Contribution Guide

Contents

Asking a question

Questions about how to use GLFW should be asked either in the support section of the forum, under the Stack Overflow tag or Game Development tag on Stack Exchange or in the IRC channel `#glfw` on Freenode.

Questions about the design or implementation of GLFW or about future plans should be asked in the dev section of the forum or in the IRC channel. Please don't open a GitHub issue to discuss design questions without first checking with a maintainer.

Reporting a bug

If GLFW is behaving unexpectedly at run-time, start by setting an error callback. GLFW will often tell you the cause of an error via this callback. If it doesn't, that might be a separate bug.

If GLFW is crashing or triggering asserts, make sure that all your object handles and other pointers are valid.

For bugs where it makes sense, a short, self contained example is absolutely invaluable. Just put it inline in the body text. Note that if the bug is reproducible with one of the test programs that come with GLFW, just mention that instead.

Don't worry about adding too much information. Unimportant information can be abbreviated or removed later, but missing information can stall bug fixing, especially when your schedule doesn't align with that of the maintainer.

Please provide text as text, not as images. This includes code, error messages and any other text. Text in images cannot be found by other users searching for the same problem and may have to be re-typed by maintainers when debugging.

You don't need to manually indent your code or other text to quote it with GitHub Markdown; just surround it with triple backticks:

```
```
Some quoted text.
```
```

You can also add syntax highlighting by appending the common file extension:

```c
int five(void)
{
    return 5;
}
```

There are issue labels for both platforms and GPU manufacturers, so there is no need to mention these in the subject line. If you do, it will be removed when the issue is labeled.

If your bug is already reported, please add any new information you have, or if it already has everything, give it a :+1:.

Reporting a compile or link bug

Note: GLFW needs many system APIs to do its job, which on some platforms means linking to many system libraries. If you are using GLFW as a static library, that means your application needs to link to these in addition to GLFW.

Note: Check the Compiling GLFW guide and or Building applications guide for before opening an issue of this kind. Most issues are caused by a missing package or linker flag.

Always include the **operating system name and version** (e.g. `Windows 7 64-bit` or `Ubuntu 15.10`) and the **compiler name and version** (e.g. `Visual C++ 2015 Update 2`). If you are using an official release of GLFW, include the **GLFW release version** (e.g. `3.1.2`), otherwise include the **GLFW commit ID** (e.g. `3795d78b14ef06008889cc422a1fb8d642597751`) from Git.

Please also include the **complete build log** from your compiler and linker, even if it's long. It can always be shortened later, if necessary.

Quick template

```
OS and version:
Compiler version:
Release or commit:
Build log:
```

Reporting a segfault or other crash bug

Always include the **operating system name and version** (e.g. `Windows 7 64-bit` or `Ubuntu 15.10`). If you are using an official release of GLFW, include the **GLFW release version** (e.g. `3.1.2`), otherwise include the **GLFW commit ID** (e.g. `3795d78b14ef06008889cc422a1fb8d642597751`) from Git.

Please also include any **error messages** provided to your application via the error callback and the **full call stack** of the crash, or if the crash does not occur in debug mode, mention that instead.

Quick template

```
OS and version:
Release or commit:
Error messages:
Call stack:
```

Reporting a context creation bug

Note: Windows ships with graphics drivers that do not support OpenGL. If GLFW says that your machine lacks support for OpenGL, it very likely does. Install drivers from the computer manufacturer or graphics card manufacturer (Nvidia, AMD, Intel) to fix this.

Note: AMD only supports OpenGL ES on Windows via EGL. See the GLFW_CONTEXT_CREATION_API hint for how to select EGL.

Please verify that context creation also fails with the `glfwinfo` tool before reporting it as a bug. This tool is included in the GLFW source tree as `tests/glfwinfo.c` and is built along with the library. It has switches for all GLFW context and framebuffer hints. Run `glfwinfo -h` for a complete list.

Always include the **operating system name and version** (e.g. `Windows 7 64-bit` or `Ubuntu 15.10`). If you are using an official release of GLFW, include the **GLFW release version** (e.g. `3.1.2`), otherwise include the **GLFW commit ID** (e.g. `3795d78b14ef06008889cc422a1fb8d642597751`) from Git.

If you are running your program in a virtual machine, please mention this and include the **VM name and version** (e.g. `VirtualBox 5.1`).

Please also include the **GLFW version string** (`3.2.0 X11 EGL clock_gettime /dev/js`), as described here, the **GPU model and driver version** (e.g. `GeForce GTX660 with 352.79`), and the **output of `glfwinfo`** (with switches matching any hints you set in your code) when reporting this kind of bug. If this tool doesn't run on the machine, mention that instead.

Quick template

```
OS and version:
GPU and driver:
Release or commit:
Version string:
glfwinfo output:
```

Reporting a monitor or video mode bug

Note: On headless systems on some platforms, no monitors are reported. This causes glfwGetPrimaryMonitor to return `NULL`, which not all applications are prepared for.

Note: Some third-party tools report more video modes than are approved of by the OS. For safety and compatibility, GLFW only reports video modes the OS wants programs to use. This is not a bug.

The `monitors` tool is included in the GLFW source tree as `tests/monitors.c` and is built along with the library. It lists all information GLFW provides about monitors it detects.

Always include the **operating system name and version** (e.g. `Windows 7 64-bit` or `Ubuntu 15.10`). If you are using an official release of GLFW, include the **GLFW release version** (e.g. `3.1.2`), otherwise include the **GLFW commit ID** (e.g. `3795d78b14ef06008889cc422a1fb8d642597751`) from Git.

If you are running your program in a virtual machine, please mention this and include the **VM name and version** (e.g. `VirtualBox 5.1`).

Please also include any **error messages** provided to your application via the error callback and the **output of `monitors`** when reporting this kind of bug. If this tool doesn't run on the machine, mention this instead.

Quick template

```
OS and version:
Release or commit:
Error messages:
monitors output:
```

Reporting a window, input or event bug

Note: The exact ordering of related window events will sometimes differ.

Note: Window moving and resizing (by the user) will block the main thread on some platforms. This is not a bug. Set a refresh callback if you want to keep the window contents updated during a move or size operation.

The `events` tool is included in the GLFW source tree as `tests/events.c` and is built along with the library. It prints all information provided to every callback supported by GLFW as events occur. Each event is listed with the time and a unique number to make discussions about event logs easier. The tool has command-line options for creating multiple windows and full screen windows.

Always include the **operating system name and version** (e.g. `Windows 7 64-bit` or `Ubuntu 15.10`). If you are using an official release of GLFW, include the **GLFW release version** (e.g. `3.1.2`), otherwise include the **GLFW commit ID** (e.g. `3795d78b14ef06008889cc422a1fb8d642597751`) from Git.

If you are running your program in a virtual machine, please mention this and include the **VM name and version** (e.g. `VirtualBox 5.1`).

Please also include any **error messages** provided to your application via the error callback and if relevant, the **output of `events`** when reporting this kind of bug. If this tool doesn't run on the machine, mention this instead.

X11: If possible, please include what desktop environment (e.g. GNOME, Unity, KDE) and/or window manager (e.g. Openbox, dwm, Window Maker) you are running. If the bug is related to keyboard input, please include any input method (e.g. ibus, SCIM) you are using.

Quick template

```
OS and version:
Release or commit:
Error messages:
events output:
```

Reporting some other library bug

Always include the **operating system name and version** (e.g. `Windows 7 64-bit` or `Ubuntu 15.10`). If you are using an official release of GLFW, include the **GLFW release version** (e.g. `3.1.2`), otherwise include the **GLFW commit ID** (e.g. `3795d78b14ef06008889cc422a1fb8d642597751`) from Git.

Please also include any **error messages** provided to your application via the error callback, if relevant.

Quick template

```
OS and version:
Release or commit:
Error messages:
```

Reporting a documentation bug

If you found a bug in the documentation, including this file, then it's fine to just link to that web page or mention that source file. You don't need to match the source to the output or vice versa.

Reporting a website bug

If the bug is in the documentation (anything under `/docs/`) then please see the section above. Bugs in the rest of the site are reported to the website source repository.

Requesting a feature

Please explain why you need the feature and how you intend to use it. If you have a specific API design in mind, please add that as well. If you have or are planning to write code for the feature, see the section below.

If there already is a request for the feature you need, add your specific use case unless it is already mentioned. If it is, give it a :+1:.

Contributing a bug fix

Note: You must have all necessary intellectual property rights to any code you contribute. If you did not write the code yourself, you must explain where it came from and under what license you received it. Even code using the same license as GLFW may not be copied without attribution.

There is no preferred patch size. A one character fix is just as welcome as a thousand line one, if that is the appropriate size for the fix.

In addition to the code, a complete bug fix includes:

- Change log entry in `README.md`, describing the incorrect behavior
- Credits entries for all authors of the bug fix

Bug fixes will not be rejected because they don't include all the above parts, but please keep in mind that maintainer time is finite and that there are many other bugs and features to work on.

If the patch fixes a bug introduced after the last release, it should not get a change log entry.

If you haven't already, read the excellent article How to Write a Git Commit Message.

Contributing a feature

Note: You must have all necessary rights to any code you contribute. If you did not write the code yourself, you must explain where it came from and under what license. Even code using the same license as GLFW may not be copied without attribution.

Note: If you haven't already implemented the feature, check first if there already is an open issue for it and if it's already being developed in an experimental branch.

There is no preferred patch size. A one character change is just as welcome as one adding a thousand line one, if that is the appropriate size for the feature.

In addition to the code, a complete feature includes:

- Change log entry in `README.md`, listing all new symbols
- News page entry, briefly describing the feature
- Guide documentation, with minimal examples, in the relevant guide
- Reference documentation, with all applicable tags
- Cross-references and mentions in appropriate places

- Credits entries for all authors of the feature

If the feature requires platform-specific code, at minimum stubs must be added for the new platform function to all supported and experimental platforms.

If it adds a new callback, support for it must be added to `tests/event.c`.

If it adds a new monitor property, support for it must be added to `tests/monitor.c`.

If it adds a new OpenGL, OpenGL ES or Vulkan option or extension, support for it must be added to `tests/glfwinfo.c` and the behavior of the library when the extension is missing documented in `docs/compat.dox`.

If you haven't already, read the excellent article How to Write a Git Commit Message.

Features will not be rejected because they don't include all the above parts, but please keep in mind that maintainer time is finite and that there are many other features and bugs to work on.

Please also keep in mind that any part of the public API that has been included in a release cannot be changed until the next *major* version. Features can be added and existing parts can sometimes be overloaded (in the general sense of doing more things, not in the C++ sense), but code written to the API of one minor release should both compile and run on subsequent minor releases.