 \*my\_jpg)
- Load a texture from a buffer.*

- [GRRLIB\\_texImg](#) \* [GRRLIB\\_LoadTextureJPGEx](#) (const u8 \*my\_jpg, const int)  
*Load a texture from a buffer.*
- [GRRLIB\\_texImg](#) \* [GRRLIB\\_LoadTextureBMP](#) (const u8 \*my\_bmp)  
*Load a texture from a buffer.*
- bool [GRRLIB\\_GeckoInit](#) ()  
*Initialize USB Gecko.*
- void [GRRLIB\\_GeckoPrintf](#) (const char \*text,...)  
*Print Gecko.*
- void [GRRLIB\\_SetBackgroundColour](#) (u8 r, u8 g, u8 b, u8 a)  
*Set the background parameter when screen is cleared.*
- void [GRRLIB\\_Camera3dSettings](#) (f32 posx, f32 posy, f32 posz, f32 upx, f32 upy, f32 upz, f32 lookx, f32 looky, f32 lookz)  
*Set the camera parameter (contributed my chris\_c aka DaShAmAn).*
- void [GRRLIB\\_3dMode](#) (f32 minDist, f32 maxDist, f32 fov, bool texturemode, bool normalmode)  
*Set up the position matrix (contributed by chris\_c aka DaShAmAn).*
- void [GRRLIB\\_2dMode](#) ()  
*Go back to 2D mode (contributed by chris\_c aka DaShAmAn).*
- void [GRRLIB\\_ObjectViewBegin](#) (void)  
*Init the object matrix to draw object.*
- void [GRRLIB\\_ObjectViewScale](#) (f32 scalx, f32 scaly, f32 scalz)  
*Scale the object matrix to draw object.*
- void [GRRLIB\\_ObjectViewRotate](#) (f32 angx, f32 angy, f32 angz)  
*Rotate the object matrix to draw object .*
- void [GRRLIB\\_ObjectViewTrans](#) (f32 posx, f32 posy, f32 posz)  
*Translate the object matrix to draw object.*
- void [GRRLIB\\_ObjectViewEnd](#) (void)  
*Concat the object and the view matrix and calculate the inverse normal matrix.*
- void [GRRLIB\\_ObjectView](#) (f32 posx, f32 posy, f32 posz, f32 angx, f32 angy, f32 angz, f32 scalx, f32 scaly, f32 scalz)  
*Set the view matrix to draw object (in this order scale, rotate AND trans).*
- void [GRRLIB\\_ObjectViewInv](#) (f32 posx, f32 posy, f32 posz, f32 angx, f32 angy, f32 angz, f32 scalx, f32 scaly, f32 scalz)  
*Set the view matrix to draw object (in this order scale, trans AND rotate).*
- void [GRRLIB\\_SetTexture](#) ([GRRLIB\\_texImg](#) \*tex, bool rep)  
*Set the texture to an object (contributed by chris\_c aka DaShAmAn).*
- void [GRRLIB\\_DrawTorus](#) (f32 r, f32 R, int nsides, int rings, bool filled, u32 col)  
*Draw a torus (with normal).*
- void [GRRLIB\\_DrawSphere](#) (f32 r, int lats, int longs, bool filled, u32 col)  
*Draw a sphere (with normal).*
- void [GRRLIB\\_DrawCube](#) (f32 size, bool filled, u32 col)  
*Draw a cube (with normal).*
- void [GRRLIB\\_DrawCylinder](#) (f32 r, f32 h, int d, bool filled, u32 col)  
*Draw a cylinder (with normal).*
- void [GRRLIB\\_DrawCone](#) (f32 r, f32 h, int d, bool filled, u32 col)  
*Draw a cone (with normal).*
- void [GRRLIB\\_DrawTessPanel](#) (f32 w, f32 wstep, f32 h, f32 hstep, bool filled, u32 col)  
*Draw a Tesselated panel (with normal).*
- void [GRRLIB\\_SetLightAmbient](#) (u32 ambientcolor)  
*Set ambient color.*
- void [GRRLIB\\_SetLightDiff](#) (u8 num, guVector pos, f32 distattn, f32 brightness, u32 lightcolor)  
*Set diffuse light parameters.*

- void [GRRLIB\\_SetLightSpec](#) (u8 num, guVector dir, f32 shy, u32 lightcolor, u32 speccolor)  
*Set specular light parameters.*
- void [GRRLIB\\_SetLightSpot](#) (u8 num, guVector pos, guVector lookat, f32 angAttn0, f32 angAttn1, f32 angAttn2, f32 distAttn0, f32 distAttn1, f32 distAttn2, u32 lightcolor)  
*Set Spot light parameters.*
- void [GRRLIB\\_SetLightOff](#) (void)  
*Set all lights off, like at init.*
- [GRRLIB\\_ttfFont](#) \* [GRRLIB\\_LoadTTF](#) (const u8 \*file\_base, s32 file\_size)  
*Load a TTF from a buffer.*
- void [GRRLIB\\_FreeTTF](#) ([GRRLIB\\_ttfFont](#) \*myFont)  
*Free memory allocated by TTF fonts.*
- void [GRRLIB\\_PrintfTTF](#) (int x, int y, [GRRLIB\\_ttfFont](#) \*myFont, const char \*string, unsigned int fontSize, const u32 color)  
*Print function for TTF font.*
- void [GRRLIB\\_PrintfTTFW](#) (int x, int y, [GRRLIB\\_ttfFont](#) \*myFont, const wchar\_t \*string, unsigned int fontSize, const u32 color)  
*Print function for TTF font.*
- u32 [GRRLIB\\_WidthTTF](#) ([GRRLIB\\_ttfFont](#) \*myFont, const char \*, unsigned int)  
*Get the width of a text in pixel.*
- u32 [GRRLIB\\_WidthTTFW](#) ([GRRLIB\\_ttfFont](#) \*myFont, const wchar\_t \*, unsigned int)  
*Get the width of a text in pixel.*

### 9.3.1 Detailed Description

GRRLIB library function prototypes. Do not include [GRRLIB\\_\\_lib.h](#) directly, include only [GRRLIB.h](#).



## Chapter 10

# Example Documentation

### 10.1 template/source/main.c

This example shows the minimum code required to use GRRLIB. It could be used as a template to start a new project. More elaborate examples can be found inside the *examples* folder.

```
/*=====
GRRLIB (GX Version)
- Template Code -

Minimum Code To Use GRRLIB
=====*/
#include <grplib.h>
#include <stdlib.h>
#include <wiiose/wpad.h>
int main(int argc, char **argv) {
    // Initialise the Graphics & Video subsystem
    GRRLIB_Init();
    // Initialise the Wiimotes
    WPAD_Init();
    // Loop forever
    while(1) {
        WPAD_ScanPads(); // Scan the Wiimotes
        // If [HOME] was pressed on the first Wiimote, break out of the loop
        if (WPAD_ButtonsDown(0) & WPAD_BUTTON_HOME) break;
        // -----
        // Place your drawing code here
        // -----
        GRRLIB_Render(); // Render the frame buffer to the TV
    }
    GRRLIB_Exit(); // Be a good boy, clear the memory allocated by GRRLIB
    exit(0); // Use exit() to exit a program, do not use 'return' from main()
}
```





# Index

## blend

GRRLIB\_drawSettings, 61

## data

GRRLIB\_bytemapChar, 57

## Everything in GRRLIB, 15

GRRLIB\_3dMode, 21

GRRLIB\_BLEND\_ADD, 21

GRRLIB\_BLEND\_ALPHA, 21

GRRLIB\_BLEND\_INV, 21

GRRLIB\_BLEND\_MULTI, 21

GRRLIB\_BLEND\_SCREEN, 21

GRRLIB\_blendMode, 21

GRRLIB\_BMFX\_Blur, 22

GRRLIB\_BMFX\_FlipH, 22

GRRLIB\_BMFX\_FlipV, 23

GRRLIB\_BMFX\_Grayscale, 23

GRRLIB\_BMFX\_Invert, 23

GRRLIB\_BMFX\_Pixelate, 24

GRRLIB\_BMFX\_Scatter, 24

GRRLIB\_BMFX\_Sepia, 25

GRRLIB\_Camera3dSettings, 25

GRRLIB\_Circle, 26

GRRLIB\_ClearTex, 26

GRRLIB\_ClipDrawing, 27

GRRLIB\_CompoEnd, 27

GRRLIB\_CompoStart, 27

GRRLIB\_CreateEmptyTexture, 28

GRRLIB\_DrawCone, 28

GRRLIB\_DrawCube, 28

GRRLIB\_DrawCylinder, 29

GRRLIB\_DrawImg, 29

GRRLIB\_DrawImgQuad, 30

GRRLIB\_DrawPart, 30

GRRLIB\_DrawSphere, 31

GRRLIB\_DrawTessPanel, 31

GRRLIB\_DrawTile, 32

GRRLIB\_DrawTileQuad, 32

GRRLIB\_DrawTorus, 33

GRRLIB\_Exit, 33

GRRLIB\_FillScreen, 33

GRRLIB\_FlushTex, 33

GRRLIB\_FreeBMF, 34

GRRLIB\_FreeTexture, 34

GRRLIB\_FreeTTF, 35

GRRLIB\_Geckolnit, 35

GRRLIB\_GeckoPrintf, 35

GRRLIB\_GetAntiAliasing, 36

GRRLIB\_GetBlend, 36

GRRLIB\_GetPixelFromFB, 36

GRRLIB\_GetPixelFromtexImg, 36

GRRLIB\_GXEngine, 37

GRRLIB\_Init, 37

GRRLIB\_InitTileSet, 38

GRRLIB\_Line, 38

GRRLIB\_LoadBMF, 39

GRRLIB\_LoadFile, 39

GRRLIB\_LoadTexture, 40

GRRLIB\_LoadTextureBMP, 40

GRRLIB\_LoadTextureFromFile, 40

GRRLIB\_LoadTextureJPG, 41

GRRLIB\_LoadTextureJPGEx, 41

GRRLIB\_LoadTexturePNG, 42

GRRLIB\_LoadTTF, 42

GRRLIB\_NGone, 42

GRRLIB\_NGoneFilled, 43

GRRLIB\_NPlot, 43

GRRLIB\_ObjectView, 44

GRRLIB\_ObjectViewInv, 44

GRRLIB\_ObjectViewRotate, 45

GRRLIB\_ObjectViewScale, 45

GRRLIB\_ObjectViewTrans, 45

GRRLIB\_Plot, 46

GRRLIB\_PrintBMF, 46

GRRLIB\_Printf, 47

GRRLIB\_PrintfTTF, 47

GRRLIB\_PrintfTTFW, 47

GRRLIB\_PtInRect, 48

GRRLIB\_Rectangle, 49

GRRLIB\_RectInRect, 49

GRRLIB\_RectOnRect, 50

GRRLIB\_Screen2Texture, 50

GRRLIB\_ScrShot, 51

GRRLIB\_SetAntiAliasing, 51

GRRLIB\_SetBackgroundColour, 51

GRRLIB\_SetBlend, 52

GRRLIB\_SetHandle, 52

GRRLIB\_SetLightAmbient, 52

GRRLIB\_SetLightDiff, 52

GRRLIB\_SetLightSpec, 53

GRRLIB\_SetLightSpot, 53

GRRLIB\_SetMidHandle, 54

GRRLIB\_SetPixelToFB, 54

GRRLIB\_SetPixelTotexImg, 55

GRRLIB\_SetTexture, 55

GRRLIB\_WidthTTF, 55

GRRLIB\_WidthTTFW, 56

RGBA, 20

- grrlib.h, [65](#)
- GRRLIB\_3dMode
  - Everything in GRRLIB, [21](#)
- GRRLIB\_\_inline.h, [67](#)
- GRRLIB\_\_lib.h, [68](#)
- GRRLIB\_BLEND\_ADD
  - Everything in GRRLIB, [21](#)
- GRRLIB\_BLEND\_ALPHA
  - Everything in GRRLIB, [21](#)
- GRRLIB\_BLEND\_INV
  - Everything in GRRLIB, [21](#)
- GRRLIB\_BLEND\_MULTI
  - Everything in GRRLIB, [21](#)
- GRRLIB\_BLEND\_SCREEN
  - Everything in GRRLIB, [21](#)
- GRRLIB\_blendMode
  - Everything in GRRLIB, [21](#)
- GRRLIB\_BMFX\_Blur
  - Everything in GRRLIB, [22](#)
- GRRLIB\_BMFX\_FlipH
  - Everything in GRRLIB, [22](#)
- GRRLIB\_BMFX\_FlipV
  - Everything in GRRLIB, [23](#)
- GRRLIB\_BMFX\_Grayscale
  - Everything in GRRLIB, [23](#)
- GRRLIB\_BMFX\_Invert
  - Everything in GRRLIB, [23](#)
- GRRLIB\_BMFX\_Pixelate
  - Everything in GRRLIB, [24](#)
- GRRLIB\_BMFX\_Scatter
  - Everything in GRRLIB, [24](#)
- GRRLIB\_BMFX\_Sepia
  - Everything in GRRLIB, [25](#)
- GRRLIB\_bytemapChar, [57](#)
  - data, [57](#)
  - height, [58](#)
  - relx, [58](#)
  - rely, [58](#)
  - width, [58](#)
- GRRLIB\_bytemapFont, [59](#)
  - name, [59](#)
  - nbChar, [60](#)
  - palette, [60](#)
  - version, [60](#)
- GRRLIB\_Camera3dSettings
  - Everything in GRRLIB, [25](#)
- GRRLIB\_Circle
  - Everything in GRRLIB, [26](#)
- GRRLIB\_ClearTex
  - Everything in GRRLIB, [26](#)
- GRRLIB\_ClipDrawing
  - Everything in GRRLIB, [27](#)
- GRRLIB\_CompoEnd
  - Everything in GRRLIB, [27](#)
- GRRLIB\_CompoStart
  - Everything in GRRLIB, [27](#)
- GRRLIB\_CreateEmptyTexture
  - Everything in GRRLIB, [28](#)
- GRRLIB\_DrawCone
  - Everything in GRRLIB, [28](#)
- GRRLIB\_DrawCube
  - Everything in GRRLIB, [28](#)
- GRRLIB\_DrawCylinder
  - Everything in GRRLIB, [29](#)
- GRRLIB\_DrawImg
  - Everything in GRRLIB, [29](#)
- GRRLIB\_DrawImgQuad
  - Everything in GRRLIB, [30](#)
- GRRLIB\_DrawPart
  - Everything in GRRLIB, [30](#)
- GRRLIB\_drawSettings, [60](#)
  - blend, [61](#)
  - lights, [61](#)
- GRRLIB\_DrawSphere
  - Everything in GRRLIB, [31](#)
- GRRLIB\_DrawTessPanel
  - Everything in GRRLIB, [31](#)
- GRRLIB\_DrawTile
  - Everything in GRRLIB, [32](#)
- GRRLIB\_DrawTileQuad
  - Everything in GRRLIB, [32](#)
- GRRLIB\_DrawTorus
  - Everything in GRRLIB, [33](#)
- GRRLIB\_Exit
  - Everything in GRRLIB, [33](#)
- GRRLIB\_FillScreen
  - Everything in GRRLIB, [33](#)
- GRRLIB\_FlushTex
  - Everything in GRRLIB, [33](#)
- GRRLIB\_Font, [61](#)
- GRRLIB\_FreeBMF
  - Everything in GRRLIB, [34](#)
- GRRLIB\_FreeTexture
  - Everything in GRRLIB, [34](#)
- GRRLIB\_FreeTTF
  - Everything in GRRLIB, [35](#)
- GRRLIB\_GeckoInit
  - Everything in GRRLIB, [35](#)
- GRRLIB\_GeckoPrintf
  - Everything in GRRLIB, [35](#)
- GRRLIB\_GetAntiAliasing
  - Everything in GRRLIB, [36](#)
- GRRLIB\_GetBlend
  - Everything in GRRLIB, [36](#)
- GRRLIB\_GetPixelFromFB
  - Everything in GRRLIB, [36](#)
- GRRLIB\_GetPixelFromtexImg
  - Everything in GRRLIB, [36](#)
- GRRLIB\_GXEngine
  - Everything in GRRLIB, [37](#)
- GRRLIB\_Init
  - Everything in GRRLIB, [37](#)
- GRRLIB\_InitTileSet
  - Everything in GRRLIB, [38](#)
- GRRLIB\_Line
  - Everything in GRRLIB, [38](#)

- GRRLIB\_LoadBMF
  - Everything in GRRLIB, 39
- GRRLIB\_LoadFile
  - Everything in GRRLIB, 39
- GRRLIB\_LoadTexture
  - Everything in GRRLIB, 40
- GRRLIB\_LoadTextureBMP
  - Everything in GRRLIB, 40
- GRRLIB\_LoadTextureFromFile
  - Everything in GRRLIB, 40
- GRRLIB\_LoadTextureJPG
  - Everything in GRRLIB, 41
- GRRLIB\_LoadTextureJPGEx
  - Everything in GRRLIB, 41
- GRRLIB\_LoadTexturePNG
  - Everything in GRRLIB, 42
- GRRLIB\_LoadTTF
  - Everything in GRRLIB, 42
- GRRLIB\_NGone
  - Everything in GRRLIB, 42
- GRRLIB\_NGoneFilled
  - Everything in GRRLIB, 43
- GRRLIB\_NPlot
  - Everything in GRRLIB, 43
- GRRLIB\_ImageView
  - Everything in GRRLIB, 44
- GRRLIB\_ImageViewInv
  - Everything in GRRLIB, 44
- GRRLIB\_ImageViewRotate
  - Everything in GRRLIB, 45
- GRRLIB\_ImageViewScale
  - Everything in GRRLIB, 45
- GRRLIB\_ImageViewTrans
  - Everything in GRRLIB, 45
- GRRLIB\_Plot
  - Everything in GRRLIB, 46
- GRRLIB\_PrintBMF
  - Everything in GRRLIB, 46
- GRRLIB\_Printf
  - Everything in GRRLIB, 47
- GRRLIB\_PrintfTTF
  - Everything in GRRLIB, 47
- GRRLIB\_PrintfTTFW
  - Everything in GRRLIB, 47
- GRRLIB\_PtInRect
  - Everything in GRRLIB, 48
- GRRLIB\_Rectangle
  - Everything in GRRLIB, 49
- GRRLIB\_RectInRect
  - Everything in GRRLIB, 49
- GRRLIB\_RectOnRect
  - Everything in GRRLIB, 50
- GRRLIB\_Screen2Texture
  - Everything in GRRLIB, 50
- GRRLIB\_ScrShot
  - Everything in GRRLIB, 51
- GRRLIB\_SetAntiAliasing
  - Everything in GRRLIB, 51
- GRRLIB\_SetBackgroundColour
  - Everything in GRRLIB, 51
- GRRLIB\_SetBlend
  - Everything in GRRLIB, 52
- GRRLIB\_SetHandle
  - Everything in GRRLIB, 52
- GRRLIB\_SetLightAmbient
  - Everything in GRRLIB, 52
- GRRLIB\_SetLightDiff
  - Everything in GRRLIB, 52
- GRRLIB\_SetLightSpec
  - Everything in GRRLIB, 53
- GRRLIB\_SetLightSpot
  - Everything in GRRLIB, 53
- GRRLIB\_SetMidHandle
  - Everything in GRRLIB, 54
- GRRLIB\_SetPixelToFB
  - Everything in GRRLIB, 54
- GRRLIB\_SetPixelTotexImg
  - Everything in GRRLIB, 55
- GRRLIB\_SetTexture
  - Everything in GRRLIB, 55
- GRRLIB\_texImg, 62
  - nbtileh, 63
  - nbtilew, 63
  - tiletex, 63
  - tileh, 63
  - tilestart, 63
  - tilew, 64
  - w, 64
- GRRLIB\_WidthTTF
  - Everything in GRRLIB, 55
- GRRLIB\_WidthTTFW
  - Everything in GRRLIB, 56
- height
  - GRRLIB\_bytemapChar, 58
- lights
  - GRRLIB\_drawSettings, 61
- name
  - GRRLIB\_bytemapFont, 59
- nbChar
  - GRRLIB\_bytemapFont, 60
- nbtileh
  - GRRLIB\_texImg, 63
- nbtilew
  - GRRLIB\_texImg, 63
- palette
  - GRRLIB\_bytemapFont, 60
- relx
  - GRRLIB\_bytemapChar, 58
- rely
  - GRRLIB\_bytemapChar, 58
- RGBA
  - Everything in GRRLIB, 20

tiledtex  
    GRRLIB\_texImg, [63](#)

tileh  
    GRRLIB\_texImg, [63](#)

tilestart  
    GRRLIB\_texImg, [63](#)

tilew  
    GRRLIB\_texImg, [64](#)

version  
    GRRLIB\_bytemapFont, [60](#)

w  
    GRRLIB\_texImg, [64](#)

width  
    GRRLIB\_bytemapChar, [58](#)