Glview Library Reference

QNX Software Systems Limited
1001 Farrar Road
Ottawa, Ontario
K2K 0B3
Canada

Voice: +1 613 591-0931
Fax: +1 613 591-3579
Email: info@qnx.com
Web: http://www.qnx.com/

Electronic edition published: Wednesday, October 8, 2014

©2012–2014, QNX Software Systems Limited, a subsidiary of BlackBerry Limited.
All rights reserved.

QNX Software Systems Limited
1001 Farrar Road
Ottawa, Ontario
K2K 0B3
Canada

Voice: +1 613 591-0931
Fax: +1 613 591-3579
Email: info@qnx.com
Web: http://www.qnx.com/

QNX, QNX CAR, Momentics, Neutrino, and Aviage are trademarks of BlackBerry Limited, which are registered and/or used in certain jurisdictions, and used under license by QNX Software Systems Limited. All other trademarks belong to their respective owners.

Electronic edition published: Wednesday, October 8, 2014

Table of Contents

Glview Library Reference

The `glview` library allows you to more easily develop apps with OpenGL ES 1.1 and OpenGL ES 2.0.

This header file provides all the functions used by `libglview`. The purpose of this library is to simplify development when you want to use OpenGL ES 1.1 and OpenGL ES 2.0 for rendering graphics in apps.

API component	Description
Constants in glview.h (p. 11)	Constants and macro definitions in the `glview` library.
Data types in glview.h (p. 13)	Structures, typedefs, and typed enumerations that are available in the `glview` library.
Functions in glview.h (p. 23)	Functions in the `glview` library.

Typographical conventions

Throughout this manual, we use certain typographical conventions to distinguish technical terms. In general, the conventions we use conform to those found in IEEE POSIX publications.

The following table summarizes our conventions:

Reference	Example
Code examples	`if( stream == NULL)`
Command options	`-lR`
Commands	`make`
Constants	`NULL`
Data types	`unsigned short`
Environment variables	***PATH***
File and pathnames	`/dev/null`
Function names	*exit()*
Keyboard chords	**Ctrl–Alt–Delete**
Keyboard input	`Username`
Keyboard keys	**Enter**
Program output	`login:`
Variable names	*stdin*
Parameters	*parm1*
User-interface components	**Navigator**
Window title	**Options**

We use an arrow in directions for accessing menu items, like this:

You'll find the Other... menu item under **Perspective** **Show View**.

We use notes, cautions, and warnings to highlight important messages:

Notes point out something important or useful.

Cautions tell you about commands or procedures that may have unwanted or undesirable side effects.

Warnings tell you about commands or procedures that could be dangerous to your files, your hardware, or even yourself.

Note to Windows users

In our documentation, we typically use a forward slash (/) as a delimiter in pathnames, including those pointing to Windows files. We also generally follow POSIX/UNIX filesystem conventions.

Technical support

Technical assistance is available for all supported products.

To obtain technical support for any QNX product, visit the Support area on our website (*www.qnx.com*). You'll find a wide range of support options, including community forums.

Chapter 1

Glview Library Reference

The `glview` library allows you to more easily develop apps with OpenGL ES 1.1 and OpenGL ES 2.0.

This header file provides all the functions used by `libglview`. The purpose of this library is to simplify development when you want to use OpenGL ES 1.1 and OpenGL ES 2.0 for rendering graphics in apps.

API component	Description
Constants in glview.h (p. 11)	Constants and macro definitions in the `glview` library.
Data types in glview.h (p. 13)	Structures, typedefs, and typed enumerations that are available in the `glview` library.
Functions in glview.h (p. 23)	Functions in the `glview` library.

Chapter 2
Constants in *glview.h*

Constants and macro definitions in the glview library.

Definitions in *glview.h*

Preprocessor macro definitions for the glview.h header file in the libglview library.

Defines:

```
#define GLVIEW_VERSION (1000001)
```

The version of the library.

The version number is determined as follows: (Major * 1000000) + (Minor * 1000) + Patch

See also:

glview_get_version() (p. 27)

```
#define GLVIEW_VERSION_STRING "1.0.1"
```

The string version of the library.

See also:

GLVIEW_VERSION

```
#define GLVIEW_SUCCESS (0)
```

A return code that indicates that a function completed successfully.

```
#define GLVIEW_FAILURE (-1)
```

A return code that indicates that a function did not complete successfully.

Generally, check `errno` for a reason.

Library:

`libglview`

Chapter 3
Data types in *glview.h*

Structures, typedefs, and typed enumerations that are available in the glview library.

glview_api_t

OpenGL versions An enumeration that indicates the version of OpenGL ES to use.

Synopsis:

```
#include <glview/glview.h>


typedef enum {
      GLVIEW_API_OPENGLES_11 = 0
      GLVIEW_API_OPENGLES_20 = 1
} glview_api_t;
```

Data:

GLVIEW_API_OPENGLES_11

GLVIEW_API_OPENGLES_20

Library:

`libglview`

Description:

background_callback

The callback that is invoked when the app is put into the background.

Synopsis:

```
#include <glview/glview.h>

typedef void(* background_callback)(void *callback_data);
```

Library:

`libglview`

Description:

An app is put into the background whenever it no longer occupies the entire screen. An app is put into the background when:

- the user puts the app into a thumbnail
- the user switches to another app
- an idle timeout occurs causing the device to go into standby mode

display_once_callback

The callback that is executed on a call to glview_display_once().

Synopsis:

```
#include <glview/glview.h>

typedef void(* display_once_callback)(void *callback_data);
```

Library:

`libglview`

Description:

This callback is provided as a parameter to the *glview_display_once()* (p. 24) function and will be called immediately from within that function call.

event_callback

The callback that is invoked when the app receives an event.

Synopsis:

```
#include <glview/glview.h>

typedef void(* event_callback)(bps_event_t *event, int domain, int code, void
 *callback_data);
```

Library:

`libglview`

Description:

This callback is invoked for every event read from the application's event queue. Processing events occurs during each iteration of the *glview_loop()* (p. 30) execution loop. Each event popped from the queue is passed into the event_callback.

finalize_callback

The callback that is fired once the execution loop is exited, but before glview is destroyed.

Synopsis:

```
#include <glview/glview.h>

typedef void(* finalize_callback)(void *callback_data);
```

Library:

`libglview`

Description:

The `finalize_callback` function is invoked within the *glview_loop()* (p. 30) function after the application is given an exit event but before the graphic stack is destroyed. Once the `finalize_callback` function returns, the graphic stack will be taken down and glview will be destroyed.

foreground_callback

The callback that is invoked when the app is put into the foreground.

Synopsis:

```
#include <glview/glview.h>

typedef void(* foreground_callback)(void *callback_data);
```

Library:

`libglview`

Description:

An app is put into the foreground whenever it goes fullscreen. An app is put into the foreground when:

- the user switches to the app
- the device wakes up after it has previously gone into standby mode with the app in the foreground

frame_callback

The callback that is invoked every time glview is about to display a frame.

Synopsis:

```
#include <glview/glview.h>

typedef void(* frame_callback)(void *callback_data);
```

Library:

`libglview`

Description:

The frame_callback is invoked within the *glview_loop()* (p. 30) function every time the application is expected to draw a single frame. By default this occurs 60 times a second.

The callback will not be invoked if the app is no longer expected to draw a frame. This can occur if the app has been minimized, or is hidden in some way (by the user's action, or on idle timeout). An app can still draw during this time (for example, to show a pause screen) by calling the *glview_display_once()* (p. 24) function.

initialize_callback

The callback that is invoked prior to entering the execution loop.

Synopsis:

```
#include <glview/glview.h>

typedef void(* initialize_callback)(void *callback_data);
```

Library:

`libglview`

Description:

The initialize_callback is fired from within the *glview_loop()* (p. 30) function. Prior to executing this callback, glview will have already completed initializing the graphics stack. Events are not processed until after this call returns.

resize_callback

The callback that is invoked when an orientation change occurs that the app must respond to.

Synopsis:

```
#include <glview/glview.h>

typedef void(* resize_callback)(unsigned int width, unsigned int height, void
 *callback_data);
```

Library:

`libglview`

Description:

The application descriptor file (bar-descriptor.xml) specifies the orientation behavior for an app. If the behavior is set to `default` or `auto-orient`, then any registered resize_callback will be invoked whenever the device is turned from landscape to portrait or vice-versa. Turning the device 180 degrees does not result in executing the `resize_callback`.

Chapter 4
Functions in *glview.h*

Functions in the glview library.

glview_display_once()

Provide a callback that will be called to draw a single frame.

Synopsis:

```
#include <glview/glview.h>

GLVIEW_API int glview_display_once(display_once_callback display_once_callback)
```

Arguments:

display_once_callback

The callback that will be fired to draw a single frame. This is the callback_data specified during the call to *glview_set_callback_data()* (p. 39).

Library:

`libglview`

Description:

When the app has been put into the background (due to idle timeout, or by user action) rendering no longer occurs in the glview execution loop. This function allows the app to draw a pause screen (or anything else the app wishes to display). The callback will be invoked immediately, and once the function completes glview will swap the buffers to immediately display what was drawn.

There are other common scenarios that may use this function:

- Whenever the app wishes to display something prior to entering the *glview_loop()* (p. 30)function.
- To dislay some kind of loading indicator while the app's thread is performing a long-running operation.

Returns:

`GLVIEW_SUCCESS` upon success, `GLVIEW_FAILURE` otherwise, with errno set to the following:

- `EPERM`: *glview_initialize()* (p. 29) was not called prior to *glview_display_once()* (p. 24).

glview_get_callback_data()

Get the callback data that will be passed into callback functions.

Synopsis:

```
#include <glview/glview.h>

GLVIEW_API int glview_get_callback_data(void **callback_data)
```

Arguments:

callback_data

The out parameter that gets set to the app-specific data obtained by calls to the *glview_set_callback_data()* (p. 39) function.

Library:

`libglview`

Description:

Retrieve the app specific data that is being passed into any invoked callback.

Returns:

`GLVIEW_SUCCESS` upon success, `GLVIEW_FAILURE` otherwise, with errno set to the following:

- `EINVAL`: A `NULL` callback_data was passed in.

glview_get_size()

Fetch the current surface size.

Synopsis:

```
#include <glview/glview.h>

GLVIEW_API int glview_get_size(unsigned int *width, unsigned int *height)
```

Arguments:

width

If this parameter is non-null then it will be set to the current width of the surface.

height

If this parameter is non-null then it will be set to the current height of the surface.

Library:

`libglview`

Description:

Query the surface to determine the size of the current surface.

Returns:

`GLVIEW_SUCCESS` upon success, `GLVIEW_FAILURE` otherwise, with errno set to the following:

- `EPERM`: `glview_intialize()` was not called prior to *glview_get_size()* (p. 26).

glview_get_version()

Retrieve the version of glview.

Synopsis:

```
#include <glview/glview.h>

GLVIEW_API int glview_get_version(void)
```

Arguments:

Library:

`libglview`

Description:

The *glview_get_version()* (p. 27) function retrieves the version of the glview library that your application is using.

Returns:

The version of the glview library using the scheme described for `GLVIEW_VERSION`.

glview_get_vsync()

Fetch the vsync setting.

Synopsis:

```
#include <glview/glview.h>

GLVIEW_API int glview_get_vsync(int *vsync)
```

Arguments:

vsync

The output parameter that the current vsync setting will be written to. This parameter cannot be `NULL`. A non-zero value indicates the number of video frames that will occur before the buffer is swapped. A zero value indicates that vsync is disabled.

Library:

`libglview`

Description:

Fetch the current vsync setting. By default, vsync is enabled (set to 1).

Returns:

`GLVIEW_SUCCESS` upon success, `GLVIEW_FAILURE` otherwise, with errno set to one of the following:

- `EPERM`: `glview_intialize()` was not called prior to *glview_get_vsync()* (p. 28).
- `EINVAL`: output parameter is `NULL`.

glview_initialize()

Initialize glview for use.

Synopsis:

```
#include <glview/glview.h>

GLVIEW_API int glview_initialize(glview_api_t api, frame_callback
frame_callback)
```

Arguments:

api

The version of OpenGL ES the app intends to use.

frame_callback

The callback that is fired to draw the frame. This callback is mandatory, and cannot be `NULL`.

Library:

`libglview`

Description:

This call must be made prior to calling any other glview function. After initialization callbacks can be registered to hook into various phases of the execution loop. Call the *glview_loop()* (p. 30) function in your app to start the glview execution loop.

Returns:

`GLVIEW_SUCCESS` upon success, `GLVIEW_FAILURE` otherwise, with errno set to one of the following:

- `EPERM`: glview has already been initialized.
- `EFAULT`: frame_callback is `NULL`.

glview_loop()

Enter glview's execution loop.

Synopsis:

```
#include <glview/glview.h>

GLVIEW_API int glview_loop(void)
```

Arguments:

Library:

`libglview`

Description:

This function contains glview's main execution loop. You must call *glview_initialize()* (p. 29) before calling this function.

This function invokes the initialize callback before entering the execution loop. Once started, the execution loop runs for the life of the app, invoking at various points the registered callbacks. The only mandatory callback is the display callback, which is called once per loop.

The execution loop occurs in the following order:

- Pull all events off the event queue and process them:
  - On background: fire the background callback (if registered)
  - On foreground: fire the foreground callback (if registered)
  - On orientation change: fire the resize callback (if registered)
  - Fire the event callback with event details (if registered)
  - On exit: break out of the execution loop.
- Call the display callback if the app is in the foreground.
- Swap the graphic's buffers.
- Repeat from the top.

When the user exits the app, the execution loop breaks, and the finalize callback is invoked.

The *glview_loop()* (p. 30) function does not return until the user exits the application. When *glview_loop()* (p. 30) returns, the graphics stack has already been taken down, and glview has already been destroyed. Any calls to glview after *glview_loop()* (p. 30) has returned will either fail or have unspecified behavior.

Returns:

GLVIEW_SUCCESS when the user exits the application, GLVIEW_FAILURE otherwise, with errno set to one of the following:

- EPERM: *glview_initialize()* (p. 29) was not called prior to *glview_loop()* (p. 30).
- EACCES: Failed to set BPS channel.

glview_register_background_callback()

Register a callback to be fired whenever the app is put into the background.

Synopsis:

```
#include <glview/glview.h>

GLVIEW_API int glview_register_background_callback(background_callback
background_callback)
```

Arguments:

background_callback

The callback to be fired on a background event. If this parameter is `NULL` then any previously set callback is disabled.

Library:

`libglview`

Description:

Calling *glview_register_background_callback()* (p. 32) with a `NULL` background callback will disable any previously set callback from being fired on a background event.

Returns:

`GLVIEW_SUCCESS`

glview_register_event_callback()

Register a callback to be fired for every event the app receives.

Synopsis:

```
#include <glview/glview.h>

GLVIEW_API int glview_register_event_callback(event_callback event_callback)
```

Arguments:

event_callback

The callback to be fired on an event. If this parameter is `NULL` then any previously set callback is disabled.

Library:

`libglview`

Description:

Calling *glview_register_event_callback()* (p. 33) with a `NULL` event callback will disable any previously set callback from being fired on an event.

Returns:

`GLVIEW_SUCCESS`

glview_register_finalize_callback()

Register a callback that will be fired once the execution loop is exited, but before glview is destroyed.

Synopsis:

```
#include <glview/glview.h>

GLVIEW_API int glview_register_finalize_callback(finalize_callback
finalize_callback)
```

Arguments:

finalize_callback

The function to call immediately after exiting the event loop. If finalize_callback is NULL, any previously registered callback will be disabled and no function will be called.

Library:

`libglview`

Description:

Returns:

`GLVIEW_SUCCESS`

glview_register_foreground_callback()

Register a callback to be fired whenever the app is put into the foreground.

Synopsis:

```
#include <glview/glview.h>

GLVIEW_API int glview_register_foreground_callback(foreground_callback
foreground_callback)
```

Arguments:

foreground_callback

The callback to be fired on a foreground event. If this parameter is `NULL` then any previously set callback is disabled.

Library:

`libglview`

Description:

Calling *glview_register_foreground_callback()* (p. 35) with a `NULL` foreground callback will disable any previously set callback from being fired on a foreground event.

Returns:

`GLVIEW_SUCCESS`

glview_register_frame_callback()

Register a callback that will be fired every time the app is expected to draw a frame.

Synopsis:

```
#include <glview/glview.h>

GLVIEW_API int glview_register_frame_callback(frame_callback frame_callback)
```

Arguments:

frame_callback

The function to call after all events have been processed, and the app is expected to draw the frame.

Library:

`libglview`

Description:

The display callback is initially set by the *glview_initialize()* (p. 29) function and is the only mandatory callback. An app can use this function to set a different display callback. A display callback must always be registered and valid. Setting the callback to `NULL` is invalid and will fail.

Returns:

`GLVIEW_SUCCESS` upon success, `GLVIEW_FAILURE` otherwise, with errno set to the following:

- `EFAULT`: Attempt to register a NULL display callback. In the event of `GLVIEW_FAILURE`, the previously registered frame_callback will remain.

glview_register_initialize_callback()

Register a callback that will be invoked prior to entering the execution loop.

Synopsis:

```
#include <glview/glview.h>

GLVIEW_API int glview_register_initialize_callback(initialize_callback
initialize_callback)
```

Arguments:

initialize_callback

The function to call prior to *glview_loop()* (p. 30) entering the execution loop. If initialize_callback is `NULL`, any previously registered callback will be disabled and no function will be called prior to entering the execution loop.

Library:

`libglview`

Description:

Returns:

`GLVIEW_SUCCESS`

glview_register_resize_callback()

Register a callback to be fired whenever an orientation occurs that the app must respond to.

Synopsis:

```
#include <glview/glview.h>

GLVIEW_API int glview_register_resize_callback(resize_callback resize_callback)
```

Arguments:

resize_callback

The callback to be fired on a resize event. If this parameter is `NULL` then any previously set callback is disabled.

Library:

`libglview`

Description:

Calling `glview_register_resize_callback` with a NULL `resize_callback` will prevent any previously set callback from being fired on a resize event. Any resize callback will never be fired if the app is set to landscape or portrait only.

Returns:

`GLVIEW_SUCCESS` upon success, `GLVIEW_FAILURE` otherwise, with errno set to the following:

- `EPERM`: *glview_initialize()* (p. 29) was not called prior to *glview_register_resize_callback()* (p. 38).

glview_set_callback_data()

Set the callback data that will be passed into callback functions.

Synopsis:

```
#include <glview/glview.h>

GLVIEW_API int glview_set_callback_data(void *callback_data)
```

Arguments:

callback_data

Application-specific data that will be passed as a parameter to each of the registered callbacks whenever they are fired. This can be changed by calling *glview_set_callback_data()* (p. 39) again. This can be set to `NULL`, in which case any previously set callback_data will no longer be passed into a callback.

Library:

`libglview`

Description:

Every callback that the app registers has, as a paramter, a `void`

- `callback_data`. Whatever `callback_data` that is set here will be passed into any registered callback when invoked by glview.

Calling this function with a `NULL` parameter will result in a NULL being passed into any registered callback when invoked by glview.

Returns:

`GLVIEW_SUCCESS`

glview_set_vsync()

Change the vsync setting.

Synopsis:

```
#include <glview/glview.h>

GLVIEW_API int glview_set_vsync(int vsync)
```

Arguments:

vsync

If this parameter is non-zero, it specifies the minimum number of video frames before a buffer swap. If it is zero then vsync is disabled.

Library:

`libglview`

Description:

By default, vsync is enabled (set to 1). Call this function to disable vsync or to change the vsync interval.

Returns:

`GLVIEW_SUCCESS` upon success, `GLVIEW_FAILURE` otherwise, with errno set to the following:

- `EPERM`: *glview_intialize()* (p. 29) was not called prior to *glview_set_vsync()* (p. 40).

Index

T