gLib2D

Beta2

Generated by Doxygen 1.7.1

Wed Nov 3 2010 13:11:30

Contents

Chapter 1

gLib2D Documentation

1.1 Introduction

gLib2D by Geecko - A simple, fast, light-weight 2D graphics library.

This library has been designed to replace the old graphics.c library and to simplify the use of the pspgu.

The goals : keep it simple, keep it small, keep it fast.

1.2 Known limitations

- Only rectangles can be used. Other primitives, like triangles or lines are just skipped.
- You can't get infos about current properties.
- When some 512∗512 rotated, colorized and scaled textures are rendered at a time, the framerate ∗could∗ go under 60 fps.

1.3 Installation

- Simply put glib2d.c and glib2d.h in your source directory.
- Then add glib2d.o and link "-lpng -ljpeg -lz -lpspgu -lm" in your Makefile.
- You're done !

1.4 License

This work is licensed under the Creative Commons BY-SA 3.0 License.

See `http://creativecommons.org/licenses/by-sa/3.0/` for more details.

1.5 Contact

Please report bugs or submit ideas at : `geecko.dev@free.fr`

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Data Structure Documentation

4.1 gImage Struct Reference

Image structure.

`#include <glib2d.h>`

Data Fields

- int tw
- int th
- int w
- int h
- float ratio
- bool swizzled
- bool can_blend
- gColor ∗ data

4.1.1 Detailed Description

Image structure.

4.1.2 Field Documentation

4.1.2.1 bool gImage::can_blend

Can the texture blend ?

4.1.2.2 gColor∗ gImage::data

Pointer to the texture raw data.

4.1.2.3 int gImage::h

Texture height, as seen when draw.

4.1.2.4 float gImage::ratio

Width/Height ratio.

4.1.2.5 bool gImage::swizzled

Is the texture swizzled ?

4.1.2.6 int gImage::th

Real texture height. A power of two.

4.1.2.7 int gImage::tw

Real texture width. A power of two.

4.1.2.8 int gImage::w

Texture width, as seen when draw.

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

- glib2d.h

Chapter 5

File Documentation

5.1 glib2d.h File Reference

gLib2D Header

Data Structures

- struct gImage

    Image structure.

Defines

- #define USE_PNG

    Choose if the PNG support is enabled.

- #define USE_JPEG

    Choose if the JPEG support is enabled.

- #define G_SCR_W (480)

    Screen width constant, in pixels.

- #define G_SCR_H (272)

    Screen height constant, in pixels.

- #define G_FALSE 0

    Boolean constant, false or 0.

- #define G_TRUE !0

    Boolean constant, true or 1.

- #define G_RGBA(r, g, b, a) ((r)|((g)<<8)|((b)<<16)|((a)<<24))

    Create a gColor.

- #define G_GET_R(color) (((color)) & 0xFF)

  Get red channel value from a gColor.

- #define G_GET_G(color) (((color)>>8) & 0xFF)

  Get green channel value from a gColor.

- #define G_GET_B(color) (((color)>>16) & 0xFF)

  Get blue channel value from a gColor.

- #define G_GET_A(color) (((color)>>24) & 0xFF)

  Get alpha channel value from a gColor.

- #define G_MODULATE(color, luminance, alpha)

  gColor modulation.

Typedefs

- typedef char bool

  Boolean type.

- typedef unsigned char gAlpha

  Alpha type.

- typedef unsigned int gColor

  Color type.

- typedef int gEnum

  Enumeration type.

Enumerations

- enum gColors {

  RED = 0xFF0000FF, **GREEN** = 0xFF00FF00, **BLUE** = 0xFFFF0000, **CYAN** = 0xFFFFFF00,

  MAGENTA = 0xFFFF00FF, **YELLOW** = 0xFF00FFFF, **AZURE** = 0xFFFF7F00, **VIOLET** = 0xFFFF007F,

  ROSE = 0xFF7F00FF, **ORANGE** = 0xFF007FFF, **CHARTREUSE** = 0xFF00FF7F, **SPRING_GREEN** = 0xFF7FFF00,

  WHITE = 0xFFFFFFFF, **LITEGRAY** = 0xFFBFBFBF, **GRAY** = 0xFF7F7F7F, **DARKGRAY** = 0xFF3F3F3F,

  BLACK = 0xFF000000 }

  Colors enumeration.

- enum gCoord_Mode {

  G_UP_LEFT, **G_UP_RIGHT**, **G_DOWN_RIGHT**, **G_DOWN_LEFT**,

  G_CENTER }

  Coord modes enumeration.

Functions

- void gClear (gColor color)

  Clears screen & depth buffer, starts rendering.

- void gClearZ ()

  Clears depth buffer.

- void gBegin (gImage ∗tex)

  Begins object rendering.

- void gEnd ()

  Ends object rendering.

- void gReset ()

  Resets current transformation and attribution.

- void gFlip (bool use_vsync)

  Flips the screen.

- void gAdd ()

  Pushes the current transformation & attribution to a new object.

- void gPush ()

  Saves the current transformation to stack.

- void gPop ()

  Restore the current transformation from stack.

- void gTexFree (gImage ∗∗tex)

  Frees an image & set the pointer to NULL.

- gImage ∗ gTexLoad (char path[ ], bool use_swizzle)

  Loads an image.

- void gResetCoord ()

  Resets the current coordinates.

- void gSetCoordMode (gEnum mode)

  Set coordinate mode.

- void gSetCoordXY (float x, float y)

  Sets the new position.

- void gSetCoordXYZ (float x, float y, float z)

  Sets the new position, with depth support.

- void gSetCoordXYRelative (float x, float y, bool use_rot)

  Sets the new position, relative to the current.

- void gSetCoordXYZRelative (float x, float y, float z, bool use_rot)

  Sets the new position, with depth support, relative to the current.

- void gResetCrop ()

  Resets the current crop.

- void gSetCropXY (int x, int y)

  Sets the new crop position.

- void gSetCropWH (int w, int h)

  Sets the new crop size.

- void gSetCropXYRelative (int x, int y)

  Sets the new crop position, relative to the current.

- void gSetCropWHRelative (int w, int h)

  Sets the new crop size, relative to the current.

- void gResetScale ()

  Resets the current scale.

- void gSetScale (float w, float h)

  Sets the new scale.

- void gSetScaleWH (int w, int h)

  Sets the new scale, in pixels.

- void gSetScaleRelative (float w, float h)

  Sets the new scale, relative to the current.

- void gSetScaleWHRelative (int w, int h)

  Sets the new scale, in pixels, relative to the current.

- void gResetColor ()

  Resets the current color.

- void gResetAlpha ()

  Resets the current alpha.

- void gSetColor (gColor color)

  Sets the new color.

- void gSetAlpha (gAlpha alpha)

  Sets the new alpha.

- void gSetAlphaRelative (int alpha)

  Sets the new alpha, relative to the current alpha.

- void gResetRotation ()

  Resets the current rotation.

- void gSetRotationRad (float radians)

  Sets the new rotation, in radians.

- void gSetRotation (float degrees)

  Sets the new rotation, in degrees.

- void gSetRotationRadRelative (float radians)

  Sets the new rotation, relative to the current, in radians.

- void gSetRotationRelative (float degrees)

  Sets the new rotation, relative to the current, in degrees.

- void gSetTexBlend (bool use)

  Use the alpha blending with the texture.

- void gSetTexLinear (bool use)

  Use the bilinear filter with the texture.

5.1.1 Detailed Description

gLib2D Header

Version

Beta 2

5.1.2 Define Documentation

5.1.2.1 #define G_MODULATE(*color*, *luminance*, *alpha*)

Value:

```
G_RGBA(luminance*G_GET_R(color)/255, \
       luminance*G_GET_G(color)/255, \
       luminance*G_GET_B(color)/255, \
       alpha     *G_GET_A(color)/255)
```

gColor modulation.

This macro modulates the luminance & alpha of a gColor. Input range is from 0 to 255.

5.1.2.2 #define G_RGBA(*r*, *g*, *b*, *a*) ((r)|((g)<<8)|((b)<<16)|((a)<<24))

Create a gColor.

This macro creates a gColor from 4 values, red, green, blue and alpha. Input range is from 0 to 255.

5.1.2.3 #define USE_JPEG

Choose if the JPEG support is enabled.

Otherwise, this part will be not compiled to gain some space. Enable this to get JPEG support, disable to avoid compilation errors when libjpeg is not linked in the Makefile.

5.1.2.4 #define USE_PNG

Choose if the PNG support is enabled.

Otherwise, this part will be not compiled to gain some space. Enable this to get PNG support, disable to avoid compilation errors when libpng is not linked in the Makefile.

5.1.3 Enumeration Type Documentation

5.1.3.1 enum gColors

Colors enumeration.

Primary, secondary, tertiary and grayscale colors are defined.

5.1.3.2 enum gCoord_Mode

Coord modes enumeration.

Choose where the coordinates correspond in the object. This can be a corner or the center.

5.1.4 Function Documentation

5.1.4.1 void gAdd ( )

Pushes the current transformation & attribution to a new object.

This function must be called during object rendering. This is a ∗basic∗ function.

5.1.4.2 void gBegin (gImage ∗ *tex*)

Begins object rendering.

Parameters

tex Pointer to a texture, pass NULL to get a colored object.

This function begins object rendering. Calls gReset(). Only one texture can be used, but multiple objects can be rendered at a time. gBegin() / gEnd() couple can be called multiple times in the loop, to render multiple textures.

5.1.4.3 void gClear (gColor *color*)

Clears screen & depth buffer, starts rendering.

Parameters

color	Screen clear color

This function clears the screen, and calls gClearZ() if depth coordinate is used in the loop. You MUST call this function at the beginning of the loop to start the render process. Will automatically init the GU.

5.1.4.4 void gClearZ ( )

Clears depth buffer.

This function clears the zbuffer to zero (z range 0-65535).

5.1.4.5 void gEnd ( )

Ends object rendering.

This function ends object rendering. Must be called after gBegin() to add objects to the display list. Automatically adapts pspgu functionnalities to get the best performance possible.

5.1.4.6 void gFlip (bool *use_vsync*)

Flips the screen.

Parameters

use_vsync	Limit FPS to 60 ?

This function must be called at the end of the loop. Inverts screen buffers to display the whole thing.

5.1.4.7 void gPop ( )

Restore the current transformation from stack.

This function must be called during object rendering. The stack is 64 saves high. Use it like the OpenGL one.

5.1.4.8 void gPush ( )

Saves the current transformation to stack.

This function must be called during object rendering. The stack is 64 saves high. Use it like the OpenGL one.

5.1.4.9 void gReset ( )

Resets current transformation and attribution.

This function must be called during object rendering. Calls gResetCoord(), gResetRotation(), gResetScale(), gResetColor(), gResetAlpha() and gResetCrop().

5.1.4.10 void gResetAlpha ( )

Resets the current alpha.

This function must be called during object rendering. Sets gSetAlpha() to 255.

5.1.4.11 void gResetColor ( )

Resets the current color.

This function must be called during object rendering. Sets gSetColor() to WHITE.

5.1.4.12 void gResetCoord ( )

Resets the current coordinates.

This function must be called during object rendering. Sets gSetCoordMode() to G_UP_LEFT and gSetCoordXYZ() to (0,0,0).

5.1.4.13 void gResetCrop ( )

Resets the current crop.

This function must be called during object rendering. Sets gSetCropXY() to (0;0) and gSetCropWH() to (tex->w,tex->h) or (10,10).

5.1.4.14 void gResetRotation ( )

Resets the current rotation.

This function must be called during object rendering. Sets gSetRotation() to 0°.

5.1.4.15 void gResetScale ( )

Resets the current scale.

This function must be called during object rendering. Sets the scale to the current image size or (10,10).

5.1.4.16 void gSetAlpha (gAlpha *alpha*)

Sets the new alpha.

Parameters

alpha The new alpha (0-255).

This function must be called during object rendering. Can be used for both textured and non-textured objects.

5.1.4.17 void gSetAlphaRelative (int *alpha*)

Sets the new alpha, relative to the current alpha.

Parameters

alpha The new alpha increment.

This function must be called during object rendering. Can be used for both textured and non-textured objects.

5.1.4.18 void gSetColor (gColor *color*)

Sets the new color.

Parameters

color The new color.

This function must be called during object rendering. Can be used to colorize a non-textured objet. Can also be used as a color mask for a texture.

5.1.4.19 void gSetCoordMode (gEnum *mode*)

Set coordinate mode.

Parameters

mode A gCoord_Mode.

This function must be called during object rendering. Defines where the coordinates correspond in the object. Works even if the texture is inverted.

5.1.4.20 void gSetCoordXY (float *x*, float *y*)

Sets the new position.

Parameters

x New x, in pixels.

y New y, in pixels.

This function must be called during object rendering.

5.1.4.21 void gSetCoordXYRelative (float *x*, float *y*, bool *use_rot*)

Sets the new position, relative to the current.

Parameters

x New x increment, in pixels.

y New y increment, in pixels.

use_rot Take account of the rotation ?

This function must be called during object rendering.

5.1.4.22 void gSetCoordXYZ (float *x*, float *y*, float *z*)

Sets the new position, with depth support.

Parameters

- ***x*** New x, in pixels.
- ***y*** New y, in pixels.
- ***z*** New z, in pixels. (0-65535)

This function must be called during object rendering.

5.1.4.23 void gSetCoordXYZRelative (float *x*, float *y*, float *z*, bool *use_rot*)

Sets the new position, with depth support, relative to the current.

Parameters

- ***x*** New x increment, in pixels.
- ***y*** New y increment, in pixels.
- ***z*** New z increment, in pixels.
- ***use_rot*** Take account of the rotation ?

This function must be called during object rendering.

5.1.4.24 void gSetCropWH (int *w*, int *h*)

Sets the new crop size.

Parameters

- ***w*** New width, in pixels.
- ***h*** New height, in pixels.

This function must be called during object rendering. Defines the rectangle size of the crop. If the rectangle is larger or next to the image, texture will be repeated. Useful for a tileset.

5.1.4.25 void gSetCropWHRelative (int *w*, int *h*)

Sets the new crop size, relative to the current.

Parameters

- ***w*** New width increment, in pixels.
- ***h*** New height increment, in pixels.

This function must be called during object rendering. Defines the rectangle size of the crop. If the rectangle is larger or next to the image, texture will be repeated. Useful for a tileset.

5.1.4.26 void gSetCropXY (int *x*, int *y*)

Sets the new crop position.

Parameters

- ***x*** New x, in pixels.
- ***y*** New y, in pixels.

This function must be called during object rendering. Defines the rectangle position of the crop. If the rectangle is larger or next to the image, texture will be repeated. Useful for a tileset.

5.1.4.27 void gSetCropXYRelative (int *x*, int *y*)

Sets the new crop position, relative to the current.

Parameters

- ***x*** New x increment, in pixels.
- ***y*** New y increment, in pixels.

This function must be called during object rendering. Defines the rectangle position of the crop. If the rectangle is larger or next to the image, texture will be repeated. Useful for a tileset.

5.1.4.28 void gSetRotation (float *degrees*)

Sets the new rotation, in degrees.

Parameters

- ***degrees*** The new angle.

This function must be called during object rendering. The rotation center is the actual coordinates.

5.1.4.29 void gSetRotationRad (float *radians*)

Sets the new rotation, in radians.

Parameters

- ***radians*** The new angle.

This function must be called during object rendering. The rotation center is the actual coordinates.

5.1.4.30 void gSetRotationRadRelative (float *radians*)

Sets the new rotation, relative to the current, in radians.

Parameters

- ***radians*** The new angle increment.

This function must be called during object rendering. The rotation center is the actual coordinates.

5.1.4.31 void gSetRotationRelative (float *degrees*)

Sets the new rotation, relative to the current, in degrees.

Parameters

degrees The new angle increment.

This function must be called during object rendering. The rotation center is the actual coordinates.

5.1.4.32 void gSetScale (float *w*, float *h*)

Sets the new scale.

Parameters

w Width scale factor.

h Height scale factor.

This function must be called during object rendering. gResetScale() is called, then width & height scale are multiplied by these values. Note: negative values can be passed to invert the image.

5.1.4.33 void gSetScaleRelative (float *w*, float *h*)

Sets the new scale, relative to the current.

Parameters

w Width scale factor.

h Height scale factor.

This function must be called during object rendering. Current width & height scale are multiplied by these values. Note: negative values can be passed to invert the image.

5.1.4.34 void gSetScaleWH (int *w*, int *h*)

Sets the new scale, in pixels.

Parameters

w New width, in pixels.

h New height, in pixels.

This function must be called during object rendering. Note: negative values can be passed to invert the image.

5.1.4.35 void gSetScaleWHRelative (int *w*, int *h*)

Sets the new scale, in pixels, relative to the current.

Parameters

- ***w*** New width to increment, in pixels.
- ***h*** New height to increment, in pixels.

This function must be called during object rendering. Note: negative values can be passed to invert the image.

5.1.4.36 void gSetTexBlend (bool *use*)

Use the alpha blending with the texture.

Parameters

- ***use*** G_TRUE to activate (better look, by default). G_FALSE to desactivate (better performance).

This function must be called during object rendering. Automaticaly disabled when gImage::can_blend is set to G_FALSE.

5.1.4.37 void gSetTexLinear (bool *use*)

Use the bilinear filter with the texture.

Parameters

- ***use*** G_TRUE to activate (better look, by default). G_FALSE to desactivate (better performance).

This function must be called during object rendering. Only useful when scaling.

5.1.4.38 void gTexFree (gImage ∗∗ *tex*)

Frees an image & set the pointer to NULL.

Parameters

- ***tex*** Pointer to the variable which contains the pointer.

This function is used to gain memory when an image is useless. Must pass the pointer to the variable which contains the pointer, to set it to NULL. This is a more secure approach.

5.1.4.39 gImage∗ gTexLoad (char *path[ ],* bool *use_swizzle*)

Loads an image.

Parameters

- ***path*** Path to the file.
- ***use_swizzle*** Pass G_TRUE to get _more_ speed (recommended). Pass G_FALSE to avoid artifacts.

Returns

Pointer to the image.

This function loads an image file. There is support for PNG & JPEG files (if USE_PNG and USE_JPEG are defined). Swizzling is enabled only for 16∗16+ textures (useless on small textures). Image support up to 512∗512 only.

Index