GLFW documentation

Generated by Doxygen 1.9.4

Chapter 1

TinyCThread API Reference

1.1 Introduction

TinyCThread is a minimal, portable implementation of basic threading classes for C.

They closely mimic the functionality and naming of the C11 standard, and should be easily replaceable with the corresponding standard variants.

1.2 Portability

The Win32 variant uses the native Win32 API for implementing the thread classes, while for other systems, the POSIX threads API (pthread) is used.

1.3 Miscellaneous

The following special keywords are available: _Thread_local.

For more detailed information, browse the different sections of this documentation. A good place to start is: tinycthread.h.

Chapter 2

Acknowledgements

GLFW exists because people around the world donated their time and lent their skills. This list only includes contributions to the main repository and excludes other invaluable contributions like language bindings and text and video tutorials.

- Bobyshev Alexander
- Laurent Aphecetche
- Matt Arsenault
- ashishgamedev
- David Avedissian
- Luca Bacci
- Keith Bauer
- John Bartholomew
- Coşku Baş
- Niklas Behrens
- Andrew Belt
- Nevyn Bengtsson
- Niklas Bergström
- Denis Bernard
- Doug Binks
- blanco
- Waris Boonyasiriwat
- Kyle Brenneman
- Rok Breulj
- Kai Burjack
- Martin Capitanio
- Nicolas Caramelli

- David Carlier
- Arturo Castro
- Chi-kwan Chan
- TheChocolateOre
- Joseph Chua
- Ian Clarkson
- Michał Cichoń
- Lambert Clara
- Anna Clarke
- Josh Codd
- Yaron Cohen-Tal
- Omar Cornut
- Andrew Corrigan
- Bailey Cosier
- Noel Cower
- CuriouserThing
- Jason Daly
- Jarrod Davis
- Olivier Delannoy
- Paul R. Deppe
- Michael Dickens
-
- Mario Dorn
- Wolfgang Draxinger
- Jonathan Dummer
- Ralph Eastwood
- Fredrik Ehnbom
- Robin Eklind
- Jan Ekström
- Siavash Eliasi
- Ahmad Fatoum
- Felipe Ferreira
- Michael Fogleman
- Jason Francis
- Gerald Franz
- Mário Freitas

- GeO4d
- Marcus Geelnard
- ghuser404
- Charles Giessen
- Ryan C. Gordon
- Stephen Gowen
- Kovid Goyal
- Kevin Grandemange
- Eloi Marín Gratacós
- Stefan Gustavson
- Andrew Gutekanst
- Stephen Gutekanst
- Jonathan Hale
- hdf89shfdfs
- Sylvain Hellegouarch
- Matthew Henry
- heromyth
- Lucas Hinderberger
- Paul Holden
- Warren Hu
- Charles Huber
- Brent Huisman
- illustris
- InKryption
- IntellectualKitty
- Aaron Jacobs
- JannikGM
- Erik S. V. Jansson
- jjYBdx4IL
- Toni Jovanoski
- Arseny Kapoulkine
- Cem Karan
- Osman Keskin
- Koray Kilinc
- Josh Kilmer
- Byunghoon Kim

- Cameron King
- Peter Knut
- Christoph Kubisch
- Yuri Kunde Schlesner
- Rokas Kupstys
- Konstantin Käfer
- Eric Larson
- Francis Lecavalier
- Jong Won Lee
- Robin Leffmann
- Glenn Lewis
- Shane Liesegang
- Anders Lindqvist
- Leon Linhart
- Marco Lizza
- Eyal Lotem
- Aaron Loucks
- Luflosi
- lukect
- Tristam MacDonald
- Hans Mackowiak
-
- Zbigniew Mandziejewicz
- Adam Marcus
- Célestin Marot
- Kyle McDonald
- David V. McKay
- David Medlock
- Bryce Mehring
- Jonathan Mercier
- Marcel Metz
- Liam Middlebrook
- Ave Milia
- Jonathan Miller
- Kenneth Miller
- Bruce Mitchener

- Jack Moffitt
- Jeff Molofee
- Alexander Monakov
- Pierre Morel
- Jon Morton
- Pierre Moulon
- Martins Mozeiko
- Pascal Muetschard
- Julian Møller
- ndogxj
- n3rdopolis
- Kristian Nielsen
- Kamil Nowakowski
- onox
- Denis Ovod
- Ozzy
- Andri Pálsson
- luz paz
- Peoro
- Braden Pellett
- Christopher Pelloux
- Arturo J. Pérez
- Vladimir Perminov
- Anthony Pesch
- Orson Peters
- Emmanuel Gil Peyrot
- Cyril Pichard
- Keith Pitt
- Stanislav Podgorskiy
- Konstantin Podsvirov
- Nathan Poirier
- Alexandre Pretyman
- Pablo Prietz
- przemekmirek
- pthom
- Guillaume Racicot

- Philip Rideout
- Eddie Ringle
- Max Risuhin
- Jorge Rodriguez
- Jari Ronkainen
- Luca Rood
- Ed Ropple
- Aleksey Rybalkin
- Mikko Rytkönen
- Riku Salminen
- Brandon Schaefer
- Sebastian Schuberth
- Christian Sdunek
- Matt Sealey
- Steve Sexton
- Arkady Shapkin
- Ali Sherief
- Yoshiki Shibukawa
- Dmitri Shuralyov
- Joao da Silva
- Daniel Sieger
- Daniel Skorupski
- Slemmie
- Anthony Smith
- Bradley Smith
- Cliff Smolinsky
- Patrick Snape
- Erlend Sogge Heggen
- Olivier Sohn
- Julian Squires
- Johannes Stein
- Pontus Stenetorp
- Michael Stocker
- Justin Stoecker
- Elviss Strazdins
- Paul Sultana

- Nathan Sweet
- TTK-Bandit
- Jared Tiala
- Sergey Tikhomirov
- Arthur Tombs
- TronicLabs
- Ioannis Tsakpinis
- Samuli Tuomola
- Matthew Turner
- urraka
- Elias Vanderstuyft
- Stef Velzel
- Jari Vetoniemi
- Ricardo Vieira
- Nicholas Vitovitch
- Simon Voordouw
- Corentin Wallez
- Torsten Walluhn
- Patrick Walton
- Xo Wang
- Jay Weisskopf
- Frank Wille
- Andy Williams
- Joel Winarske
- Richard A. Wilkes
- Tatsuya Yatagawa
- Ryogo Yoshimura
- Lukas Zanner
- Andrey Zholos
- Aihui Zhu
- Santi Zupancic
- Jonas Ådahl
- Lasse Öörni
- Leonard König
- All the unmentioned and anonymous contributors in the GLFW community, for bug reports, patches, feedback, testing and encouragement

Chapter 3

Building applications

This is about compiling and linking applications that use GLFW. For information on how to write such applications, start with the introductory tutorial. For information on how to compile the GLFW library itself, see Compiling GLFW.

This is not a tutorial on compilation or linking. It assumes basic understanding of how to compile and link a C program as well as how to use the specific compiler of your chosen development environment. The compilation and linking process should be explained in your C programming material and in the documentation for your development environment.

3.1 Including the GLFW header file

You should include the GLFW header in the source files where you use OpenGL or GLFW.

```
#include <GLFW/glfw3.h>
```

This header defines all the constants and declares all the types and function prototypes of the GLFW API. By default it also includes the OpenGL header from your development environment. See option macros below for how to select OpenGL ES headers and more.

The GLFW header also defines any platform-specific macros needed by your OpenGL header, so that it can be included without needing any window system headers.

It does this only when needed, so if window system headers are included, the GLFW header does not try to redefine those symbols. The reverse is not true, i.e. `windows.h` cannot cope if any Win32 symbols have already been defined.

In other words:

- Use the GLFW header to include OpenGL or OpenGL ES headers portably
- Do not include window system headers unless you will use those APIs directly
- If you do need such headers, include them before the GLFW header

If you are using an OpenGL extension loading library such as `glad`, the extension loader header should be included before the GLFW one. GLFW attempts to detect any OpenGL or OpenGL ES header or extension loader header included before it and will then disable the inclusion of the default OpenGL header. Most extension loaders also define macros that disable similar headers below it.

```
#include <glad/gl.h>
#include <GLFW/glfw3.h>
```

Both of these mechanisms depend on the extension loader header defining a known macro. If yours doesn't or you don't know which one your users will pick, the GLFW_INCLUDE_NONE macro will explicitly to prevent the GLFW header from including the OpenGL header. This will also allow you to include the two headers in any order.

```
#define GLFW_INCLUDE_NONE
#include <GLFW/glfw3.h>
#include <glad/gl.h>
```

3.1.1 GLFW header option macros

These macros may be defined before the inclusion of the GLFW header and affect its behavior.

GLFW_DLL is required on Windows when using the GLFW DLL, to tell the compiler that the GLFW functions are defined in a DLL.

The following macros control which OpenGL or OpenGL ES API header is included. Only one of these may be defined at a time.

Note

GLFW does not provide any of the API headers mentioned below. They are provided by your development environment or your OpenGL, OpenGL ES or Vulkan SDK, and most of them can be downloaded from the Khronos Registry.

GLFW_INCLUDE_GLCOREARB makes the GLFW header include the modern `GL/glcorearb.h` header (`OpenGL/gl3.h` on macOS) instead of the regular OpenGL header.

GLFW_INCLUDE_ES1 makes the GLFW header include the OpenGL ES 1.x `GLES/gl.h` header instead of the regular OpenGL header.

GLFW_INCLUDE_ES2 makes the GLFW header include the OpenGL ES 2.0 `GLES2/gl2.h` header instead of the regular OpenGL header.

GLFW_INCLUDE_ES3 makes the GLFW header include the OpenGL ES 3.0 `GLES3/gl3.h` header instead of the regular OpenGL header.

GLFW_INCLUDE_ES31 makes the GLFW header include the OpenGL ES 3.1 `GLES3/gl31.h` header instead of the regular OpenGL header.

GLFW_INCLUDE_ES32 makes the GLFW header include the OpenGL ES 3.2 `GLES3/gl32.h` header instead of the regular OpenGL header.

GLFW_INCLUDE_NONE makes the GLFW header not include any OpenGL or OpenGL ES API header. This is useful in combination with an extension loading library.

If none of the above inclusion macros are defined, the standard OpenGL `GL/gl.h` header (`OpenGL/gl.h` on macOS) is included, unless GLFW detects the inclusion guards of any OpenGL, OpenGL ES or extension loader header it knows about.

The following macros control the inclusion of additional API headers. Any number of these may be defined simultaneously, and/or together with one of the above macros.

GLFW_INCLUDE_VULKAN makes the GLFW header include the Vulkan `vulkan/vulkan.h` header in addition to any selected OpenGL or OpenGL ES header.

GLFW_INCLUDE_GLEXT makes the GLFW header include the appropriate extension header for the OpenGL or OpenGL ES header selected above after and in addition to that header.

GLFW_INCLUDE_GLU makes the header include the GLU header in addition to the header selected above. This should only be used with the standard OpenGL header and only for compatibility with legacy code. GLU has been deprecated and should not be used in new code.

Note

None of these macros may be defined during the compilation of GLFW itself. If your build includes GLFW and you define any these in your build files, make sure they are not applied to the GLFW sources.

3.2 Link with the right libraries

GLFW is essentially a wrapper of various platform-specific APIs and therefore needs to link against many different system libraries. If you are using GLFW as a shared library / dynamic library / DLL then it takes care of these links. However, if you are using GLFW as a static library then your executable will need to link against these libraries.

On Windows and macOS, the list of system libraries is static and can be hard-coded into your build environment. See the section for your development environment below. On Linux and other Unix-like operating systems, the list varies but can be retrieved in various ways as described below.

A good general introduction to linking is `Beginner's Guide to Linkers` by David Drysdale.

3.2.1 With MinGW or Visual C++ on Windows

The static version of the GLFW library is named `glfw3`. When using this version, it is also necessary to link with some libraries that GLFW uses.

When using MinGW to link an application with the static version of GLFW, you must also explicitly link with `gdi32`. Other toolchains including MinGW-w64 include it in the set of default libraries along with other dependencies like `user32` and `kernel32`.

The link library for the GLFW DLL is named `glfw3dll`. When compiling an application that uses the DLL version of GLFW, you need to define the GLFW_DLL macro *before* any inclusion of the GLFW header. This can be done either with a compiler switch or by defining it in your source code.

3.2.2 With CMake and GLFW source

This section is about using CMake to compile and link GLFW along with your application. If you want to use an installed binary instead, see With CMake and installed GLFW binaries.

With a few changes to your `CMakeLists.txt` you can have the GLFW source tree built along with your application.

Add the root directory of the GLFW source tree to your project. This will add the `glfw` target to your project.

```
add_subdirectory(path/to/glfw)
```

Once GLFW has been added, link your application against the `glfw` target. This adds the GLFW library and its link-time dependencies as it is currently configured, the include directory for the GLFW header and, when applicable, the GLFW_DLL macro.

```
target_link_libraries(myapp glfw)
```

Note that the `glfw` target does not depend on OpenGL, as GLFW loads any OpenGL, OpenGL ES or Vulkan libraries it needs at runtime. If your application calls OpenGL directly, instead of using a modern extension loader library, use the OpenGL CMake package.

```
find_package(OpenGL REQUIRED)
```

If OpenGL is found, the `OpenGL::GL` target is added to your project, containing library and include directory paths. Link against this like any other library.

```
target_link_libraries(myapp OpenGL::GL)
```

For a minimal example of a program and GLFW sources built with CMake, see the `GLFW CMake Starter` on GitHub.

3.2.3 With CMake and installed GLFW binaries

This section is about using CMake to link GLFW after it has been built and installed. If you want to build it along with your application instead, see With CMake and GLFW source.

With a few changes to your `CMakeLists.txt` you can locate the package and target files generated when GLFW is installed.

```
find_package(glfw3 3.4 REQUIRED)
```

Once GLFW has been added to the project, link against it with the `glfw` target. This adds the GLFW library and its link-time dependencies, the include directory for the GLFW header and, when applicable, the GLFW_DLL macro.

```
target_link_libraries(myapp glfw)
```

Note that the `glfw` target does not depend on OpenGL, as GLFW loads any OpenGL, OpenGL ES or Vulkan libraries it needs at runtime. If your application calls OpenGL directly, instead of using a modern extension loader library, use the OpenGL CMake package.

```
find_package(OpenGL REQUIRED)
```

If OpenGL is found, the `OpenGL::GL` target is added to your project, containing library and include directory paths. Link against this like any other library.

```
target_link_libraries(myapp OpenGL::GL)
```

3.2.4 With makefiles and pkg-config on Unix

GLFW supports `pkg-config`, and the `glfw3.pc` pkg-config file is generated when the GLFW library is built and is installed along with it. A pkg-config file describes all necessary compile-time and link-time flags and dependencies needed to use a library. When they are updated or if they differ between systems, you will get the correct ones automatically.

A typical compile and link command-line when using the static version of the GLFW library may look like this:

```
cc $(pkg-config --cflags glfw3) -o myprog myprog.c $(pkg-config --static --libs glfw3)
```

If you are using the shared version of the GLFW library, omit the `--static` flag.

```
cc $(pkg-config --cflags glfw3) -o myprog myprog.c $(pkg-config --libs glfw3)
```

You can also use the `glfw3.pc` file without installing it first, by using the `PKG_CONFIG_PATH` environment variable.

```
env PKG_CONFIG_PATH=path/to/glfw/src cc $(pkg-config --cflags glfw3) -o myprog myprog.c $(pkg-config --libs
       glfw3)
```

The dependencies do not include OpenGL, as GLFW loads any OpenGL, OpenGL ES or Vulkan libraries it needs at runtime. If your application calls OpenGL directly, instead of using a modern extension loader library, you should add the `gl` pkg-config package.

```
cc $(pkg-config --cflags glfw3 gl) -o myprog myprog.c $(pkg-config --libs glfw3 gl)
```

3.2.5 With Xcode on macOS

If you are using the dynamic library version of GLFW, add it to the project dependencies.

If you are using the static library version of GLFW, add it and the Cocoa, OpenGL and IOKit frameworks to the project as dependencies. They can all be found in `/System/Library/Frameworks`.

3.2.6 With command-line on macOS

It is recommended that you use pkg-config when building from the command line on macOS. That way you will get any new dependencies added automatically. If you still wish to build manually, you need to add the required frameworks and libraries to your command-line yourself using the `-l` and `-framework` switches.

If you are using the dynamic GLFW library, which is named `libglfw.3.dylib`, do:

```
cc -o myprog myprog.c -lglfw -framework Cocoa -framework OpenGL -framework IOKit
```

If you are using the static library, named `libglfw3.a`, substitute `-lglfw3` for `-lglfw`.

Note that you do not add the `.framework` extension to a framework when linking against it from the command-line.

Note

Your machine may have `libGL.*.dylib` style OpenGL library, but that is for the X Window System and will not work with the macOS native version of GLFW.

Chapter 4

Standards conformance

This guide describes the various API extensions used by this version of GLFW. It lists what are essentially implementation details, but which are nonetheless vital knowledge for developers intending to deploy their applications on a wide range of machines.

The information in this guide is not a part of GLFW API, but merely preconditions for some parts of the library to function on a given machine. Any part of this information may change in future versions of GLFW and that will not be considered a breaking API change.

4.1 X11 extensions, protocols and IPC standards

As GLFW uses Xlib directly, without any intervening toolkit library, it has sole responsibility for interacting well with the many and varied window managers in use on Unix-like systems. In order for applications and window managers to work well together, a number of standards and conventions have been developed that regulate behavior outside the scope of the X11 API; most importantly the Inter-Client Communication Conventions Manual (ICCCM) and Extended Window Manager Hints (EWMH) standards.

GLFW uses the _MOTIF_WM_HINTS window property to support borderless windows. If the running window manager does not support this property, the GLFW_DECORATED hint will have no effect.

GLFW uses the ICCCM WM_DELETE_WINDOW protocol to intercept the user attempting to close the GLFW window. If the running window manager does not support this protocol, the close callback will never be called.

GLFW uses the EWMH _NET_WM_PING protocol, allowing the window manager notify the user when the application has stopped responding, i.e. when it has ceased to process events. If the running window manager does not support this protocol, the user will not be notified if the application locks up.

GLFW uses the EWMH _NET_WM_STATE_FULLSCREEN window state to tell the window manager to make the GLFW window full screen. If the running window manager does not support this state, full screen windows may not work properly. GLFW has a fallback code path in case this state is unavailable, but every window manager behaves slightly differently in this regard.

GLFW uses the EWMH _NET_WM_BYPASS_COMPOSITOR window property to tell a compositing window manager to un-redirect full screen GLFW windows. If the running window manager uses compositing but does not support this property then additional copying may be performed for each buffer swap of full screen windows.

GLFW uses the clipboard manager protocol to push a clipboard string (i.e. selection) owned by a GLFW window about to be destroyed to the clipboard manager. If there is no running clipboard manager, the clipboard string will be unavailable once the window has been destroyed.

GLFW uses the X drag-and-drop protocol to provide file drop events. If the application originating the drag does not support this protocol, drag and drop will not work.

GLFW uses the XRandR 1.3 extension to provide multi-monitor support. If the running X server does not support this version of this extension, multi-monitor support will not function and only a single, desktop-spanning monitor will be reported.

GLFW uses the XRandR 1.3 and Xf86vidmode extensions to provide gamma ramp support. If the running X server does not support either or both of these extensions, gamma ramp support will not function.

GLFW uses the Xkb extension and detectable auto-repeat to provide keyboard input. If the running X server does not support this extension, a non-Xkb fallback path is used.

GLFW uses the XInput2 extension to provide raw, non-accelerated mouse motion when the cursor is disabled. If the running X server does not support this extension, regular accelerated mouse motion will be used.

GLFW uses both the XRender extension and the compositing manager to support transparent window framebuffers. If the running X server does not support this extension or there is no running compositing manager, the GLFW_↩ TRANSPARENT_FRAMEBUFFER framebuffer hint will have no effect.

GLFW uses both the Xcursor extension and the freedesktop cursor conventions to provide an expanded set of standard cursor shapes. If the running X server does not support this extension or the current cursor theme does not support the conventions, the GLFW_RESIZE_NWSE_CURSOR, GLFW_RESIZE_NESW_CURSOR and GLFW_↩ NOT_ALLOWED_CURSOR shapes will not be available and other shapes may use legacy images.

4.2 Wayland protocols and IPC standards

As GLFW uses libwayland directly, without any intervening toolkit library, it has sole responsibility for interacting well with every compositor in use on Unix-like systems. Most of the features are provided by the core protocol, while cursor support is provided by the libwayland-cursor helper library, EGL integration by libwayland-egl, and keyboard handling by libxkbcommon. In addition, GLFW uses some protocols from wayland-protocols to provide additional features if the compositor supports them.

GLFW uses xkbcommon 0.5.0 to provide key and text input support. Earlier versions are not supported.

GLFW uses the xdg-shell protocol to provide better window management. This protocol is part of wayland-protocols 1.12, and is mandatory for GLFW to display a window.

GLFW uses the relative pointer protocol alongside the pointer constraints protocol to implement disabled cursor. These two protocols are part of wayland-protocols 1.1, and mandatory at build time. If the running compositor does not support both of these protocols, disabling the cursor will have no effect.

GLFW uses the idle inhibit protocol to prohibit the screensaver from starting. This protocol is part of wayland-protocols 1.6, and mandatory at build time. If the running compositor does not support this protocol, the screensaver may start even for full screen windows.

GLFW uses the xdg-decoration protocol to request decorations to be drawn around its windows. This protocol is part of wayland-protocols 1.15, and mandatory at build time. If the running compositor does not support this protocol, a very simple frame will be drawn by GLFW itself, using the viewporter protocol alongside subsurfaces. This protocol is part of wayland-protocols 1.4, and mandatory at build time. If the running compositor does not support this protocol either, no decorations will be drawn around windows.

4.3 GLX extensions

The GLX API is the default API used to create OpenGL contexts on Unix-like systems using the X Window System.

GLFW uses the GLX 1.3 `GLXFBConfig` functions to enumerate and select framebuffer pixel formats. If GLX 1.3 is not supported, glfwInit will fail.

GLFW uses the `GLX_MESA_swap_control`, `GLX_EXT_swap_control` and `GLX_SGI_swap_control` extensions to provide vertical retrace synchronization (or *vsync*), in that order of preference. Where none of these extension are available, calling glfwSwapInterval will have no effect.

GLFW uses the `GLX_ARB_multisample` extension to create contexts with multisampling anti-aliasing. Where this extension is unavailable, the `GLFW_SAMPLES` hint will have no effect.

GLFW uses the `GLX_ARB_create_context` extension when available, even when creating OpenGL contexts of version 2.1 and below. Where this extension is unavailable, the `GLFW_CONTEXT_VERSION_MAJOR` and `GLFW_CONTEXT_VERSION_MINOR` hints will only be partially supported, the `GLFW_CONTEXT_DEBUG` hint will have no effect, and setting the `GLFW_OPENGL_PROFILE` or `GLFW_OPENGL_FORWARD_COMPAT` hints to `GLFW_TRUE` will cause glfwCreateWindow to fail.

GLFW uses the `GLX_ARB_create_context_profile` extension to provide support for context profiles. Where this extension is unavailable, setting the `GLFW_OPENGL_PROFILE` hint to anything but `GLFW_OPENGL_ANY_PROFILE`, or setting `GLFW_CLIENT_API` to anything but `GLFW_OPENGL_API` or `GLFW_NO_API` will cause glfwCreateWindow to fail.

GLFW uses the `GLX_ARB_context_flush_control` extension to provide control over whether a context is flushed when it is released (made non-current). Where this extension is unavailable, the `GLFW_CONTEXT_RELEASE_BEHAVIOR` hint will have no effect and the context will always be flushed when released.

GLFW uses the `GLX_ARB_framebuffer_sRGB` and `GLX_EXT_framebuffer_sRGB` extensions to provide support for sRGB framebuffers. Where both of these extensions are unavailable, the `GLFW_SRGB_CAPABLE` hint will have no effect.

4.4 WGL extensions

The WGL API is used to create OpenGL contexts on Microsoft Windows and other implementations of the Win32 API, such as Wine.

GLFW uses either the `WGL_EXT_extension_string` or the `WGL_ARB_extension_string` extension to check for the presence of all other WGL extensions listed below. If both are available, the EXT one is preferred. If neither is available, no other extensions are used and many GLFW features related to context creation will have no effect or cause errors when used.

GLFW uses the `WGL_EXT_swap_control` extension to provide vertical retrace synchronization (or *vsync*). Where this extension is unavailable, calling glfwSwapInterval will have no effect.

GLFW uses the `WGL_ARB_pixel_format` and `WGL_ARB_multisample` extensions to create contexts with multisampling anti-aliasing. Where these extensions are unavailable, the `GLFW_SAMPLES` hint will have no effect.

GLFW uses the `WGL_ARB_create_context` extension when available, even when creating OpenGL contexts of version 2.1 and below. Where this extension is unavailable, the `GLFW_CONTEXT_VERSION_MAJOR` and `GLFW_CONTEXT_VERSION_MINOR` hints will only be partially supported, the `GLFW_CONTEXT_DEBUG` hint will have no effect, and setting the `GLFW_OPENGL_PROFILE` or `GLFW_OPENGL_FORWARD_COMPAT` hints to `GLFW_TRUE` will cause glfwCreateWindow to fail.

GLFW uses the `WGL_ARB_create_context_profile` extension to provide support for context profiles. Where this extension is unavailable, setting the `GLFW_OPENGL_PROFILE` hint to anything but `GLFW_OPENGL_ANY_PROFILE` will cause glfwCreateWindow to fail.

GLFW uses the `WGL_ARB_context_flush_control` extension to provide control over whether a context is flushed when it is released (made non-current). Where this extension is unavailable, the `GLFW_CONTEXT_RELEASE_BEHAVIOR` hint will have no effect and the context will always be flushed when released.

GLFW uses the `WGL_ARB_framebuffer_sRGB` and `WGL_EXT_framebuffer_sRGB` extensions to provide support for sRGB framebuffers. Where both of these extension are unavailable, the `GLFW_SRGB_CAPABLE` hint will have no effect.

4.5 OpenGL on macOS

Support for OpenGL 3.2 and above was introduced with OS X 10.7 and even then only forward-compatible, core profile contexts are supported. Support for OpenGL 4.1 was introduced with OS X 10.9, also limited to forward-compatible, core profile contexts. There is also still no mechanism for requesting debug contexts or no-error contexts. Versions of Mac OS X earlier than 10.7 support at most OpenGL version 2.1.

Because of this, on OS X 10.7 and later, the `GLFW_CONTEXT_VERSION_MAJOR` and `GLFW_CONTEXT_VERSION_MINOR` hints will cause glfwCreateWindow to fail if given version 3.0 or 3.1. The `GLFW_OPENGL_PROFILE` hint must be set to `GLFW_OPENGL_CORE_PROFILE` when creating OpenGL 3.2 and later contexts. The `GLFW_CONTEXT_DEBUG` and `GLFW_CONTEXT_NO_ERROR` hints are ignored.

Also, on Mac OS X 10.6 and below, the `GLFW_CONTEXT_VERSION_MAJOR` and `GLFW_CONTEXT_VERSION_MINOR` hints will fail if given a version above 2.1, setting the `GLFW_OPENGL_PROFILE` or `GLFW_OPENGL_FORWARD_COMPAT` hints to a non-default value will cause glfwCreateWindow to fail and the `GLFW_CONTEXT_DEBUG` hint is ignored.

4.6 Vulkan loader and API

By default, GLFW uses the standard system-wide Vulkan loader to access the Vulkan API on all platforms except macOS. This is installed by both graphics drivers and Vulkan SDKs. If either the loader or at least one minimally functional ICD is missing, glfwVulkanSupported will return `GLFW_FALSE` and all other Vulkan-related functions will fail with an GLFW_API_UNAVAILABLE error.

4.7 Vulkan WSI extensions

The Vulkan WSI extensions are used to create Vulkan surfaces for GLFW windows on all supported platforms.

GLFW uses the `VK_KHR_surface` and `VK_KHR_win32_surface` extensions to create surfaces on Microsoft Windows. If any of these extensions are not available, glfwGetRequiredInstanceExtensions will return an empty list and window surface creation will fail.

GLFW uses the `VK_KHR_surface` and either the `VK_MVK_macos_surface` or `VK_EXT_metal_surface` extensions to create surfaces on macOS. If any of these extensions are not available, glfwGetRequiredInstanceExtensions will return an empty list and window surface creation will fail.

GLFW uses the `VK_KHR_surface` and either the `VK_KHR_xlib_surface` or `VK_KHR_xcb_surface` extensions to create surfaces on X11. If `VK_KHR_surface` or both `VK_KHR_xlib_surface` and `VK_KHR_xcb_surface` are not available, glfwGetRequiredInstanceExtensions will return an empty list and window surface creation will fail.

GLFW uses the `VK_KHR_surface` and `VK_KHR_wayland_surface` extensions to create surfaces on Wayland. If any of these extensions are not available, glfwGetRequiredInstanceExtensions will return an empty list and window surface creation will fail.

Chapter 5

Compiling GLFW

This is about compiling the GLFW library itself. For information on how to build applications that use GLFW, see Building applications.

5.1 Using CMake

GLFW behaves like most other libraries that use CMake so this guide mostly describes the standard configure, generate and compile sequence. If you are already familiar with this from other projects, you may want to focus on the Installing dependencies and CMake options sections for GLFW-specific information.

GLFW uses `CMake` to generate project files or makefiles for your chosen development environment. To compile GLFW, first generate these files with CMake and then use them to compile the GLFW library.

If you are on Windows and macOS you can `download CMake` from their site.

If you are on a Unix-like system such as Linux, FreeBSD or Cygwin or have a package system like Fink, MacPorts or Homebrew, you can install its CMake package.

CMake is a complex tool and this guide will only show a few of the possible ways to set up and compile GLFW. The CMake project has their own much more detailed `CMake user guide` that includes everything in this guide not specific to GLFW. It may be a useful companion to this one.

5.1.1 Installing dependencies

The C/C++ development environments in Visual Studio, Xcode and MinGW come with all necessary dependencies for compiling GLFW, but on Unix-like systems like Linux and FreeBSD you will need a few extra packages.

5.1.1.1 Dependencies for X11

To compile GLFW for X11, you need to have the X11 development packages installed. They are not needed to build or run programs that use GLFW.

On Debian and derivates like Ubuntu and Linux Mint the `xorg-dev` meta-package pulls in the development packages for all of X11.

```
sudo apt install xorg-dev
```

On Fedora and derivatives like Red Hat the X11 extension packages `libXcursor-devel`, `libXi-devel`, `libXinerama-devel` and `libXrandr-devel` required by GLFW pull in all its other dependencies.

```
sudo dnf install libXcursor-devel libXi-devel libXinerama-devel libXrandr-devel
```

On FreeBSD the X11 headers are installed along the end-user X11 packages, so if you have an X server running you should have the headers as well. If not, install the `xorgproto` package.

```
pkg install xorgproto
```

On Cygwin the `libXcursor-devel`, `libXi-devel`, `libXinerama-devel`, `libXrandr-devel` and `libXrender-devel` packages in the Libs section of the GUI installer will install all the headers and other development related files GLFW requires for X11.

Once you have the required dependencies, move on to Generating build files with CMake.

5.1.1.2 Dependencies for Wayland and X11

To compile GLFW for both Wayland and X11, you need to have the X11, Wayland and xkbcommon development packages installed. They are not needed to build or run programs that use GLFW. You will also need to set the GLFW_BUILD_WAYLAND CMake option in the next step when generating build files.

On Debian and derivates like Ubuntu and Linux Mint you will need the `libwayland-dev`, `libxkbcommon-dev` and `wayland-protocols` packages and the `xorg-dev` meta-package. These will pull in all other dependencies.

```
sudo apt install libwayland-dev libxkbcommon-dev wayland-protocols xorg-dev
```

On Fedora and derivatives like Red Hat you will need the `wayland-devel`, `libxkbcommon-devel`, `wayland-protocols-devel`, `libXcursor-devel`, `libXi-devel`, `libXinerama-devel` and `libXrandr-devel` packages. These will pull in all other dependencies.

```
sudo dnf install wayland-devel libxkbcommon-devel wayland-protocols-devel libXcursor-devel libXi-devel
       libXinerama-devel libXrandr-devel
```

On FreeBSD you will need the `wayland`, `libxkbcommon` and `wayland-protocols` packages. The X11 headers are installed along the end-user X11 packages, so if you have an X server running you should have the headers as well. If not, install the `xorgproto` package.

```
pkg install wayland libxkbcommon wayland-protocols xorgproto
```

Once you have the required dependencies, move on to Generating build files with CMake.

5.1.2 Generating build files with CMake

Once you have all necessary dependencies it is time to generate the project files or makefiles for your development environment. CMake needs two paths for this:

- the path to the root directory of the GLFW source tree (not its `src` subdirectory)
- the path to the directory where the generated build files and compiled binaries will be placed

If these are the same, it is called an in-tree build, otherwise it is called an out-of-tree build.

Out-of-tree builds are recommended as they avoid cluttering up the source tree. They also allow you to have several build directories for different configurations all using the same source tree.

A common pattern when building a single configuration is to have a build directory named `build` in the root of the source tree.

5.1.2.1 Generating with the CMake GUI

Start the CMake GUI and set the paths to the source and build directories described above. Then press *Configure* and *Generate*.

If you wish change any CMake variables in the list, press *Configure* and then *Generate* to have the new values take effect. The variable list will be populated after the first configure step.

By default GLFW will use X11 on Linux and other Unix-like systems other than macOS. To include support for Wayland as well, set the GLFW_BUILD_WAYLAND option in the GLFW section of the variable list, then apply the new value as described above.

Once you have generated the project files or makefiles for your chosen development environment, move on to Compiling the library.

5.1.2.2 Generating with command-line CMake

To make a build directory, pass the source and build directories to the `cmake` command. These can be relative or absolute paths. The build directory is created if it doesn't already exist.

```
cmake -S path/to/glfw -B path/to/build
```

It is common to name the build directory `build` and place it in the root of the source tree when only planning to build a single configuration.

```
cd path/to/glfw
cmake -S . -B build
```

Without other flags these will generate Visual Studio project files on Windows and makefiles on other platforms. You can choose other targets using the `-G` flag.

```
cmake -S path/to/glfw -B path/to/build -G Xcode
```

By default GLFW will use X11 on Linux and other Unix-like systems other than macOS. To also include support for Wayland, set the GLFW_BUILD_WAYLAND CMake option.

```
cmake -S path/to/glfw -B path/to/build -D GLFW_BUILD_WAYLAND=1
```

Once you have generated the project files or makefiles for your chosen development environment, move on to Compiling the library.

5.1.3 Compiling the library

You should now have all required dependencies and the project files or makefiles necessary to compile GLFW. Go ahead and compile the actual GLFW library with these files as you would with any other project.

With Visual Studio open `GLFW.sln` and use the Build menu. With Xcode open `GLFW.xcodeproj` and use the Project menu.

With Linux, macOS and other forms of Unix, run `make`.

```
cd path/to/build
make
```

With MinGW, it is `mingw32-make`.

```
cd path/to/build
mingw32-make
```

Any CMake build directory can also be built with the `cmake` command and the `--build` flag.

```
cmake --build path/to/build
```

This will run the platform specific build tool the directory was generated for.

Once the GLFW library is compiled you are ready to build your application, linking it to the GLFW library. See Building applications for more information.

5.2 CMake options

The CMake files for GLFW provide a number of options, although not all are available on all supported platforms. Some of these are de facto standards among projects using CMake and so have no `GLFW_` prefix.

If you are using the GUI version of CMake, these are listed and can be changed from there. If you are using the command-line version of CMake you can use the `ccmake` ncurses GUI to set options. Some package systems like Ubuntu and other distributions based on Debian GNU/Linux have this tool in a separate `cmake-curses-gui` package.

Finally, if you don't want to use any GUI, you can set options from the `cmake` command-line with the `-D` flag.

```
cmake -S path/to/glfw -B path/to/build -D BUILD_SHARED_LIBS=ON
```

5.2.1 Shared CMake options

BUILD_SHARED_LIBS determines whether GLFW is built as a static library or as a DLL / shared library / dynamic library. This is disabled by default, producing a static GLFW library. This variable has no `GLFW_` prefix because it is defined by CMake. If you want to change the library only for GLFW when it is part of a larger project, see GLFW_LIBRARY_TYPE.

GLFW_LIBRARY_TYPE allows you to override BUILD_SHARED_LIBS only for GLFW, without affecting other libraries in a larger project. When set, the value of this option must be a valid CMake library type. Set it to `STATIC` to build GLFW as a static library, `SHARED` to build it as a shared library / dynamic library / DLL, or `OBJECT` to make GLFW a CMake object library.

GLFW_BUILD_EXAMPLES determines whether the GLFW examples are built along with the library. This is enabled by default unless GLFW is being built as a sub-project of a larger CMake project.

GLFW_BUILD_TESTS determines whether the GLFW test programs are built along with the library. This is enabled by default unless GLFW is being built as a sub-project of a larger CMake project.

GLFW_BUILD_DOCS determines whether the GLFW documentation is built along with the library. This is enabled by default if `Doxygen` is found by CMake during configuration.

5.2.2 Win32 specific CMake options

GLFW_BUILD_WIN32 determines whether to include support for Win32 when compiling the library. This option is only available when compiling for Windows. This is enabled by default.

USE_MSVC_RUNTIME_LIBRARY_DLL determines whether to use the DLL version or the static library version of the Visual C++ runtime library. When enabled, the DLL version of the Visual C++ library is used. This is enabled by default.

On CMake 3.15 and later you can set the standard CMake `CMAKE_MSVC_RUNTIME_LIBRARY` variable instead of this GLFW-specific option.

GLFW_USE_HYBRID_HPG determines whether to export the `NvOptimusEnablement` and `AmdPowerXpressRequestHighPerformance` symbols, which force the use of the high-performance GPU on Nvidia Optimus and AMD PowerXpress systems. These symbols need to be exported by the EXE to be detected by the driver, so the override will not work if GLFW is built as a DLL. This is disabled by default, letting the operating system and driver decide.

5.2.3 macOS specific CMake options

GLFW_BUILD_COCOA determines whether to include support for Cocoa when compiling the library. This option is only available when compiling for macOS. This is enabled by default.

5.2.4 Unix-like system specific CMake options

GLFW_BUILD_WAYLAND determines whether to include support for Wayland when compiling the library. This option is only available when compiling for Linux and other Unix-like systems other than macOS. This is disabled by default.

GLFW_BUILD_X11 determines whether to include support for X11 when compiling the library. This option is only available when compiling for Linux and other Unix-like systems other than macOS. This is enabled by default.

5.3 Cross-compilation with CMake and MinGW

Both Cygwin and many Linux distributions have MinGW or MinGW-w64 packages. For example, Cygwin has the `mingw64-i686-gcc` and `mingw64-x86_64-gcc` packages for 32- and 64-bit version of MinGW-w64, while Debian GNU/Linux and derivatives like Ubuntu have the `mingw-w64` package for both.

GLFW has CMake toolchain files in the `CMake` subdirectory that set up cross-compilation of Windows binaries. To use these files you set the `CMAKE_TOOLCHAIN_FILE` CMake variable with the `-D` flag add an option when configuring and generating the build files.

```
cmake -S path/to/glfw -B path/to/build -D CMAKE_TOOLCHAIN_FILE=path/to/file
```

The exact toolchain file to use depends on the prefix used by the MinGW or MinGW-w64 binaries on your system. You can usually see this in the /usr directory. For example, both the Ubuntu and Cygwin MinGW-w64 packages have `/usr/x86_64-w64-mingw32` for the 64-bit compilers, so the correct invocation would be:

```
cmake -S path/to/glfw -B path/to/build -D CMAKE_TOOLCHAIN_FILE=CMake/x86_64-w64-mingw32.cmake
```

The path to the toolchain file is relative to the path to the GLFW source tree passed to the `-S` flag, not to the current directory.

For more details see the CMake toolchain guide.

5.4 Compiling GLFW manually

If you wish to compile GLFW without its CMake build environment then you will have to do at least some of the platform detection yourself. There are preprocessor macros for enabling support for the platforms (window systems) available. There are also optional, platform-specific macros for various features.

When building, GLFW will expect the necessary configuration macros to be defined on the command-line. The GLFW CMake files set these as private compile definitions on the GLFW target but if you compile the GLFW sources manually you will need to define them yourself.

The window system is used to create windows, handle input, monitors, gamma ramps and clipboard. The options are:

- **_GLFW_COCOA** to use the Cocoa frameworks
- **_GLFW_WIN32** to use the Win32 API

- **_GLFW_X11** to use the X Window System
- **_GLFW_WAYLAND** to use the Wayland API (incomplete)

The **_GLFW_WAYLAND** and **_GLFW_X11** macros may be combined and produces a library that attempts to detect the appropriate platform at initialization.

If you are building GLFW as a shared library / dynamic library / DLL then you must also define **_GLFW_BUILD_DLL**. Otherwise, you must not define it.

If you are using a custom name for the Vulkan, EGL, GLX, OSMesa, OpenGL, GLESv1 or GLESv2 library, you can override the default names by defining those you need of **_GLFW_VULKAN_LIBRARY**, **_GLFW_EGL_LIBRARY**, **_GLFW_GLX_LIBRARY**, **_GLFW_OSMESA_LIBRARY**, **_GLFW_OPENGL_LIBRARY**, **_GLFW_GLESV1_LIBRARY** and **_GLFW_GLESV2_LIBRARY**. Otherwise, GLFW will use the built-in default names.

Note

None of the GLFW header option macros may be defined during the compilation of GLFW. If you define any of these in your build files, make sure they are not applied to the GLFW sources.

Chapter 6

Context guide

This guide introduces the OpenGL and OpenGL ES context related functions of GLFW. For details on a specific function in this category, see the Context reference. There are also guides for the other areas of the GLFW API.

- Introduction to the API
- Window guide
- Vulkan guide
- Monitor guide
- Input guide

6.1 Context objects

A window object encapsulates both a top-level window and an OpenGL or OpenGL ES context. It is created with glfwCreateWindow and destroyed with glfwDestroyWindow or glfwTerminate. See Window creation for more information.

As the window and context are inseparably linked, the window object also serves as the context handle.

To test the creation of various kinds of contexts and see their properties, run the `glfwinfo` test program.

Note

Vulkan does not have a context and the Vulkan instance is created via the Vulkan API itself. If you will be using Vulkan to render to a window, disable context creation by setting the GLFW_CLIENT_API hint to `GLFW_NO↩_API`. For more information, see the Vulkan guide.

6.1.1 Context creation hints

There are a number of hints, specified using glfwWindowHint, related to what kind of context is created. See context related hints in the window guide.

6.1.2 Context object sharing

When creating a window and its OpenGL or OpenGL ES context with glfwCreateWindow, you can specify another window whose context the new one should share its objects (textures, vertex and element buffers, etc.) with.

```
GLFWwindow* second_window = glfwCreateWindow(640, 480, "Second Window", NULL, first_window);
```

Object sharing is implemented by the operating system and graphics driver. On platforms where it is possible to choose which types of objects are shared, GLFW requests that all types are shared.

See the relevant chapter of the OpenGL or OpenGL ES reference documents for more information. The name and number of this chapter unfortunately varies between versions and APIs, but has at times been named *Shared Objects and Multiple Contexts*.

GLFW comes with a barebones object sharing example program called `sharing`.

6.1.3 Offscreen contexts

GLFW doesn't support creating contexts without an associated window. However, contexts with hidden windows can be created with the GLFW_VISIBLE window hint.

```
glfwWindowHint(GLFW_VISIBLE, GLFW_FALSE);
GLFWwindow* offscreen_context = glfwCreateWindow(640, 480, "", NULL, NULL);
```

The window never needs to be shown and its context can be used as a plain offscreen context. Depending on the window manager, the size of a hidden window's framebuffer may not be usable or modifiable, so framebuffer objects are recommended for rendering with such contexts.

You should still process events as long as you have at least one window, even if none of them are visible.

6.1.4 Windows without contexts

You can disable context creation by setting the GLFW_CLIENT_API hint to `GLFW_NO_API`. Windows without contexts must not be passed to glfwMakeContextCurrent or glfwSwapBuffers.

6.2 Current context

Before you can make OpenGL or OpenGL ES calls, you need to have a current context of the correct type. A context can only be current for a single thread at a time, and a thread can only have a single context current at a time.

When moving a context between threads, you must make it non-current on the old thread before making it current on the new one.

The context of a window is made current with glfwMakeContextCurrent.

```
glfwMakeContextCurrent(window);
```

The window of the current context is returned by glfwGetCurrentContext.

```
GLFWwindow* window = glfwGetCurrentContext();
```

The following GLFW functions require a context to be current. Calling any these functions without a current context will generate a GLFW_NO_CURRENT_CONTEXT error.

- glfwSwapInterval
- glfwExtensionSupported
- glfwGetProcAddress

6.3 Buffer swapping

See Buffer swapping in the window guide.

6.4 OpenGL and OpenGL ES extensions

One of the benefits of OpenGL and OpenGL ES is their extensibility. Hardware vendors may include extensions in their implementations that extend the API before that functionality is included in a new version of the OpenGL or OpenGL ES specification, and some extensions are never included and remain as extensions until they become obsolete.

An extension is defined by:

- An extension name (e.g. `GL_ARB_gl_spirv`)
- New OpenGL tokens (e.g. `GL_SPIR_V_BINARY_ARB`)
- New OpenGL functions (e.g. `glSpecializeShaderARB`)

Note the `ARB` affix, which stands for Architecture Review Board and is used for official extensions. The extension above was created by the ARB, but there are many different affixes, like `NV` for Nvidia and `AMD` for, well, AMD. Any group may also use the generic `EXT` affix. Lists of extensions, together with their specifications, can be found at the `OpenGL Registry` and `OpenGL ES Registry`.

6.4.1 Loading extension with a loader library

An extension loader library is the easiest and best way to access both OpenGL and OpenGL ES extensions and modern versions of the core OpenGL or OpenGL ES APIs. They will take care of all the details of declaring and loading everything you need. One such library is `glad` and there are several others.

The following example will use glad but all extension loader libraries work similarly.

First you need to generate the source files using the glad Python script. This example generates a loader for any version of OpenGL, which is the default for both GLFW and glad, but loaders for OpenGL ES, as well as loaders for specific API versions and extension sets can be generated. The generated files are written to the `output` directory.

```
python main.py --generator c --no-loader --out-path output
```

The `--no-loader` option is added because GLFW already provides a function for loading OpenGL and OpenGL ES function pointers, one that automatically uses the selected context creation API, and glad can call this instead of having to implement its own. There are several other command-line options as well. See the glad documentation for details.

Add the generated `output/src/glad.c`, `output/include/glad/glad.h` and `output/include/KHR/khrplatform.h` files to your build. Then you need to include the glad header file, which will replace the OpenGL header of your development environment. By including the glad header before the GLFW header, it suppresses the development environment's OpenGL or OpenGL ES header.

```
#include <glad/glad.h>
#include <GLFW/glfw3.h>
```

Finally you need to initialize glad once you have a suitable current context.

```
window = glfwCreateWindow(640, 480, "My Window", NULL, NULL);
if (!window)
{
    ...
```

```
}
glfwMakeContextCurrent(window);
gladLoadGLLoader((GLADloadproc) glfwGetProcAddress);
```

Once glad has been loaded, you have access to all OpenGL core and extension functions supported by both the context you created and the glad loader you generated and you are ready to start rendering.

You can specify a minimum required OpenGL or OpenGL ES version with context hints. If your needs are more complex, you can check the actual OpenGL or OpenGL ES version with context attributes, or you can check whether a specific version is supported by the current context with the `GLAD_GL_VERSION_x_x` booleans.

```
if (GLAD_GL_VERSION_3_2)
{
    // Call OpenGL 3.2+ specific code
}
```

To check whether a specific extension is supported, use the `GLAD_GL_xxx` booleans.

```
if (GLAD_GL_ARB_gl_spirv)
{
    // Use GL_ARB_gl_spirv
}
```

6.4.2 Loading extensions manually

Do not use this technique unless it is absolutely necessary. An extension loader library will save you a ton of tedious, repetitive, error prone work.

To use a certain extension, you must first check whether the context supports that extension and then, if it introduces new functions, retrieve the pointers to those functions. GLFW provides glfwExtensionSupported and glfwGetProcAddress for manual loading of extensions and new API functions.

This section will demonstrate manual loading of OpenGL extensions. The loading of OpenGL ES extensions is identical except for the name of the extension header.

6.4.2.1 The glext.h header

The `glext.h` extension header is a continually updated file that defines the interfaces for all OpenGL extensions. The latest version of this can always be found at the `OpenGL Registry`. There are also extension headers for the various versions of OpenGL ES at the `OpenGL ES Registry`. It it strongly recommended that you use your own copy of the extension header, as the one included in your development environment may be several years out of date and may not include the extensions you wish to use.

The header defines function pointer types for all functions of all extensions it supports. These have names like `PFNGLSPECIALIZESHADERARBPROC` (for `glSpecializeShaderARB`), i.e. the name is made uppercase and `PFN` (pointer to function) and `PROC` (procedure) are added to the ends.

To include the extension header, define GLFW_INCLUDE_GLEXT before including the GLFW header.

```
#define GLFW_INCLUDE_GLEXT
#include <GLFW/glfw3.h>
```

6.4.2.2 Checking for extensions

A given machine may not actually support the extension (it may have older drivers or a graphics card that lacks the necessary hardware features), so it is necessary to check at run-time whether the context supports the extension. This is done with glfwExtensionSupported.

```
if (glfwExtensionSupported("GL_ARB_gl_spirv"))
{
    // The extension is supported by the current context
}
```

The argument is a null terminated ASCII string with the extension name. If the extension is supported, glfwExtensionSupported returns `GLFW_TRUE`, otherwise it returns `GLFW_FALSE`.

6.4.2.3 Fetching function pointers

Many extensions, though not all, require the use of new OpenGL functions. These functions often do not have entry points in the client API libraries of your operating system, making it necessary to fetch them at run time. You can retrieve pointers to these functions with glfwGetProcAddress.

```
PFNGLSPECIALIZESHADERARBPROC pfnSpecializeShaderARB = glfwGetProcAddress("glSpecializeShaderARB");
```

In general, you should avoid giving the function pointer variables the (exact) same name as the function, as this may confuse your linker. Instead, you can use a different prefix, like above, or some other naming scheme.

Now that all the pieces have been introduced, here is what they might look like when used together.

```
#define GLFW_INCLUDE_GLEXT
#include <GLFW/glfw3.h>
#define glSpecializeShaderARB pfnSpecializeShaderARB
PFNGLSPECIALIZESHADERARBPROC pfnSpecializeShaderARB;
// Flag indicating whether the extension is supported
int has_ARB_gl_spirv = 0;
void load_extensions(void)
{
    if (glfwExtensionSupported("GL_ARB_gl_spirv"))
    {
        pfnSpecializeShaderARB = (PFNGLSPECIALIZESHADERARBPROC)
            glfwGetProcAddress("glSpecializeShaderARB");
        has_ARB_gl_spirv = 1;
    }
}
void some_function(void)
{
    if (has_ARB_gl_spirv)
    {
        // Now the extension function can be called as usual
        glSpecializeShaderARB(...);
    }
}
```

Chapter 7

Contribution Guide

7.1 Contents

7.2 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 `Libera.Chat`.

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.

7.3 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:.

7.3.1 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.

7.3.1.1 Quick template

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

7.3.2 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.

7.3.2.1 Quick template

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

7.3.3 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.

7.3.3.1 Quick template

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

7.3.4 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.

7.3.4.1 Quick template

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

7.3.5 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.

7.3.5.1 Quick template

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

7.3.6 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.

7.3.6.1 Quick template

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

7.3.7 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.

7.3.8 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`.

7.4 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:.

7.5 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`.

7.6 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.

Chapter 8

Input guide

This guide introduces the input related functions of GLFW. For details on a specific function in this category, see the Input reference. There are also guides for the other areas of GLFW.

- Introduction to the API
- Window guide
- Context guide
- Vulkan guide
- Monitor guide

GLFW provides many kinds of input. While some can only be polled, like time, or only received via callbacks, like scrolling, many provide both callbacks and polling. Callbacks are more work to use than polling but is less CPU intensive and guarantees that you do not miss state changes.

All input callbacks receive a window handle. By using the window user pointer, you can access non-global structures or objects from your callbacks.

To get a better feel for how the various events callbacks behave, run the `events` test program. It register every callback supported by GLFW and prints out all arguments provided for every event, along with time and sequence information.

8.1 Event processing

GLFW needs to poll the window system for events both to provide input to the application and to prove to the window system that the application hasn't locked up. Event processing is normally done each frame after buffer swapping. Even when you have no windows, event polling needs to be done in order to receive monitor and joystick connection events.

There are three functions for processing pending events. glfwPollEvents, processes only those events that have already been received and then returns immediately.

```
glfwPollEvents();
```

This is the best choice when rendering continuously, like most games do.

If you only need to update the contents of the window when you receive new input, glfwWaitEvents is a better choice.

```
glfwWaitEvents();
```

It puts the thread to sleep until at least one event has been received and then processes all received events. This saves a great deal of CPU cycles and is useful for, for example, editing tools.

If you want to wait for events but have UI elements or other tasks that need periodic updates, glfwWaitEventsTimeout lets you specify a timeout.

```
glfwWaitEventsTimeout(0.7);
```

It puts the thread to sleep until at least one event has been received, or until the specified number of seconds have elapsed. It then processes any received events.

If the main thread is sleeping in glfwWaitEvents, you can wake it from another thread by posting an empty event to the event queue with glfwPostEmptyEvent.

```
glfwPostEmptyEvent();
```

Do not assume that callbacks will *only* be called in response to the above functions. While it is necessary to process events in one or more of the ways above, window systems that require GLFW to register callbacks of its own can pass events to GLFW in response to many window system function calls. GLFW will pass those events on to the application callbacks before returning.

For example, on Windows the system function that glfwSetWindowSize is implemented with will send window size events directly to the event callback that every window has and that GLFW implements for its windows. If you have set a window size callback GLFW will call it in turn with the new size before everything returns back out of the glfwSetWindowSize call.

8.2 Keyboard input

GLFW divides keyboard input into two categories; key events and character events. Key events relate to actual physical keyboard keys, whereas character events relate to the Unicode code points generated by pressing some of them.

Keys and characters do not map 1:1. A single key press may produce several characters, and a single character may require several keys to produce. This may not be the case on your machine, but your users are likely not all using the same keyboard layout, input method or even operating system as you.

8.2.1 Key input

If you wish to be notified when a physical key is pressed or released or when it repeats, set a key callback.

```
glfwSetKeyCallback(window, key_callback);
```

The callback function receives the keyboard key, platform-specific scancode, key action and modifier bits.

```
void key_callback(GLFWwindow* window, int key, int scancode, int action, int mods)
{
    if (key == GLFW_KEY_E && action == GLFW_PRESS)
        activate_airship();
}
```

The action is one of `GLFW_PRESS`, `GLFW_REPEAT` or `GLFW_RELEASE`. The key will be `GLFW_KEY_UNKNOWN` if GLFW lacks a key token for it, for example *E-mail* and *Play* keys.

The scancode is unique for every key, regardless of whether it has a key token. Scancodes are platform-specific but consistent over time, so keys will have different scancodes depending on the platform but they are safe to save to disk. You can query the scancode for any named key on the current platform with glfwGetKeyScancode.

```
const int scancode = glfwGetKeyScancode(GLFW_KEY_X);
set_key_mapping(scancode, swap_weapons);
```

The last reported state for every named key is also saved in per-window state arrays that can be polled with glfwGetKey.

```
int state = glfwGetKey(window, GLFW_KEY_E);
```

```
if (state == GLFW_PRESS)
{
    activate_airship();
}
```

The returned state is one of GLFW_PRESS or GLFW_RELEASE.

This function only returns cached key event state. It does not poll the system for the current physical state of the key.

Whenever you poll state, you risk missing the state change you are looking for. If a pressed key is released again before you poll its state, you will have missed the key press. The recommended solution for this is to use a key callback, but there is also the GLFW_STICKY_KEYS input mode.
```
glfwSetInputMode(window, GLFW_STICKY_KEYS, GLFW_TRUE);
```

When sticky keys mode is enabled, the pollable state of a key will remain GLFW_PRESS until the state of that key is polled with glfwGetKey. Once it has been polled, if a key release event had been processed in the meantime, the state will reset to GLFW_RELEASE, otherwise it will remain GLFW_PRESS.

If you wish to know what the state of the Caps Lock and Num Lock keys was when input events were generated, set the GLFW_LOCK_KEY_MODS input mode.
```
glfwSetInputMode(window, GLFW_LOCK_KEY_MODS, GLFW_TRUE);
```

When this input mode is enabled, any callback that receives modifier bits will have the GLFW_MOD_CAPS_LOCK bit set if Caps Lock was on when the event occurred and the GLFW_MOD_NUM_LOCK bit set if Num Lock was on.

The GLFW_KEY_LAST constant holds the highest value of any named key.

8.2.2 Text input

GLFW supports text input in the form of a stream of Unicode code points, as produced by the operating system text input system. Unlike key input, text input obeys keyboard layouts and modifier keys and supports composing characters using dead keys. Once received, you can encode the code points into UTF-8 or any other encoding you prefer.

Because an unsigned int is 32 bits long on all platforms supported by GLFW, you can treat the code point argument as native endian UTF-32.

If you wish to offer regular text input, set a character callback.
```
glfwSetCharCallback(window, character_callback);
```

The callback function receives Unicode code points for key events that would have led to regular text input and generally behaves as a standard text field on that platform.
```
void character_callback(GLFWwindow* window, unsigned int codepoint)
{
}
```

8.2.3 Key names

If you wish to refer to keys by name, you can query the keyboard layout dependent name of printable keys with glfwGetKeyName.
```
const char* key_name = glfwGetKeyName(GLFW_KEY_W, 0);
show_tutorial_hint("Press %s to move forward", key_name);
```

This function can handle both keys and scancodes. If the specified key is GLFW_KEY_UNKNOWN then the scancode is used, otherwise it is ignored. This matches the behavior of the key callback, meaning the callback arguments can always be passed unmodified to this function.

8.3 Mouse input

Mouse input comes in many forms, including mouse motion, button presses and scrolling offsets. The cursor appearance can also be changed, either to a custom image or a standard cursor shape from the system theme.

8.3.1 Cursor position

If you wish to be notified when the cursor moves over the window, set a cursor position callback.
```
glfwSetCursorPosCallback(window, cursor_position_callback);
```

The callback functions receives the cursor position, measured in screen coordinates but relative to the top-left corner of the window content area. On platforms that provide it, the full sub-pixel cursor position is passed on.
```
static void cursor_position_callback(GLFWwindow* window, double xpos, double ypos)
{
}
```

The cursor position is also saved per-window and can be polled with glfwGetCursorPos.
```
double xpos, ypos;
glfwGetCursorPos(window, &xpos, &ypos);
```

8.3.2 Cursor mode

The GLFW_CURSOR input mode provides several cursor modes for special forms of mouse motion input. By default, the cursor mode is GLFW_CURSOR_NORMAL, meaning the regular arrow cursor (or another cursor set with glfwSetCursor) is used and cursor motion is not limited.

If you wish to implement mouse motion based camera controls or other input schemes that require unlimited mouse movement, set the cursor mode to GLFW_CURSOR_DISABLED.
```
glfwSetInputMode(window, GLFW_CURSOR, GLFW_CURSOR_DISABLED);
```

This will hide the cursor and lock it to the specified window. GLFW will then take care of all the details of cursor re-centering and offset calculation and providing the application with a virtual cursor position. This virtual position is provided normally via both the cursor position callback and through polling.

Note

> You should not implement your own version of this functionality using other features of GLFW. It is not supported and will not work as robustly as GLFW_CURSOR_DISABLED.

If you only wish the cursor to become hidden when it is over a window but still want it to behave normally, set the cursor mode to GLFW_CURSOR_HIDDEN.
```
glfwSetInputMode(window, GLFW_CURSOR, GLFW_CURSOR_HIDDEN);
```

This mode puts no limit on the motion of the cursor.

To exit out of either of these special modes, restore the GLFW_CURSOR_NORMAL cursor mode.
```
glfwSetInputMode(window, GLFW_CURSOR, GLFW_CURSOR_NORMAL);
```

8.3.3 Raw mouse motion

When the cursor is disabled, raw (unscaled and unaccelerated) mouse motion can be enabled if available.

Raw mouse motion is closer to the actual motion of the mouse across a surface. It is not affected by the scaling and acceleration applied to the motion of the desktop cursor. That processing is suitable for a cursor while raw motion is better for controlling for example a 3D camera. Because of this, raw mouse motion is only provided when the cursor is disabled.

Call glfwRawMouseMotionSupported to check if the current machine provides raw motion and set the GLFW_↩ RAW_MOUSE_MOTION input mode to enable it. It is disabled by default.

```
if (glfwRawMouseMotionSupported())
    glfwSetInputMode(window, GLFW_RAW_MOUSE_MOTION, GLFW_TRUE);
```

If supported, raw mouse motion can be enabled or disabled per-window and at any time but it will only be provided when the cursor is disabled.

8.3.4 Cursor objects

GLFW supports creating both custom and system theme cursor images, encapsulated as GLFWcursor objects. They are created with glfwCreateCursor or glfwCreateStandardCursor and destroyed with glfwDestroyCursor, or glfwTerminate, if any remain.

8.3.4.1 Custom cursor creation

A custom cursor is created with glfwCreateCursor, which returns a handle to the created cursor object. For example, this creates a 16x16 white square cursor with the hot-spot in the upper-left corner:

```
unsigned char pixels[16 * 16 * 4];
memset(pixels, 0xff, sizeof(pixels));
GLFWimage image;
image.width = 16;
image.height = 16;
image.pixels = pixels;
GLFWcursor* cursor = glfwCreateCursor(&image, 0, 0);
```

If cursor creation fails, NULL will be returned, so it is necessary to check the return value.

The image data is 32-bit, little-endian, non-premultiplied RGBA, i.e. eight bits per channel with the red channel first. The pixels are arranged canonically as sequential rows, starting from the top-left corner.

8.3.4.2 Standard cursor creation

A cursor with a standard shape from the current system cursor theme can be can be created with glfwCreateStandardCursor.

```
GLFWcursor* url_cursor = glfwCreateStandardCursor(GLFW_POINTING_HAND_CURSOR);
```

These cursor objects behave in the exact same way as those created with glfwCreateCursor except that the system cursor theme provides the actual image.

A few of these shapes are not available everywhere. If a shape is unavailable, NULL is returned. See glfwCreateStandardCursor for details.

8.3.4.3 Cursor destruction

When a cursor is no longer needed, destroy it with glfwDestroyCursor.
```
glfwDestroyCursor(cursor);
```

Cursor destruction always succeeds. If the cursor is current for any window, that window will revert to the default cursor. This does not affect the cursor mode. All remaining cursors are destroyed when glfwTerminate is called.

8.3.4.4 Cursor setting

A cursor can be set as current for a window with glfwSetCursor.
```
glfwSetCursor(window, cursor);
```

Once set, the cursor image will be used as long as the system cursor is over the content area of the window and the cursor mode is set to GLFW_CURSOR_NORMAL.

A single cursor may be set for any number of windows.

To revert to the default cursor, set the cursor of that window to NULL.
```
glfwSetCursor(window, NULL);
```

When a cursor is destroyed, any window that has it set will revert to the default cursor. This does not affect the cursor mode.

8.3.5 Cursor enter/leave events

If you wish to be notified when the cursor enters or leaves the content area of a window, set a cursor enter/leave callback.
```
glfwSetCursorEnterCallback(window, cursor_enter_callback);
```

The callback function receives the new classification of the cursor.
```
void cursor_enter_callback(GLFWwindow* window, int entered)
{
    if (entered)
    {
        // The cursor entered the content area of the window
    }
    else
    {
        // The cursor left the content area of the window
    }
}
```

You can query whether the cursor is currently inside the content area of the window with the GLFW_HOVERED window attribute.
```
if (glfwGetWindowAttrib(window, GLFW_HOVERED))
{
    highlight_interface();
}
```

8.3.6 Mouse button input

If you wish to be notified when a mouse button is pressed or released, set a mouse button callback.

```
glfwSetMouseButtonCallback(window, mouse_button_callback);
```

The callback function receives the mouse button, button action and modifier bits.

```
void mouse_button_callback(GLFWwindow* window, int button, int action, int mods)
{
    if (button == GLFW_MOUSE_BUTTON_RIGHT && action == GLFW_PRESS)
        popup_menu();
}
```

The action is one of `GLFW_PRESS` or `GLFW_RELEASE`.

Mouse button states for named buttons are also saved in per-window state arrays that can be polled with glfwGetMouseButton.

```
int state = glfwGetMouseButton(window, GLFW_MOUSE_BUTTON_LEFT);
if (state == GLFW_PRESS)
{
    upgrade_cow();
}
```

The returned state is one of `GLFW_PRESS` or `GLFW_RELEASE`.

This function only returns cached mouse button event state. It does not poll the system for the current state of the mouse button.

Whenever you poll state, you risk missing the state change you are looking for. If a pressed mouse button is released again before you poll its state, you will have missed the button press. The recommended solution for this is to use a mouse button callback, but there is also the `GLFW_STICKY_MOUSE_BUTTONS` input mode.

```
glfwSetInputMode(window, GLFW_STICKY_MOUSE_BUTTONS, GLFW_TRUE);
```

When sticky mouse buttons mode is enabled, the pollable state of a mouse button will remain `GLFW_PRESS` until the state of that button is polled with glfwGetMouseButton. Once it has been polled, if a mouse button release event had been processed in the meantime, the state will reset to `GLFW_RELEASE`, otherwise it will remain `GLFW_PRESS`.

The `GLFW_MOUSE_BUTTON_LAST` constant holds the highest value of any named button.

8.3.7 Scroll input

If you wish to be notified when the user scrolls, whether with a mouse wheel or touchpad gesture, set a scroll callback.

```
glfwSetScrollCallback(window, scroll_callback);
```

The callback function receives two-dimensional scroll offsets.

```
void scroll_callback(GLFWwindow* window, double xoffset, double yoffset)
{
}
```

A normal mouse wheel, being vertical, provides offsets along the Y-axis.

8.4 Joystick input

The joystick functions expose connected joysticks and controllers, with both referred to as joysticks. It supports up to sixteen joysticks, ranging from GLFW_JOYSTICK_1, GLFW_JOYSTICK_2 up to and including GLFW_JOYSTICK_16 or GLFW_JOYSTICK_LAST. You can test whether a joystick is present with glfwJoystickPresent.

```
int present = glfwJoystickPresent(GLFW_JOYSTICK_1);
```

Each joystick has zero or more axes, zero or more buttons, zero or more hats, a human-readable name, a user pointer and an SDL compatible GUID.

Detected joysticks are added to the beginning of the array. Once a joystick is detected, it keeps its assigned ID until it is disconnected or the library is terminated, so as joysticks are connected and disconnected, there may appear gaps in the IDs.

Joystick axis, button and hat state is updated when polled and does not require a window to be created or events to be processed. However, if you want joystick connection and disconnection events reliably delivered to the joystick callback then you must process events.

To see all the properties of all connected joysticks in real-time, run the joysticks test program.

8.4.1 Joystick axis states

The positions of all axes of a joystick are returned by glfwGetJoystickAxes. See the reference documentation for the lifetime of the returned array.

```
int count;
const float* axes = glfwGetJoystickAxes(GLFW_JOYSTICK_5, &count);
```

Each element in the returned array is a value between -1.0 and 1.0.

8.4.2 Joystick button states

The states of all buttons of a joystick are returned by glfwGetJoystickButtons. See the reference documentation for the lifetime of the returned array.

```
int count;
const unsigned char* buttons = glfwGetJoystickButtons(GLFW_JOYSTICK_3, &count);
```

Each element in the returned array is either GLFW_PRESS or GLFW_RELEASE.

For backward compatibility with earlier versions that did not have glfwGetJoystickHats, the button array by default also includes all hats. See the reference documentation for glfwGetJoystickButtons for details.

8.4.3 Joystick hat states

The states of all hats are returned by glfwGetJoystickHats. See the reference documentation for the lifetime of the returned array.

```
int count;
const unsigned char* hats = glfwGetJoystickHats(GLFW_JOYSTICK_7, &count);
```

Each element in the returned array is one of the following:

Name	Value
GLFW_HAT_CENTERED	0
GLFW_HAT_UP	1
GLFW_HAT_RIGHT	2
GLFW_HAT_DOWN	4
GLFW_HAT_LEFT	8
GLFW_HAT_RIGHT_UP	GLFW_HAT_RIGHT | GLFW_HAT_UP
GLFW_HAT_RIGHT_DOWN	GLFW_HAT_RIGHT | GLFW_HAT_DOWN

The diagonal directions are bitwise combinations of the primary (up, right, down and left) directions and you can test for these individually by ANDing it with the corresponding direction.

```
if (hats[2] & GLFW_HAT_RIGHT)
{
    // State of hat 2 could be right-up, right or right-down
}
```

For backward compatibility with earlier versions that did not have glfwGetJoystickHats, all hats are by default also included in the button array. See the reference documentation for glfwGetJoystickButtons for details.

8.4.4 Joystick name

The human-readable, UTF-8 encoded name of a joystick is returned by glfwGetJoystickName. See the reference documentation for the lifetime of the returned string.

```
const char* name = glfwGetJoystickName(GLFW_JOYSTICK_4);
```

Joystick names are not guaranteed to be unique. Two joysticks of the same model and make may have the same name. Only the joystick ID is guaranteed to be unique, and only until that joystick is disconnected.

8.4.5 Joystick user pointer

Each joystick has a user pointer that can be set with glfwSetJoystickUserPointer and queried with glfwGetJoystickUserPointer. This can be used for any purpose you need and will not be modified by GLFW. The value will be kept until the joystick is disconnected or until the library is terminated.

The initial value of the pointer is NULL.

8.4.6 Joystick configuration changes

If you wish to be notified when a joystick is connected or disconnected, set a joystick callback.

```
glfwSetJoystickCallback(joystick_callback);
```

The callback function receives the ID of the joystick that has been connected and disconnected and the event that occurred.

```
void joystick_callback(int jid, int event)
{
    if (event == GLFW_CONNECTED)
    {
        // The joystick was connected
    }
    else if (event == GLFW_DISCONNECTED)
    {
        // The joystick was disconnected
    }
}
```

For joystick connection and disconnection events to be delivered on all platforms, you need to call one of the event processing functions. Joystick disconnection may also be detected and the callback called by joystick functions. The function will then return whatever it returns for a disconnected joystick.

Only glfwGetJoystickName and glfwGetJoystickUserPointer will return useful values for a disconnected joystick and only before the monitor callback returns.

8.4.7 Gamepad input

The joystick functions provide unlabeled axes, buttons and hats, with no indication of where they are located on the device. Their order may also vary between platforms even with the same device.

To solve this problem the SDL community crowdsourced the SDL_GameControllerDB project, a database of mappings from many different devices to an Xbox-like gamepad.

GLFW supports this mapping format and contains a copy of the mappings available at the time of release. See Gamepad mappings for how to update this at runtime. Mappings will be assigned to joysticks automatically any time a joystick is connected or the mappings are updated.

You can check whether a joystick is both present and has a gamepad mapping with glfwJoystickIsGamepad.

```
if (glfwJoystickIsGamepad(GLFW_JOYSTICK_2))
{
    // Use as gamepad
}
```

If you are only interested in gamepad input you can use this function instead of glfwJoystickPresent.

You can query the human-readable name provided by the gamepad mapping with glfwGetGamepadName. This may or may not be the same as the joystick name.

```
const char* name = glfwGetGamepadName(GLFW_JOYSTICK_7);
```

To retrieve the gamepad state of a joystick, call glfwGetGamepadState.

```
GLFWgamepadstate state;
if (glfwGetGamepadState(GLFW_JOYSTICK_3, &state))
{
    if (state.buttons[GLFW_GAMEPAD_BUTTON_A])
    {
        input_jump();
    }
    input_speed(state.axes[GLFW_GAMEPAD_AXIS_RIGHT_TRIGGER]);
}
```

The GLFWgamepadstate struct has two arrays; one for button states and one for axis states. The values for each button and axis are the same as for the glfwGetJoystickButtons and glfwGetJoystickAxes functions, i.e. GLFW_PRESS or GLFW_RELEASE for buttons and -1.0 to 1.0 inclusive for axes.

The sizes of the arrays and the positions within each array are fixed.

The button indices are GLFW_GAMEPAD_BUTTON_A, GLFW_GAMEPAD_BUTTON_B, GLFW_GAMEPAD_BUTTON_X, GLFW_GAMEPAD_BUTTON_Y, GLFW_GAMEPAD_BUTTON_LEFT_BUMPER, GLFW_GAMEPAD_BUTTON_RIGHT_BUMPER, GLFW_GAMEPAD_BUTTON_BACK, GLFW_GAMEPAD_BUTTON_START, GLFW_GAMEPAD_BUTTON_GUIDE, GLFW_GAMEPAD_BUTTON_LEFT_THUMB, GLFW_GAMEPAD_BUTTON_RIGHT_THUMB, GLFW_GAMEPAD_BUTTON_DPAD_UP, GLFW_GAMEPAD_BUTTON_DPAD_RIGHT, GLFW_GAMEPAD_BUTTON_DPAD_DOWN and GLFW_GAMEPAD_BUTTON_DPAD_LEFT.

For those who prefer, there are also the GLFW_GAMEPAD_BUTTON_CROSS, GLFW_GAMEPAD_BUTTON_CIRCLE, GLFW_GAMEPAD_BUTTON_SQUARE and GLFW_GAMEPAD_BUTTON_TRIANGLE aliases for the A, B, X and Y button indices.

The axis indices are GLFW_GAMEPAD_AXIS_LEFT_X, GLFW_GAMEPAD_AXIS_LEFT_Y, GLFW_GAMEPAD_AXIS_RIGHT_X, GLFW_GAMEPAD_AXIS_RIGHT_Y, GLFW_GAMEPAD_AXIS_LEFT_TRIGGER and GLFW_GAMEPAD_AXIS_RIGHT_TRIGGER.

The GLFW_GAMEPAD_BUTTON_LAST and GLFW_GAMEPAD_AXIS_LAST constants equal the largest available index for each array.

8.4.8 Gamepad mappings

GLFW contains a copy of the mappings available in SDL_GameControllerDB at the time of release. Newer ones can be added at runtime with glfwUpdateGamepadMappings.

```
const char* mappings = load_file_contents("game/data/gamecontrollerdb.txt");
glfwUpdateGamepadMappings(mappings);
```

This function supports everything from single lines up to and including the unmodified contents of the whole `gamecontrollerdb.txt` file.

If you are compiling GLFW from source with CMake you can update the built-in mappings by building the *update_mappings* target. This runs the `GenerateMappings.cmake` CMake script, which downloads `gamecontrollerdb.txt` and regenerates the `mappings.h` header file.

Below is a description of the mapping format. Please keep in mind that **this description is not authoritative**. The format is defined by the SDL and SDL_GameControllerDB projects and their documentation and code takes precedence.

Each mapping is a single line of comma-separated values describing the GUID, name and layout of the gamepad. Lines that do not begin with a hexadecimal digit are ignored.

The first value is always the gamepad GUID, a 32 character long hexadecimal string that typically identifies its make, model, revision and the type of connection to the computer. When this information is not available, the GUID is generated using the gamepad name. GLFW uses the SDL 2.0.5+ GUID format but can convert from the older formats.

The second value is always the human-readable name of the gamepad.

All subsequent values are in the form `<field>:<value>` and describe the layout of the mapping. These fields may not all be present and may occur in any order.

The button fields are `a`, `b`, `x`, `y`, `back`, `start`, `guide`, `dpup`, `dpright`, `dpdown`, `dpleft`, `leftshoulder`, `rightshoulder`, `leftstick` and `rightstick`.

The axis fields are `leftx`, `lefty`, `rightx`, `righty`, `lefttrigger` and `righttrigger`.

The value of an axis or button field can be a joystick button, a joystick axis, a hat bitmask or empty. Joystick buttons are specified as `bN`, for example `b2` for the third button. Joystick axes are specified as `aN`, for example `a7` for the eighth button. Joystick hat bit masks are specified as `hN.N`, for example `h0.8` for left on the first hat. More than one bit may be set in the mask.

Before an axis there may be a + or − range modifier, for example `+a3` for the positive half of the fourth axis. This restricts input to only the positive or negative halves of the joystick axis. After an axis or half-axis there may be the ∼ inversion modifier, for example `a2`∼ or `−a7`∼. This negates the values of the gamepad axis.

The hat bit mask match the hat states in the joystick functions.

There is also the special `platform` field that specifies which platform the mapping is valid for. Possible values are `Windows`, `Mac OS X` and `Linux`.

Below is an example of what a gamepad mapping might look like. It is the one built into GLFW for Xbox controllers accessed via the XInput API on Windows. This example has been broken into several lines to fit on the page, but real gamepad mappings must be a single line.

```
78696e7075740100000000000000000000,XInput Gamepad (GLFW),platform:Windows,a:b0,
b:b1,x:b2,y:b3,leftshoulder:b4,rightshoulder:b5,back:b6,start:b7,leftstick:b8,
rightstick:b9,leftx:a0,lefty:a1,rightx:a2,righty:a3,lefttrigger:a4,
righttrigger:a5,dpup:h0.1,dpright:h0.2,dpdown:h0.4,dpleft:h0.8,
```

Note

GLFW does not yet support the output range and modifiers + and − that were recently added to SDL. The input modifiers +, − and ∼ are supported and described above.

8.5 Time input

GLFW provides high-resolution time input, in seconds, with glfwGetTime.

```
double seconds = glfwGetTime();
```

It returns the number of seconds since the library was initialized with glfwInit. The platform-specific time sources used typically have micro- or nanosecond resolution.

You can modify the base time with glfwSetTime.

```
glfwSetTime(4.0);
```

This sets the time to the specified time, in seconds, and it continues to count from there.

You can also access the raw timer used to implement the functions above, with glfwGetTimerValue.

```
uint64_t value = glfwGetTimerValue();
```

This value is in 1 / frequency seconds. The frequency of the raw timer varies depending on the operating system and hardware. You can query the frequency, in Hz, with glfwGetTimerFrequency.

```
uint64_t frequency = glfwGetTimerFrequency();
```

8.6 Clipboard input and output

If the system clipboard contains a UTF-8 encoded string or if it can be converted to one, you can retrieve it with glfwGetClipboardString. See the reference documentation for the lifetime of the returned string.

```
const char* text = glfwGetClipboardString(NULL);
if (text)
{
    insert_text(text);
}
```

If the clipboard is empty or if its contents could not be converted, `NULL` is returned.

The contents of the system clipboard can be set to a UTF-8 encoded string with glfwSetClipboardString.

```
glfwSetClipboardString(NULL, "A string with words in it");
```

8.7 Path drop input

If you wish to receive the paths of files and/or directories dropped on a window, set a file drop callback.

```
glfwSetDropCallback(window, drop_callback);
```

The callback function receives an array of paths encoded as UTF-8.

```
void drop_callback(GLFWwindow* window, int count, const char** paths)
{
    int i;
    for (i = 0;  i < count;  i++)
        handle_dropped_file(paths[i]);
}
```

The path array and its strings are only valid until the file drop callback returns, as they may have been generated specifically for that event. You need to make a deep copy of the array if you want to keep the paths.

Chapter 9

Internal structure

There are several interfaces inside GLFW. Each interface has its own area of responsibility and its own naming conventions.

9.1 Public interface

The most well-known is the public interface, described in the glfw3.h header file. This is implemented in source files shared by all platforms and these files contain no platform-specific code. This code usually ends up calling the platform and internal interfaces to do the actual work.

The public interface uses the OpenGL naming conventions except with GLFW and glfw instead of GL and gl. For struct members, where OpenGL sets no precedent, it use headless camel case.

Examples: `glfwCreateWindow`, `GLFWwindow`, `GLFW_RED_BITS`

9.2 Native interface

The native interface is a small set of publicly available but platform-specific functions, described in the glfw3native.h header file and used to gain access to the underlying window, context and (on some platforms) display handles used by the platform interface.

The function names of the native interface are similar to those of the public interface, but embeds the name of the interface that the returned handle is from.

Examples: `glfwGetX11Window`, `glfwGetWGLContext`

9.3 Internal interface

The internal interface consists of utility functions used by all other interfaces. It is shared code implemented in the same shared source files as the public and event interfaces. The internal interface is described in the internal.h header file.

The internal interface is in charge of GLFW's global data, which it stores in a `_GLFWlibrary` struct named `_glfw`.

The internal interface uses the same style as the public interface, except all global names have a leading underscore.

Examples: `_glfwIsValidContextConfig`, `_GLFWwindow`, `_glfw.monitorCount`

9.4 Platform interface

The platform interface implements all platform-specific operations as a service to the public interface. This includes event processing. The platform interface is never directly called by application code and never directly calls application-provided callbacks. It is also prohibited from modifying the platform-independent part of the internal structs. Instead, it calls the event interface when events interesting to GLFW are received.

The platform interface mostly mirrors those parts of the public interface that needs to perform platform-specific operations on some or all platforms.

The window system bits of the platform API is called through the `_GLFWplatform` struct of function pointers, to allow runtime selection of platform. This includes the window and context creation, input and event processing, monitor and Vulkan surface creation parts of GLFW. This is located in the global `_glfw` struct.

Examples: `_glfw.platform.createWindow`

The timer, threading and module loading bits of the platform API are plain functions with a `_glfwPlatform` prefix, as these things are independent of what window system is being used.

Examples: `_glfwPlatformGetTimerValue`

The platform interface also defines structs that contain platform-specific global and per-object state. Their names mirror those of the internal interface, except that an interface-specific suffix is added.

Examples: `_GLFWwindowX11`, `_GLFWcontextWGL`

These structs are incorporated as members into the internal interface structs using special macros that name them after the specific interface used. This prevents shared code from accidentally using these members.

Examples: `window->win32.handle`, `_glfw.x11.display`

9.5 Event interface

The event interface is implemented in the same shared source files as the public interface and is responsible for delivering the events it receives to the application, either via callbacks, via window state changes or both.

The function names of the event interface use a `_glfwInput` prefix and the ObjectEvent pattern.

Examples: `_glfwInputWindowFocus`, `_glfwInputCursorPos`

9.6 Static functions

Static functions may be used by any interface and have no prefixes or suffixes. These use headless camel case.

Examples: `isValidElementForJoystick`

9.7 Configuration macros

GLFW uses a number of configuration macros to select at compile time which interfaces and code paths to use. They are defined in the GLFW CMake target.

Configuration macros the same style as tokens in the public interface, except with a leading underscore.

Examples: `_GLFW_WIN32`, `_GLFW_BUILD_DLL`

Chapter 10

Introduction to the API

This guide introduces the basic concepts of GLFW and describes initialization, error handling and API guarantees and limitations. For a broad but shallow tutorial, see Getting started instead. For details on a specific function in this category, see the Initialization, version and error reference.

There are also guides for the other areas of GLFW.

- Window guide
- Context guide
- Vulkan guide
- Monitor guide
- Input guide

10.1 Initialization and termination

Before most GLFW functions may be called, the library must be initialized. This initialization checks what features are available on the machine, enumerates monitors, initializes the timer and performs any required platform-specific initialization.

Only the following functions may be called before the library has been successfully initialized, and only from the main thread.

- glfwGetVersion
- glfwGetVersionString
- glfwPlatformSupported
- glfwGetError
- glfwSetErrorCallback
- glfwInitHint
- glfwInitAllocator
- glfwInitVulkanLoader
- glfwInit
- glfwTerminate

Calling any other function before successful initialization will cause a GLFW_NOT_INITIALIZED error.

10.1.1 Initializing GLFW

The library is initialized with glfwInit, which returns GLFW_FALSE if an error occurred.

```
if (!glfwInit())
{
    // Handle initialization failure
}
```

If any part of initialization fails, any parts that succeeded are terminated as if glfwTerminate had been called. The library only needs to be initialized once and additional calls to an already initialized library will return GLFW_TRUE immediately.

Once the library has been successfully initialized, it should be terminated before the application exits. Modern systems are very good at freeing resources allocated by programs that exit, but GLFW sometimes has to change global system settings and these might not be restored without termination.

@macos When the library is initialized the main menu and dock icon are created. These are not desirable for a command-line only program. The creation of the main menu and dock icon can be disabled with the GLFW_COCOA_MENUBAR init hint.

10.1.2 Initialization hints

Initialization hints are set before glfwInit and affect how the library behaves until termination. Hints are set with glfwInitHint.

```
glfwInitHint(GLFW_JOYSTICK_HAT_BUTTONS, GLFW_FALSE);
```

The values you set hints to are never reset by GLFW, but they only take effect during initialization. Once GLFW has been initialized, any values you set will be ignored until the library is terminated and initialized again.

Some hints are platform specific. These may be set on any platform but they will only affect their specific platform. Other platforms will ignore them. Setting these hints requires no platform specific headers or functions.

10.1.2.1 Shared init hints

GLFW_PLATFORM specifies the platform to use for windowing and input. Possible values are GLFW_ANY_PLATFORM, GLFW_PLATFORM_WIN32, GLFW_PLATFORM_COCOA, GLFW_PLATFORM_X11, GLFW_PLATFORM_WAYLAND and GLFW_PLATFORM_NULL. The default value is GLFW_ANY_PLATFORM, which will choose any platform the library includes support for except for the Null backend.

GLFW_JOYSTICK_HAT_BUTTONS specifies whether to also expose joystick hats as buttons, for compatibility with earlier versions of GLFW that did not have glfwGetJoystickHats. Possible values are GLFW_TRUE and GLFW_FALSE.

GLFW_ANGLE_PLATFORM_TYPE specifies the platform type (rendering backend) to request when using Open GL ES and EGL via ANGLE. If the requested platform type is unavailable, ANGLE will use its default. Possible values are one of GLFW_ANGLE_PLATFORM_TYPE_NONE, GLFW_ANGLE_PLATFORM_TYPE_OPENGL, GLFW_ANGLE_PLATFORM_TYPE_OPENGLES, GLFW_ANGLE_PLATFORM_TYPE_D3D9, GLFW_ANGLE_PLATFORM_TYPE_D3D11, GLFW_ANGLE_PLATFORM_TYPE_VULKAN and GLFW_ANGLE_PLATFORM_TYPE_METAL.

The ANGLE platform type is specified via the EGL_ANGLE_platform_angle extension. This extension is not used if this hint is GLFW_ANGLE_PLATFORM_TYPE_NONE, which is the default value.

10.1.2.2 macOS specific init hints

GLFW_COCOA_CHDIR_RESOURCES specifies whether to set the current directory to the application to the `Contents/Resources` subdirectory of the application's bundle, if present. Possible values are `GLFW_TRUE` and `GLFW_FALSE`. This is ignored on other platforms.

GLFW_COCOA_MENUBAR specifies whether to create the menu bar and dock icon when GLFW is initialized. This applies whether the menu bar is created from a nib or manually by GLFW. Possible values are `GLFW_TRUE` and `GLFW_FALSE`. This is ignored on other platforms.

10.1.2.3 X11 specific init hints

GLFW_X11_XCB_VULKAN_SURFACE specifies whether to prefer the `VK_KHR_xcb_surface` extension for creating Vulkan surfaces, or whether to use the `VK_KHR_xlib_surface` extension. Possible values are `GLFW_TRUE` and `GLFW_FALSE`. This is ignored on other platforms.

10.1.2.4 Supported and default values

Initialization hint	Default value	Supported values
GLFW_PLATFORM	GLFW_ANY_PLATFORM	GLFW_ANY_PLATFORM, GLFW_PLATFORM_WIN32, GLFW_PLATFORM_COCOA, GLFW_PLATFORM_X11, GLFW_PLATFORM_WAYLAND or GLFW_PLATFORM_NULL
GLFW_JOYSTICK_HAT_BUTTONS	GLFW_TRUE	GLFW_TRUE or GLFW_FALSE
GLFW_ANGLE_PLATFORM_TYPE	GLFW_ANGLE_PLATFORM_TYPE_NONE	GLFW_ANGLE_PLATFORM_TYPE_NONE, GLFW_ANGLE_PLATFORM_TYPE_OPENGL, GLFW_ANGLE_PLATFORM_TYPE_OPENGLES, GLFW_ANGLE_PLATFORM_TYPE_D3D9, GLFW_ANGLE_PLATFORM_TYPE_D3D11, GLFW_ANGLE_PLATFORM_TYPE_VULKAN or GLFW_ANGLE_PLATFORM_TYPE_METAL
GLFW_COCOA_CHDIR_RESOURCES	GLFW_TRUE	GLFW_TRUE or GLFW_FALSE
GLFW_COCOA_MENUBAR	GLFW_TRUE	GLFW_TRUE or GLFW_FALSE
GLFW_X11_XCB_VULKAN_SURFACE	GLFW_TRUE	GLFW_TRUE or GLFW_FALSE

10.1.3 Runtime platform selection

GLFW can be compiled for more than one platform (window system) at once. This lets a single library binary support both X11 and Wayland on Linux and other Unix-like systems.

You can control platform selection via the GLFW_PLATFORM initialization hint. By default this is set to GLFW_ANY_PLATFORM, which will look for supported window systems in order of priority and select the first one it finds. It can also be set to any specific platform to have GLFW only look for that one.

```
glfwInitHint(GLFW_PLATFORM, GLFW_PLATFORM_X11);
```

This mechanism also provides the Null platform, which is always supported but needs to be explicitly requested. This platform is effectively a stub, emulating a window system on a single 1080p monitor, but will not interact with any actual window system.

```
glfwInitHint(GLFW_PLATFORM, GLFW_PLATFORM_NULL);
```

You can test whether a library binary was compiled with support for a specific platform with glfwPlatformSupported.

```
if (glfwPlatformSupported(GLFW_PLATFORM_WAYLAND))
    glfwInitHint(GLFW_PLATFORM, GLFW_PLATFORM_WAYLAND);
```

Once GLFW has been initialized, you can query which platform was selected with glfwGetPlatform.

```
int platform = glfwGetPlatform();
```

If you are using any native access functions, especially on Linux and other Unix-like systems, then you may need to check that you are calling the ones matching the selected platform.

10.1.4 Custom heap memory allocator

The heap memory allocator can be customized before initialization with glfwInitAllocator.

```
GLFWallocator allocator;
allocator.allocate = my_malloc;
allocator.reallocate = my_realloc;
allocator.deallocate = my_free;
allocator.user = NULL;
glfwInitAllocator(&allocator);
```

The allocator will be picked up at the beginning of initialization and will be used until GLFW has been fully terminated. Any allocator set after initialization will be picked up only at the next initialization.

The allocator will only be used for allocations that would have been made with the C standard library. Memory allocations that must be made with platform specific APIs will still use those.

The allocation function must have a signature matching GLFWallocatefun. It receives the desired size, in bytes, and the user pointer passed to glfwInitAllocator and returns the address to the allocated memory block.

```
void* my_malloc(size_t size, void* user)
{
    ...
}
```

The reallocation function must have a function signature matching GLFWreallocatefun. It receives the memory block to be reallocated, the new desired size, in bytes, and the user pointer passed to glfwInitAllocator and returns the address to the resized memory block.

```
void* my_realloc(void* block, size_t size, void* user)
{
    ...
}
```

The deallocation function must have a function signature matching GLFWdeallocatefun. It receives the memory block to be deallocated and the user pointer passed to glfwInitAllocator.

```
void my_free(void* block, void* user)
{
    ...
}
```

10.1.5 Terminating GLFW

Before your application exits, you should terminate the GLFW library if it has been initialized. This is done with glfwTerminate.

```
glfwTerminate();
```

This will destroy any remaining window, monitor and cursor objects, restore any modified gamma ramps, re-enable the screensaver if it had been disabled and free any other resources allocated by GLFW.

Once the library is terminated, it is as if it had never been initialized and you will need to initialize it again before being able to use GLFW. If the library was not initialized or had already been terminated, it return immediately.

10.2 Error handling

Some GLFW functions have return values that indicate an error, but this is often not very helpful when trying to figure out what happened or why it occurred. Other functions have no return value reserved for errors, so error notification needs a separate channel. Finally, far from all GLFW functions have return values.

The last error code for the calling thread can be queried at any time with glfwGetError.

```
int code = glfwGetError(NULL);
if (code != GLFW_NO_ERROR)
    handle_error(code);
```

If no error has occurred since the last call, GLFW_NO_ERROR (zero) is returned. The error is cleared before the function returns.

The error code indicates the general category of the error. Some error codes, such as GLFW_NOT_INITIALIZED has only a single meaning, whereas others like GLFW_PLATFORM_ERROR are used for many different errors.

GLFW often has more information about an error than its general category. You can retrieve a UTF-8 encoded human-readable description along with the error code. If no error has occurred since the last call, the description is set to NULL.

```
const char* description;
int code = glfwGetError(&description);
if (description)
    display_error_message(code, description);
```

The retrieved description string is only valid until the next error occurs. This means you must make a copy of it if you want to keep it.

You can also set an error callback, which will be called each time an error occurs. It is set with glfwSetErrorCallback.

```
glfwSetErrorCallback(error_callback);
```

The error callback receives the same error code and human-readable description returned by glfwGetError.

```
void error_callback(int code, const char* description)
{
    display_error_message(code, description);
}
```

The error callback is called after the error is stored, so calling glfwGetError from within the error callback returns the same values as the callback argument.

The description string passed to the callback is only valid until the error callback returns. This means you must make a copy of it if you want to keep it.

Reported errors are never fatal. As long as GLFW was successfully initialized, it will remain initialized and in a safe state until terminated regardless of how many errors occur. If an error occurs during initialization that causes glfwInit to fail, any part of the library that was initialized will be safely terminated.

Do not rely on a currently invalid call to generate a specific error, as in the future that same call may generate a different error or become valid.

10.3 Coordinate systems

GLFW has two primary coordinate systems: the *virtual screen* and the window *content area* or *content area*. Both use the same unit: *virtual screen coordinates*, or just *screen coordinates*, which don't necessarily correspond to pixels.

Both the virtual screen and the content area coordinate systems have the X-axis pointing to the right and the Y-axis pointing down.

Window and monitor positions are specified as the position of the upper-left corners of their content areas relative to the virtual screen, while cursor positions are specified relative to a window's content area.

Because the origin of the window's content area coordinate system is also the point from which the window position is specified, you can translate content area coordinates to the virtual screen by adding the window position. The window frame, when present, extends out from the content area but does not affect the window position.

Almost all positions and sizes in GLFW are measured in screen coordinates relative to one of the two origins above. This includes cursor positions, window positions and sizes, window frame sizes, monitor positions and video mode resolutions.

Two exceptions are the monitor physical size, which is measured in millimetres, and framebuffer size, which is measured in pixels.

Pixels and screen coordinates may map 1:1 on your machine, but they won't on every other machine, for example on a Mac with a Retina display. The ratio between screen coordinates and pixels may also change at run-time depending on which monitor the window is currently considered to be on.

10.4 Guarantees and limitations

This section describes the conditions under which GLFW can be expected to function, barring bugs in the operating system or drivers. Use of GLFW outside of these limits may work on some platforms, or on some machines, or some of the time, or on some versions of GLFW, but it may break at any time and this will not be considered a bug.

10.4.1 Pointer lifetimes

GLFW will never free any pointer you provide to it and you must never free any pointer it provides to you.

Many GLFW functions return pointers to dynamically allocated structures, strings or arrays, and some callbacks are provided with strings or arrays. These are always managed by GLFW and should never be freed by the application. The lifetime of these pointers is documented for each GLFW function and callback. If you need to keep this data, you must copy it before its lifetime expires.

Many GLFW functions accept pointers to structures or strings allocated by the application. These are never freed by GLFW and are always the responsibility of the application. If GLFW needs to keep the data in these structures or strings, it is copied before the function returns.

Pointer lifetimes are guaranteed not to be shortened in future minor or patch releases.

10.4.2 Reentrancy

GLFW event processing and object destruction are not reentrant. This means that the following functions must not be called from any callback function:

- glfwDestroyWindow
- glfwDestroyCursor
- glfwPollEvents
- glfwWaitEvents
- glfwWaitEventsTimeout
- glfwTerminate

These functions may be made reentrant in future minor or patch releases, but functions not on this list will not be made non-reentrant.

10.4.3 Thread safety

Most GLFW functions must only be called from the main thread (the thread that calls main), but some may be called from any thread once the library has been initialized. Before initialization the whole library is thread-unsafe.

The reference documentation for every GLFW function states whether it is limited to the main thread.

Initialization, termination, event processing and the creation and destruction of windows, cursors and OpenGL and OpenGL ES contexts are all restricted to the main thread due to limitations of one or several platforms.

Because event processing must be performed on the main thread, all callbacks except for the error callback will only be called on that thread. The error callback may be called on any thread, as any GLFW function may generate errors.

The error code and description may be queried from any thread.

- glfwGetError

Empty events may be posted from any thread.

- glfwPostEmptyEvent

The window user pointer and close flag may be read and written from any thread, but this is not synchronized by GLFW.

- glfwGetWindowUserPointer
- glfwSetWindowUserPointer
- glfwWindowShouldClose
- glfwSetWindowShouldClose

These functions for working with OpenGL and OpenGL ES contexts may be called from any thread, but the window object is not synchronized by GLFW.

- glfwMakeContextCurrent
- glfwGetCurrentContext
- glfwSwapBuffers
- glfwSwapInterval
- glfwExtensionSupported
- glfwGetProcAddress

The raw timer functions may be called from any thread.

- glfwGetTimerFrequency
- glfwGetTimerValue

The regular timer may be used from any thread, but reading and writing the timer offset is not synchronized by GLFW.

- glfwGetTime
- glfwSetTime

Library version information may be queried from any thread.

- glfwGetVersion
- glfwGetVersionString

Platform information may be queried from any thread.

- glfwPlatformSupported
- glfwGetPlatform

All Vulkan related functions may be called from any thread.

- glfwVulkanSupported
- glfwGetRequiredInstanceExtensions
- glfwGetInstanceProcAddress
- glfwGetPhysicalDevicePresentationSupport
- glfwCreateWindowSurface

GLFW uses synchronization objects internally only to manage the per-thread context and error states. Additional synchronization is left to the application.

Functions that may currently be called from any thread will always remain so, but functions that are currently limited to the main thread may be updated to allow calls from any thread in future releases.

10.4.4 Version compatibility

GLFW uses Semantic Versioning. This guarantees source and binary backward compatibility with earlier minor versions of the API. This means that you can drop in a newer version of the library and existing programs will continue to compile and existing binaries will continue to run.

Once a function or constant has been added, the signature of that function or value of that constant will remain unchanged until the next major version of GLFW. No compatibility of any kind is guaranteed between major versions.

Undocumented behavior, i.e. behavior that is not described in the documentation, may change at any time until it is documented.

If the reference documentation and the implementation differ, the reference documentation will almost always take precedence and the implementation will be fixed in the next release. The reference documentation will also take precedence over anything stated in a guide.

10.4.5 Event order

The order of arrival of related events is not guaranteed to be consistent across platforms. The exception is synthetic key and mouse button release events, which are always delivered after the window defocus event.

10.5 Version management

GLFW provides mechanisms for identifying what version of GLFW your application was compiled against as well as what version it is currently running against. If you are loading GLFW dynamically (not just linking dynamically), you can use this to verify that the library binary is compatible with your application.

10.5.1 Compile-time version

The compile-time version of GLFW is provided by the GLFW header with the GLFW_VERSION_MAJOR, GLFW_VERSION_MINOR and GLFW_VERSION_REVISION macros.

```
printf("Compiled against GLFW %i.%i.%i\n",
       GLFW_VERSION_MAJOR,
       GLFW_VERSION_MINOR,
       GLFW_VERSION_REVISION);
```

10.5.2 Run-time version

The run-time version can be retrieved with glfwGetVersion, a function that may be called regardless of whether GLFW is initialized.

```
int major, minor, revision;
glfwGetVersion(&major, &minor, &revision);
printf("Running against GLFW %i.%i.%i\n", major, minor, revision);
```

10.5.3 Version string

GLFW 3 also provides a compile-time generated version string that describes the version, platform, compiler and any platform-specific compile-time options. This is primarily intended for submitting bug reports, to allow developers to see which code paths are enabled in a binary.

The version string is returned by glfwGetVersionString, a function that may be called regardless of whether GLFW is initialized.

Do not use the version string to parse the GLFW library version. The glfwGetVersion function already provides the version of the running library binary.

Do not use the version string to parse what platforms are supported. The glfwPlatformSupported function lets you query platform support.

GLFW 3.4: The format of this string was changed to support the addition of runtime platform selection.

The format of the string is as follows:

- The version of GLFW
- For each supported platform:
  - The name of the window system API
  - The name of the window system specific context creation API, if applicable
- The names of the always supported context creation APIs EGL and OSMesa
- Any additional compile-time options, APIs and (on Windows) what compiler was used

For example, GLFW 3.4 compiled as a DLL for Windows with MinGW may have a version string like this:

```
3.4.0 Win32 WGL Null EGL OSMesa MinGW DLL
```

While GLFW compiled as as static library for Linux with both Wayland and X11 enabled may have a version string like this:

```
3.4.0 Wayland X11 GLX Null EGL OSMesa monotonic
```

Chapter 11

Monitor guide

This guide introduces the monitor related functions of GLFW. For details on a specific function in this category, see the Monitor reference. There are also guides for the other areas of GLFW.

- Introduction to the API
- Window guide
- Context guide
- Vulkan guide
- Input guide

11.1 Monitor objects

A monitor object represents a currently connected monitor and is represented as a pointer to the `opaque` type GLFWmonitor. Monitor objects cannot be created or destroyed by the application and retain their addresses until the monitors they represent are disconnected or until the library is terminated.

Each monitor has a current video mode, a list of supported video modes, a virtual position, a human-readable name, an estimated physical size and a gamma ramp. One of the monitors is the primary monitor.

The virtual position of a monitor is in screen coordinates and, together with the current video mode, describes the viewports that the connected monitors provide into the virtual desktop that spans them.

To see how GLFW views your monitor setup and its available video modes, run the `monitors` test program.

11.1.1 Retrieving monitors

The primary monitor is returned by glfwGetPrimaryMonitor. It is the user's preferred monitor and is usually the one with global UI elements like task bar or menu bar.

```
GLFWmonitor* primary = glfwGetPrimaryMonitor();
```

You can retrieve all currently connected monitors with glfwGetMonitors. See the reference documentation for the lifetime of the returned array.

```
int count;
GLFWmonitor** monitors = glfwGetMonitors(&count);
```

The primary monitor is always the first monitor in the returned array, but other monitors may be moved to a different index when a monitor is connected or disconnected.

11.1.2 Monitor configuration changes

If you wish to be notified when a monitor is connected or disconnected, set a monitor callback.

```
glfwSetMonitorCallback(monitor_callback);
```

The callback function receives the handle for the monitor that has been connected or disconnected and the event that occurred.

```
void monitor_callback(GLFWmonitor* monitor, int event)
{
    if (event == GLFW_CONNECTED)
    {
        // The monitor was connected
    }
    else if (event == GLFW_DISCONNECTED)
    {
        // The monitor was disconnected
    }
}
```

If a monitor is disconnected, all windows that are full screen on it will be switched to windowed mode before the callback is called. Only glfwGetMonitorName and glfwGetMonitorUserPointer will return useful values for a disconnected monitor and only before the monitor callback returns.

11.2 Monitor properties

Each monitor has a current video mode, a list of supported video modes, a virtual position, a content scale, a human-readable name, a user pointer, an estimated physical size and a gamma ramp.

11.2.1 Video modes

GLFW generally does a good job selecting a suitable video mode when you create a full screen window, change its video mode or make a windowed one full screen, but it is sometimes useful to know exactly which video modes are supported.

Video modes are represented as GLFWvidmode structures. You can get an array of the video modes supported by a monitor with glfwGetVideoModes. See the reference documentation for the lifetime of the returned array.

```
int count;
GLFWvidmode* modes = glfwGetVideoModes(monitor, &count);
```

To get the current video mode of a monitor call glfwGetVideoMode. See the reference documentation for the lifetime of the returned pointer.

```
const GLFWvidmode* mode = glfwGetVideoMode(monitor);
```

The resolution of a video mode is specified in screen coordinates, not pixels.

11.2.2 Physical size

The physical size of a monitor in millimetres, or an estimation of it, can be retrieved with glfwGetMonitorPhysicalSize. This has no relation to its current *resolution*, i.e. the width and height of its current video mode.

```
int width_mm, height_mm;
glfwGetMonitorPhysicalSize(monitor, &width_mm, &height_mm);
```

While this can be used to calculate the raw DPI of a monitor, this is often not useful. Instead use the monitor content scale and window content scale to scale your content.

11.2.3 Content scale

The content scale for a monitor can be retrieved with glfwGetMonitorContentScale.

```
float xscale, yscale;
glfwGetMonitorContentScale(monitor, &xscale, &yscale);
```

The content scale is the ratio between the current DPI and the platform's default DPI. This is especially important for text and any UI elements. If the pixel dimensions of your UI scaled by this look appropriate on your machine then it should appear at a reasonable size on other machines regardless of their DPI and scaling settings. This relies on the system DPI and scaling settings being somewhat correct.

The content scale may depend on both the monitor resolution and pixel density and on user settings. It may be very different from the raw DPI calculated from the physical size and current resolution.

11.2.4 Virtual position

The position of the monitor on the virtual desktop, in screen coordinates, can be retrieved with glfwGetMonitorPos.

```
int xpos, ypos;
glfwGetMonitorPos(monitor, &xpos, &ypos);
```

11.2.5 Work area

The area of a monitor not occupied by global task bars or menu bars is the work area. This is specified in screen coordinates and can be retrieved with glfwGetMonitorWorkarea.

```
int xpos, ypos, width, height;
glfwGetMonitorWorkarea(monitor, &xpos, &ypos, &width, &height);
```

11.2.6 Human-readable name

The human-readable, UTF-8 encoded name of a monitor is returned by glfwGetMonitorName. See the reference documentation for the lifetime of the returned string.

```
const char* name = glfwGetMonitorName(monitor);
```

Monitor names are not guaranteed to be unique. Two monitors of the same model and make may have the same name. Only the monitor handle is guaranteed to be unique, and only until that monitor is disconnected.

11.2.7 User pointer

Each monitor has a user pointer that can be set with glfwSetMonitorUserPointer and queried with glfwGetMonitorUserPointer. This can be used for any purpose you need and will not be modified by GLFW. The value will be kept until the monitor is disconnected or until the library is terminated.

The initial value of the pointer is NULL.

11.2.8 Gamma ramp

The gamma ramp of a monitor can be set with glfwSetGammaRamp, which accepts a monitor handle and a pointer to a GLFWgammaramp structure.

```
GLFWgammaramp ramp;
unsigned short red[256], green[256], blue[256];
ramp.size = 256;
ramp.red = red;
ramp.green = green;
ramp.blue = blue;
for (i = 0;  i < ramp.size;  i++)
{
    // Fill out gamma ramp arrays as desired
}
glfwSetGammaRamp(monitor, &ramp);
```

The gamma ramp data is copied before the function returns, so there is no need to keep it around once the ramp has been set.

It is recommended that your gamma ramp have the same size as the current gamma ramp for that monitor.

The current gamma ramp for a monitor is returned by glfwGetGammaRamp. See the reference documentation for the lifetime of the returned structure.

```
const GLFWgammaramp* ramp = glfwGetGammaRamp(monitor);
```

If you wish to set a regular gamma ramp, you can have GLFW calculate it for you from the desired exponent with glfwSetGamma, which in turn calls glfwSetGammaRamp with the resulting ramp.

```
glfwSetGamma(monitor, 1.0);
```

To experiment with gamma correction via the glfwSetGamma function, run the `gamma` test program.

Note

The software controlled gamma ramp is applied *in addition* to the hardware gamma correction, which today is usually an approximation of sRGB gamma. This means that setting a perfectly linear ramp, or gamma 1.0, will produce the default (usually sRGB-like) behavior.

Chapter 12

Moving from GLFW 2 to 3

This is a transition guide for moving from GLFW 2 to 3. It describes what has changed or been removed, but does *not* include new features unless they are required when moving an existing code base onto the new API. For example, the new multi-monitor functions are required to create full screen windows with GLFW 3.

12.1 Changed and removed features

12.1.1 Renamed library and header file

The GLFW 3 header is named glfw3.h and moved to the `GLFW` directory, to avoid collisions with the headers of other major versions. Similarly, the GLFW 3 library is named `glfw3`, except when it's installed as a shared library on Unix-like systems, where it uses the `soname` `libglfw.so.3`.

Old syntax

```
#include <GL/glfw.h>
```

New syntax

```
#include <GLFW/glfw3.h>
```

12.1.2 Removal of threading functions

The threading functions have been removed, including the per-thread sleep function. They were fairly primitive, under-used, poorly integrated and took time away from the focus of GLFW (i.e. context, input and window). There are better threading libraries available and native threading support is available in both `C++11` and `C11`, both of which are gaining traction.

If you wish to use the C++11 or C11 facilities but your compiler doesn't yet support them, see the `TinyThread++` and `TinyCThread` projects created by the original author of GLFW. These libraries implement a usable subset of the threading APIs in C++11 and C11, and in fact some GLFW 3 test programs use TinyCThread.

However, GLFW 3 has better support for *use from multiple threads* than GLFW 2 had. Contexts can be made current on any thread, although only a single thread at a time, and the documentation explicitly states which functions may be used from any thread and which must only be used from the main thread.

Removed functions

`glfwSleep`, `glfwCreateThread`, `glfwDestroyThread`, `glfwWaitThread`, `glfwGetThreadID`, `glfwCreateMutex`, `glfwDestroyMutex`, `glfwLockMutex`, `glfwUnlockMutex`, `glfwCreateCond`, `glfwDestroyCond`, `glfwWaitCond`, `glfwSignalCond`, `glfwBroadcastCond` and `glfwGetNumberOfProcessors`.

Removed types

`GLFWthreadfun`

12.1.3 Removal of image and texture loading

The image and texture loading functions have been removed. They only supported the Targa image format, making them mostly useful for beginner level examples. To become of sufficiently high quality to warrant keeping them in GLFW 3, they would need not only to support other formats, but also modern extensions to OpenGL texturing. This would either add a number of external dependencies (libjpeg, libpng, etc.), or force GLFW to ship with inline versions of these libraries.

As there already are libraries doing this, it is unnecessary both to duplicate the work and to tie the duplicate to GLFW. The resulting library would also be platform-independent, as both OpenGL and stdio are available wherever GLFW is.

Removed functions

`glfwReadImage`, `glfwReadMemoryImage`, `glfwFreeImage`, `glfwLoadTexture2D`, `glfwLoadMemoryTexture2D` and `glfwLoadTextureImage2D`.

12.1.4 Removal of GLFWCALL macro

The `GLFWCALL` macro, which made callback functions use `__stdcall` on Windows, has been removed. GLFW is written in C, not Pascal. Removing this macro means there's one less thing for application programmers to remember, i.e. the requirement to mark all callback functions with `GLFWCALL`. It also simplifies the creation of DLLs and DLL link libraries, as there's no need to explicitly disable `@n` entry point suffixes.

Old syntax

```
void GLFWCALL callback_function(...);
```

New syntax

```
void callback_function(...);
```

12.1.5 Window handle parameters

Because GLFW 3 supports multiple windows, window handle parameters have been added to all window-related GLFW functions and callbacks. The handle of a newly created window is returned by glfwCreateWindow (formerly `glfwOpenWindow`). Window handles are pointers to the `opaque` type GLFWwindow.

Old syntax

```
glfwSetWindowTitle("New Window Title");
```

New syntax

```
glfwSetWindowTitle(window, "New Window Title");
```

12.1.6 Explicit monitor selection

GLFW 3 provides support for multiple monitors. To request a full screen mode window, instead of passing GLFW_FULLSCREEN you specify which monitor you wish the window to use. The glfwGetPrimaryMonitor function returns the monitor that GLFW 2 would have selected, but there are many other monitor functions. Monitor handles are pointers to the opaque type GLFWmonitor.

Old basic full screen

```
glfwOpenWindow(640, 480, 8, 8, 8, 0, 24, 0, GLFW_FULLSCREEN);
```

New basic full screen

```
window = glfwCreateWindow(640, 480, "My Window", glfwGetPrimaryMonitor(), NULL);
```

Note

The framebuffer bit depth parameters of glfwOpenWindow have been turned into window hints, but as they have been given sane defaults you rarely need to set these hints.

12.1.7 Removal of automatic event polling

GLFW 3 does not automatically poll for events in glfwSwapBuffers, meaning you need to call glfwPollEvents or glfwWaitEvents yourself. Unlike buffer swap, which acts on a single window, the event processing functions act on all windows at once.

Old basic main loop

```
while (...)
{
    // Process input
    // Render output
    glfwSwapBuffers();
}
```

New basic main loop

```
while (...)
{
    // Process input
    // Render output
    glfwSwapBuffers(window);
    glfwPollEvents();
}
```

12.1.8 Explicit context management

Each GLFW 3 window has its own OpenGL context and only you, the application programmer, can know which context should be current on which thread at any given time. Therefore, GLFW 3 leaves that decision to you.

This means that you need to call glfwMakeContextCurrent after creating a window before you can call any OpenGL functions.

12.1.9 Separation of window and framebuffer sizes

Window positions and sizes now use screen coordinates, which may not be the same as pixels on machines with high-DPI monitors. This is important as OpenGL uses pixels, not screen coordinates. For example, the rectangle specified with `glViewport` needs to use pixels. Therefore, framebuffer size functions have been added. You can retrieve the size of the framebuffer of a window with glfwGetFramebufferSize function. A framebuffer size callback has also been added, which can be set with glfwSetFramebufferSizeCallback.

Old basic viewport setup

```
glfwGetWindowSize(&width, &height);
glViewport(0, 0, width, height);
```

New basic viewport setup

```
glfwGetFramebufferSize(window, &width, &height);
glViewport(0, 0, width, height);
```

12.1.10 Window closing changes

The `GLFW_OPENED` window parameter has been removed. As long as the window has not been destroyed, whether through glfwDestroyWindow or glfwTerminate, the window is "open".

A user attempting to close a window is now just an event like any other. Unlike GLFW 2, windows and contexts created with GLFW 3 will never be destroyed unless you choose them to be. Each window now has a close flag that is set to `GLFW_TRUE` when the user attempts to close that window. By default, nothing else happens and the window stays visible. It is then up to you to either destroy the window, take some other action or ignore the request.

You can query the close flag at any time with glfwWindowShouldClose and set it at any time with glfwSetWindowShouldClose.

Old basic main loop

```
while (glfwGetWindowParam(GLFW_OPENED))
{
    ...
}
```

New basic main loop

```
while (!glfwWindowShouldClose(window))
{
    ...
}
```

The close callback no longer returns a value. Instead, it is called after the close flag has been set so it can override its value, if it chooses to, before event processing completes. You may however not call glfwDestroyWindow from the close callback (or any other window related callback).

Old syntax

```
int GLFWCALL window_close_callback(void);
```

New syntax

```
void window_close_callback(GLFWwindow* window);
```

Note

GLFW never clears the close flag to `GLFW_FALSE`, meaning you can use it for other reasons to close the window as well, for example the user choosing Quit from an in-game menu.

12.1.11 Persistent window hints

The `glfwOpenWindowHint` function has been renamed to glfwWindowHint.

Window hints are no longer reset to their default values on window creation, but instead retain their values until modified by glfwWindowHint or glfwDefaultWindowHints, or until the library is terminated and re-initialized.

12.1.12 Video mode enumeration

Video mode enumeration is now per-monitor. The glfwGetVideoModes function now returns all available modes for a specific monitor instead of requiring you to guess how large an array you need. The `glfwGetDesktopMode` function, which had poorly defined behavior, has been replaced by glfwGetVideoMode, which returns the current mode of a monitor.

12.1.13 Removal of character actions

The action parameter of the character callback has been removed. This was an artefact of the origin of GLFW, i.e. being developed in English by a Swede. However, many keyboard layouts require more than one key to produce characters with diacritical marks. Even the Swedish keyboard layout requires this for uncommon cases like ü.

Old syntax

```
void GLFWCALL character_callback(int character, int action);
```

New syntax

```
void character_callback(GLFWwindow* window, int character);
```

12.1.14 Cursor position changes

The `glfwGetMousePos` function has been renamed to glfwGetCursorPos, `glfwSetMousePos` to glfwSetCursorPos and `glfwSetMousePosCallback` to glfwSetCursorPosCallback.

The cursor position is now `double` instead of `int`, both for the direct functions and for the callback. Some platforms can provide sub-pixel cursor movement and this data is now passed on to the application where available. On platforms where this is not provided, the decimal part is zero.

GLFW 3 only allows you to position the cursor within a window using glfwSetCursorPos (formerly `glfwSet↩MousePos`) when that window is active. Unless the window is active, the function fails silently.

12.1.15 Wheel position replaced by scroll offsets

The `glfwGetMouseWheel` function has been removed. Scrolling is the input of offsets and has no absolute position. The mouse wheel callback has been replaced by a scroll callback that receives two-dimensional floating point scroll offsets. This allows you to receive precise scroll data from for example modern touchpads.

Old syntax

```
void GLFWCALL mouse_wheel_callback(int position);
```

New syntax

```
void scroll_callback(GLFWwindow* window, double xoffset, double yoffset);
```

Removed functions

`glfwGetMouseWheel`

12.1.16 Key repeat action

The GLFW_KEY_REPEAT enable has been removed and key repeat is always enabled for both keys and characters. A new key action, GLFW_REPEAT, has been added to allow the key callback to distinguish an initial key press from a repeat. Note that glfwGetKey still returns only GLFW_PRESS or GLFW_RELEASE.

12.1.17 Physical key input

GLFW 3 key tokens map to physical keys, unlike in GLFW 2 where they mapped to the values generated by the current keyboard layout. The tokens are named according to the values they would have using the standard US layout, but this is only a convenience, as most programmers are assumed to know that layout. This means that (for example) GLFW_KEY_LEFT_BRACKET is always a single key and is the same key in the same place regardless of what keyboard layouts the users of your program has.

The key input facility was never meant for text input, although using it that way worked slightly better in GLFW 2. If you were using it to input text, you should be using the character callback instead, on both GLFW 2 and 3. This will give you the characters being input, as opposed to the keys being pressed.

GLFW 3 has key tokens for all keys on a standard 105 key keyboard, so instead of having to remember whether to check for a or A, you now check for GLFW_KEY_A.

12.1.18 Joystick function changes

The glfwGetJoystickPos function has been renamed to glfwGetJoystickAxes.

The glfwGetJoystickParam function and the GLFW_PRESENT, GLFW_AXES and GLFW_BUTTONS tokens have been replaced by the glfwJoystickPresent function as well as axis and button counts returned by the glfwGetJoystickAxes and glfwGetJoystickButtons functions.

12.1.19 Win32 MBCS support

The Win32 port of GLFW 3 will not compile in MBCS mode. However, because the use of the Unicode version of the Win32 API doesn't affect the process as a whole, but only those windows created using it, it's perfectly possible to call MBCS functions from other parts of the same application. Therefore, even if an application using GLFW has MBCS mode code, there's no need for GLFW itself to support it.

12.1.20 Support for versions of Windows older than XP

All explicit support for version of Windows older than XP has been removed. There is no code that actively prevents GLFW 3 from running on these earlier versions, but it uses Win32 functions that those versions lack.

Windows XP was released in 2001, and by now (January 2015) it has not only replaced almost all earlier versions of Windows, but is itself rapidly being replaced by Windows 7 and 8. The MSDN library doesn't even provide documentation for version older than Windows 2000, making it difficult to maintain compatibility with these versions even if it was deemed worth the effort.

The Win32 API has also not stood still, and GLFW 3 uses many functions only present on Windows XP or later. Even supporting an OS as new as XP (new from the perspective of GLFW 2, which still supports Windows 95) requires runtime checking for a number of functions that are present only on modern version of Windows.

12.1.21 Capture of system-wide hotkeys

The ability to disable and capture system-wide hotkeys like Alt+Tab has been removed. Modern applications, whether they're games, scientific visualisations or something else, are nowadays expected to be good desktop citizens and allow these hotkeys to function even when running in full screen mode.

12.1.22 Automatic termination

GLFW 3 does not register glfwTerminate with `atexit` at initialization, because `exit` calls registered functions from the calling thread and while it is permitted to call `exit` from any thread, glfwTerminate must only be called from the main thread.

To release all resources allocated by GLFW, you should call glfwTerminate yourself, from the main thread, before the program terminates. Note that this destroys all windows not already destroyed with glfwDestroyWindow, invalidating any window handles you may still have.

12.1.23 GLU header inclusion

GLFW 3 does not by default include the GLU header and GLU itself has been deprecated by `Khronos`. **New projects should not use GLU**, but if you need it for legacy code that has been moved to GLFW 3, you can request that the GLFW header includes it by defining GLFW_INCLUDE_GLU before the inclusion of the GLFW header.

Old syntax

```
#include <GL/glfw.h>
```

New syntax

```
#define GLFW_INCLUDE_GLU
#include <GLFW/glfw3.h>
```

There are many libraries that offer replacements for the functionality offered by GLU. For the matrix helper functions, see math libraries like `GLM` (for C++), `linmath.h` (for C) and others. For the tessellation functions, see for example `libtess2`.

12.2 Name change tables

12.2.1 Renamed functions

GLFW 2	GLFW 3	Notes
`glfwOpenWindow`	glfwCreateWindow	All channel bit depths are now hints
`glfwCloseWindow`	glfwDestroyWindow	
`glfwOpenWindowHint`	glfwWindowHint	Now accepts all `GLFW_*_BITS` tokens
`glfwEnable`	glfwSetInputMode	
`glfwDisable`	glfwSetInputMode	
`glfwGetMousePos`	glfwGetCursorPos	
`glfwSetMousePos`	glfwSetCursorPos	
~~`glfwSetMousePosCallback`~~	~~glfwSetCursorPosCallback~~	

GLFW 2	GLFW 3	Notes
glfwSetMouseWheelCallback	glfwSetScrollCallback	Accepts two-dimensional scroll offsets as doubles
glfwGetJoystickPos	glfwGetJoystickAxes	
glfwGetWindowParam	glfwGetWindowAttrib	
glfwGetGLVersion	glfwGetWindowAttrib	Use GLFW_CONTEXT_VERSION_MAJOR, GLFW_CONTEXT_VERSION_MINOR and GLFW_CONTEXT_REVISION
glfwGetDesktopMode	glfwGetVideoMode	Returns the current mode of a monitor
glfwGetJoystickParam	glfwJoystickPresent	The axis and button counts are provided by glfwGetJoystickAxes and glfwGetJoystickButtons

12.2.2 Renamed types

GLFW 2	GLFW 3	Notes
GLFWmousewheelfun	GLFWscrollfun	
GLFWmouseposfun	GLFWcursorposfun	

12.2.3 Renamed tokens

GLFW 2	GLFW 3	Notes
GLFW_OPENGL_VERSION_MAJOR	GLFW_CONTEXT_VERSION_MAJOR	Renamed as it applies to OpenGL ES as well
GLFW_OPENGL_VERSION_MINOR	GLFW_CONTEXT_VERSION_MINOR	Renamed as it applies to OpenGL ES as well
GLFW_FSAA_SAMPLES	GLFW_SAMPLES	Renamed to match the OpenGL API
GLFW_ACTIVE	GLFW_FOCUSED	Renamed to match the window focus callback
GLFW_WINDOW_NO_RESIZE	GLFW_RESIZABLE	The default has been inverted
GLFW_MOUSE_CURSOR	GLFW_CURSOR	Used with glfwSetInputMode
GLFW_KEY_ESC	GLFW_KEY_ESCAPE	
GLFW_KEY_DEL	GLFW_KEY_DELETE	
GLFW_KEY_PAGEUP	GLFW_KEY_PAGE_UP	
GLFW_KEY_PAGEDOWN	GLFW_KEY_PAGE_DOWN	
GLFW_KEY_KP_NUM_LOCK	GLFW_KEY_NUM_LOCK	
GLFW_KEY_LCTRL	GLFW_KEY_LEFT_CONTROL	
GLFW_KEY_LSHIFT	GLFW_KEY_LEFT_SHIFT	
GLFW_KEY_LALT	GLFW_KEY_LEFT_ALT	
GLFW_KEY_LSUPER	GLFW_KEY_LEFT_SUPER	
GLFW_KEY_RCTRL	GLFW_KEY_RIGHT_CONTROL	
GLFW_KEY_RSHIFT	GLFW_KEY_RIGHT_SHIFT	
GLFW_KEY_RALT	GLFW_KEY_RIGHT_ALT	
GLFW_KEY_RSUPER	GLFW_KEY_RIGHT_SUPER	

Chapter 13

Release notes

13.1 Release notes for version 3.4

13.1.1 New features in version 3.4

13.1.1.1 Runtime platform selection

GLFW now supports being compiled for multiple backends and selecting between them at runtime with the GLFW_PLATFORM init hint. After initialization the selected platform can be queried with glfwGetPlatform. You can check if support for a given platform is compiled in with glfwPlatformSupported.

13.1.1.2 More standard cursors

GLFW now provides the standard cursor shapes GLFW_RESIZE_NWSE_CURSOR and GLFW_RESIZE_NESW_CURSOR for diagonal resizing, GLFW_RESIZE_ALL_CURSOR for omni-directional resizing and GLFW_NOT_ALLOWED_CURSOR for showing an action is not allowed.

Unlike the original set, these shapes may not be available everywhere and creation will then fail with the new GLFW_CURSOR_UNAVAILABLE error.

The cursors for horizontal and vertical resizing are now referred to as GLFW_RESIZE_EW_CURSOR and GLFW_RESIZE_NS_CURSOR, and the pointing hand cursor is now referred to as GLFW_POINTING_HAND_CURSOR. The older names are still available.

For more information see Standard cursor creation.

13.1.1.3 Mouse event passthrough

GLFW now provides the GLFW_MOUSE_PASSTHROUGH window hint for making a window transparent to mouse input, lettings events pass to whatever window is behind it. This can also be changed after window creation with the matching window attribute.

13.1.1.4 Support for ANGLE rendering backend selection

GLFW now provides the GLFW_ANGLE_PLATFORM_TYPE init hint for requesting a specific rendering backend when using `ANGLE` to create OpenGL ES contexts.

13.1.1.5 Support for custom memory allocator

GLFW now supports plugging a custom memory allocator at initialization with glfwInitAllocator. The allocator is a struct of type GLFWallocator with function pointers corresponding to the standard library functions `malloc`, `realloc` and `free`.

For more information see Custom heap memory allocator.

13.1.1.6 Support for keyboard access to Windows window menu

GLFW now provides the GLFW_WIN32_KEYBOARD_MENU window hint for enabling keyboard access to the window menu via the Alt+Space and Alt-and-then-Space shortcuts. This may be useful for more GUI-oriented applications.

13.1.2 Caveats for version 3.4

13.1.2.1 Multiple sets of native access functions

Because GLFW now supports runtime selection of platform (window system), a library binary may export native access functions for multiple platforms. Starting with version 3.4 you must not assume that GLFW is running on a platform just because it exports native access functions for it. After initialization you can query the selected platform with glfwGetPlatform.

13.1.2.2 Version string format has been changed

Because GLFW now supports runtime selection of platform (window system), the version string returned by glfwGetVersionString has been expanded. It now contains the names of all APIs for all the platforms that the library binary supports.

13.1.2.3 Joystick support is initialized on demand

The joystick part of GLFW is now initialized when first used, primarily to work around faulty Windows drivers that cause DirectInput to take up to several seconds to enumerate devices.

This change will usually not be observable. However, if your application waits for events without having first called any joystick function or created any visible windows, the wait may never unblock as GLFW may not yet have subscribed to joystick related OS events.

To work around this, call any joystick function before waiting for events, for example by setting a joystick callback.

13.1.2.4 Tests and examples are disabled when built as a sub-project

GLFW now does not build the tests and examples when it is added as a subdirectory of another CMake project. To enable these, set the GLFW_BUILD_TESTS and GLFW_BUILD_EXAMPLES cache variables before adding the GLFW subdirectory.

```
set(GLFW_BUILD_EXAMPLES ON CACHE BOOL "" FORCE)
set(GLFW_BUILD_TESTS ON CACHE BOOL "" FORCE)
add_subdirectory(path/to/glfw)
```

13.1.2.5 macOS main menu now created at initialization

GLFW now creates the main menu and completes the initialization of NSApplication during initialization. Programs that do not want a main menu can disable it with the GLFW_COCOA_MENUBAR init hint.

13.1.2.6 CoreVideo dependency has been removed

GLFW no longer depends on the CoreVideo framework on macOS and it no longer needs to be specified during compilation or linking.

13.1.2.7 Framebuffer transparency requires DWM transparency

GLFW no longer supports framebuffer transparency enabled via GLFW_TRANSPARENT_FRAMEBUFFER on Windows 7 if DWM transparency is off (the Transparency setting under Personalization > Window Color).

13.1.2.8 Empty events on X11 no longer roundtrip to server

Events posted with glfwPostEmptyEvent now use a separate unnamed pipe instead of sending an X11 client event to the helper window.

13.1.3 Deprecations in version 3.4

13.1.4 Removals in 3.4

13.1.4.1 GLFW_VULKAN_STATIC CMake option has been removed

This option was used to compile GLFW directly linked with the Vulkan loader, instead of using dynamic loading to get hold of `vkGetInstanceProcAddr` at initialization. This is now done by calling the glfwInitVulkanLoader function before initialization.

If you need backward compatibility, this macro can still be defined for GLFW 3.4 and will have no effect. The call to glfwInitVulkanLoader can be conditionally enabled in your code by checking the GLFW_VERSION_MAJOR and GLFW_VERSION_MINOR macros.

13.1.4.2 GLFW_USE_OSMESA CMake option has been removed

This option was used to compile GLFW for the Null platform. The Null platform is now always supported. To produce a library binary that only supports this platform, the way this CMake option used to do, you will instead need to disable the default platform for the target OS. This means setting the GLFW_BUILD_WIN32, GLFW_BUILD_COCOA or GLFW_BUILD_X11 CMake option to false.

You can set all of them to false and the ones that don't apply for the target OS will be ignored.

13.1.4.3 Support for the wl_shell protocol has been removed

Support for the wl_shell protocol has been removed and GLFW now only supports the XDG-Shell protocol. If your Wayland compositor does not support XDG-Shell then GLFW will fail to initialize.

13.1.5 New symbols in version 3.4

13.1.5.1 New functions in version 3.4

- glfwInitAllocator
- glfwGetPlatform
- glfwPlatformSupported
- glfwInitVulkanLoader

13.1.5.2 New types in version 3.4

- GLFWallocator
- GLFWallocatefun
- GLFWreallocatefun
- GLFWdeallocatefun

13.1.5.3 New constants in version 3.4

- GLFW_PLATFORM
- GLFW_ANY_PLATFORM
- GLFW_PLATFORM_WIN32
- GLFW_PLATFORM_COCOA
- GLFW_PLATFORM_WAYLAND
- GLFW_PLATFORM_X11
- GLFW_PLATFORM_NULL
- GLFW_PLATFORM_UNAVAILABLE
- GLFW_POINTING_HAND_CURSOR
- GLFW_RESIZE_EW_CURSOR
- GLFW_RESIZE_NS_CURSOR
- GLFW_RESIZE_NWSE_CURSOR
- GLFW_RESIZE_NESW_CURSOR
- GLFW_RESIZE_ALL_CURSOR
- GLFW_MOUSE_PASSTHROUGH
- GLFW_NOT_ALLOWED_CURSOR
- GLFW_CURSOR_UNAVAILABLE
- GLFW_WIN32_KEYBOARD_MENU
- GLFW_CONTEXT_DEBUG
- GLFW_FEATURE_UNAVAILABLE

- GLFW_FEATURE_UNIMPLEMENTED
- GLFW_ANGLE_PLATFORM_TYPE
- GLFW_ANGLE_PLATFORM_TYPE_NONE
- GLFW_ANGLE_PLATFORM_TYPE_OPENGL
- GLFW_ANGLE_PLATFORM_TYPE_OPENGLES
- GLFW_ANGLE_PLATFORM_TYPE_D3D9
- GLFW_ANGLE_PLATFORM_TYPE_D3D11
- GLFW_ANGLE_PLATFORM_TYPE_VULKAN
- GLFW_ANGLE_PLATFORM_TYPE_METAL
- GLFW_X11_XCB_VULKAN_SURFACE

13.2 Release notes for earlier versions

- Release notes for 3.3
- Release notes for 3.2
- Release notes for 3.1
- Release notes for 3.0

Chapter 14

Getting started

This guide takes you through writing a small application using GLFW 3. The application will create a window and OpenGL context, render a rotating triangle and exit when the user closes the window or presses *Escape*. This guide will introduce a few of the most commonly used functions, but there are many more.

This guide assumes no experience with earlier versions of GLFW. If you have used GLFW 2 in the past, read Moving from GLFW 2 to 3, as some functions behave differently in GLFW 3.

14.1 Step by step

14.1.1 Including the GLFW header

In the source files of your application where you use GLFW, you need to include its header file.

```
#include <GLFW/glfw3.h>
```

This header provides all the constants, types and function prototypes of the GLFW API.

By default it also includes the OpenGL header from your development environment. On some platforms this header only supports older versions of OpenGL. The most extreme case is Windows, where it typically only supports OpenGL 1.2.

Most programs will instead use an extension loader library and include its header. This example uses files generated by `glad`. The GLFW header can detect most such headers if they are included first and will then not include the one from your development environment.

```
#include <glad/gl.h>
#include <GLFW/glfw3.h>
```

To make sure there will be no header conflicts, you can define GLFW_INCLUDE_NONE before the GLFW header to explicitly disable inclusion of the development environment header. This also allows the two headers to be included in any order.

```
#define GLFW_INCLUDE_NONE
#include <GLFW/glfw3.h>
#include <glad/gl.h>
```

14.1.2 Initializing and terminating GLFW

Before you can use most GLFW functions, the library must be initialized. On successful initialization, GLFW_TRUE is returned. If an error occurred, GLFW_FALSE is returned.

```
if (!glfwInit())
{
    // Initialization failed
}
```

Note that GLFW_TRUE and GLFW_FALSE are and will always be one and zero.

When you are done using GLFW, typically just before the application exits, you need to terminate GLFW.

```
glfwTerminate();
```

This destroys any remaining windows and releases any other resources allocated by GLFW. After this call, you must initialize GLFW again before using any GLFW functions that require it.

14.1.3 Setting an error callback

Most events are reported through callbacks, whether it's a key being pressed, a GLFW window being moved, or an error occurring. Callbacks are C functions (or C++ static methods) that are called by GLFW with arguments describing the event.

In case a GLFW function fails, an error is reported to the GLFW error callback. You can receive these reports with an error callback. This function must have the signature below but may do anything permitted in other callbacks.

```
void error_callback(int error, const char* description)
{
    fprintf(stderr, "Error: %s\n", description);
}
```

Callback functions must be set, so GLFW knows to call them. The function to set the error callback is one of the few GLFW functions that may be called before initialization, which lets you be notified of errors both during and after initialization.

```
glfwSetErrorCallback(error_callback);
```

14.1.4 Creating a window and context

The window and its OpenGL context are created with a single call to glfwCreateWindow, which returns a handle to the created combined window and context object

```
GLFWwindow* window = glfwCreateWindow(640, 480, "My Title", NULL, NULL);
if (!window)
{
    // Window or OpenGL context creation failed
}
```

This creates a 640 by 480 windowed mode window with an OpenGL context. If window or OpenGL context creation fails, NULL will be returned. You should always check the return value. While window creation rarely fails, context creation depends on properly installed drivers and may fail even on machines with the necessary hardware.

By default, the OpenGL context GLFW creates may have any version. You can require a minimum OpenGL version by setting the GLFW_CONTEXT_VERSION_MAJOR and GLFW_CONTEXT_VERSION_MINOR hints *before* creation. If the required minimum version is not supported on the machine, context (and window) creation fails.

You can select the OpenGL profile by setting the GLFW_OPENGL_PROFILE hint. This program uses the core profile as that is the only profile macOS supports for OpenGL 3.x and 4.x.

```
glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
GLFWwindow* window = glfwCreateWindow(640, 480, "My Title", NULL, NULL);
if (!window)
{
    // Window or context creation failed
}
```

The window handle is passed to all window related functions and is provided to along to all window related callbacks, so they can tell which window received the event.

When a window and context is no longer needed, destroy it.

```
glfwDestroyWindow(window);
```

Once this function is called, no more events will be delivered for that window and its handle becomes invalid.

14.1.5 Making the OpenGL context current

Before you can use the OpenGL API, you must have a current OpenGL context.

```
glfwMakeContextCurrent(window);
```

The context will remain current until you make another context current or until the window owning the current context is destroyed.

If you are using an extension loader library to access modern OpenGL then this is when to initialize it, as the loader needs a current context to load from. This example uses `glad`, but the same rule applies to all such libraries.

```
gladLoadGL(glfwGetProcAddress);
```

14.1.6 Checking the window close flag

Each window has a flag indicating whether the window should be closed.

When the user attempts to close the window, either by pressing the close widget in the title bar or using a key combination like Alt+F4, this flag is set to 1. Note that **the window isn't actually closed**, so you are expected to monitor this flag and either destroy the window or give some kind of feedback to the user.

```
while (!glfwWindowShouldClose(window))
{
    // Keep running
}
```

You can be notified when the user is attempting to close the window by setting a close callback with glfwSetWindowCloseCallback. The callback will be called immediately after the close flag has been set.

You can also set it yourself with glfwSetWindowShouldClose. This can be useful if you want to interpret other kinds of input as closing the window, like for example pressing the *Escape* key.

14.1.7 Receiving input events

Each window has a large number of callbacks that can be set to receive all the various kinds of events. To receive key press and release events, create a key callback function.

```
static void key_callback(GLFWwindow* window, int key, int scancode, int action, int mods)
{
    if (key == GLFW_KEY_ESCAPE && action == GLFW_PRESS)
        glfwSetWindowShouldClose(window, GLFW_TRUE);
}
```

The key callback, like other window related callbacks, are set per-window.

```
glfwSetKeyCallback(window, key_callback);
```

In order for event callbacks to be called when events occur, you need to process events as described below.

14.1.8 Rendering with OpenGL

Once you have a current OpenGL context, you can use OpenGL normally. In this tutorial, a multi-colored rotating triangle will be rendered. The framebuffer size needs to be retrieved for `glViewport`.

```
int width, height;
glfwGetFramebufferSize(window, &width, &height);
glViewport(0, 0, width, height);
```

You can also set a framebuffer size callback using glfwSetFramebufferSizeCallback and be notified when the size changes.

The details of how to render with OpenGL is outside the scope of this tutorial, but there are many excellent resources for learning modern OpenGL. Here are a few of them:

- `Anton's OpenGL 4 Tutorials`
- `Learn OpenGL`
- `Open.GL`

These all happen to use GLFW, but OpenGL itself works the same whatever API you use to create the window and context.

14.1.9 Reading the timer

To create smooth animation, a time source is needed. GLFW provides a timer that returns the number of seconds since initialization. The time source used is the most accurate on each platform and generally has micro- or nanosecond resolution.

```
double time = glfwGetTime();
```

14.1.10 Swapping buffers

GLFW windows by default use double buffering. That means that each window has two rendering buffers; a front buffer and a back buffer. The front buffer is the one being displayed and the back buffer the one you render to.

When the entire frame has been rendered, the buffers need to be swapped with one another, so the back buffer becomes the front buffer and vice versa.

```
glfwSwapBuffers(window);
```

The swap interval indicates how many frames to wait until swapping the buffers, commonly known as *vsync*. By default, the swap interval is zero, meaning buffer swapping will occur immediately. On fast machines, many of those frames will never be seen, as the screen is still only updated typically 60-75 times per second, so this wastes a lot of CPU and GPU cycles.

Also, because the buffers will be swapped in the middle the screen update, leading to screen tearing.

For these reasons, applications will typically want to set the swap interval to one. It can be set to higher values, but this is usually not recommended, because of the input latency it leads to.

```
glfwSwapInterval(1);
```

This function acts on the current context and will fail unless a context is current.

14.1.11 Processing events

GLFW needs to communicate regularly with the window system both in order to receive events and to show that the application hasn't locked up. Event processing must be done regularly while you have visible windows and is normally done each frame after buffer swapping.

There are two methods for processing pending events; polling and waiting. This example will use event polling, which processes only those events that have already been received and then returns immediately.

```
glfwPollEvents();
```

This is the best choice when rendering continually, like most games do. If instead you only need to update your rendering once you have received new input, glfwWaitEvents is a better choice. It waits until at least one event has been received, putting the thread to sleep in the meantime, and then processes all received events. This saves a great deal of CPU cycles and is useful for, for example, many kinds of editing tools.

14.2 Putting it together

Now that you know how to initialize GLFW, create a window and poll for keyboard input, it's possible to create a small program.

This program creates a 640 by 480 windowed mode window and starts a loop that clears the screen, renders a triangle and processes events until the user either presses *Escape* or closes the window.

The program above can be found in the source package as `examples/triangle-opengl.c` and is compiled along with all other examples when you build GLFW. If you built GLFW from the source package then you already have this as `triangle-opengl.exe` on Windows, `triangle-opengl` on Linux or `triangle-opengl.app` on macOS.

This tutorial used only a few of the many functions GLFW provides. There are guides for each of the areas covered by GLFW. Each guide will introduce all the functions for that category.

- Introduction to the API
- Window guide
- Context guide
- Monitor guide
- Input guide

You can access reference documentation for any GLFW function by clicking it and the reference for each function links to related functions and guide sections.

The tutorial ends here. Once you have written a program that uses GLFW, you will need to compile and link it. How to do that depends on the development environment you are using and is best explained by the documentation for that environment. To learn about the details that are specific to GLFW, see Building applications.

Chapter 15

Support resources

See the latest documentation for tutorials, guides and the API reference.

If you have questions about using GLFW, we have a forum, and the #glfw IRC channel on Libera.Chat.

Bugs are reported to our issue tracker. Please check the contribution guide for information on what to include when reporting a bug.

Chapter 16

Vulkan guide

This guide is intended to fill the gaps between the official Vulkan resources and the rest of the GLFW documentation and is not a replacement for either. It assumes some familiarity with Vulkan concepts like loaders, devices, queues and surfaces and leaves it to the Vulkan documentation to explain the details of Vulkan functions.

To develop for Vulkan you should download the LunarG Vulkan SDK for your platform. Apart from headers and link libraries, they also provide the validation layers necessary for development.

The Vulkan Tutorial has more information on how to use GLFW and Vulkan. The Khronos Vulkan Samples also use GLFW, although with a small framework in between.

For details on a specific Vulkan support function, see the Vulkan support reference. There are also guides for the other areas of the GLFW API.

- Introduction to the API
- Window guide
- Context guide
- Monitor guide
- Input guide

16.1 Finding the Vulkan loader

GLFW itself does not ever need to be linked against the Vulkan loader.

By default, GLFW will load the Vulkan loader dynamically at runtime via its standard name: `vulkan-1.dll` on Windows, `libvulkan.so.1` on Linux and other Unix-like systems and `libvulkan.1.dylib` on macOS.

@macos GLFW will also look up and search the executable subdirectory of your application bundle.

If your code is using a Vulkan loader with a different name or in a non-standard location you will need to direct GLFW to it. Pass your version of `vkGetInstanceProcAddr` to glfwInitVulkanLoader before initializing GLFW and it will use that function for all Vulkan entry point retrieval. This prevents GLFW from dynamically loading the Vulkan loader.

```
glfwInitVulkanLoader(vkGetInstanceProcAddr);
```

@macos To make your application be redistributable you will need to set up the application bundle according to the LunarG SDK documentation. This is explained in more detail in the SDK documentation for macOS.

16.2 Including the Vulkan header file

To have GLFW include the Vulkan header, define GLFW_INCLUDE_VULKAN before including the GLFW header.

```
#define GLFW_INCLUDE_VULKAN
#include <GLFW/glfw3.h>
```

If you instead want to include the Vulkan header from a custom location or use your own custom Vulkan header then do this before the GLFW header.

```
#include <path/to/vulkan.h>
#include <GLFW/glfw3.h>
```

Unless a Vulkan header is included, either by the GLFW header or above it, the following GLFW functions will not be declared, as depend on Vulkan types.

- glfwInitVulkanLoader
- glfwGetInstanceProcAddress
- glfwGetPhysicalDevicePresentationSupport
- glfwCreateWindowSurface

The `VK_USE_PLATFORM_*_KHR` macros do not need to be defined for the Vulkan part of GLFW to work. Define them only if you are using these extensions directly.

16.3 Querying for Vulkan support

If you are linking directly against the Vulkan loader then you can skip this section. The canonical desktop loader library exports all Vulkan core and Khronos extension functions, allowing them to be called directly.

If you are loading the Vulkan loader dynamically instead of linking directly against it, you can check for the availability of a loader and ICD with glfwVulkanSupported.

```
if (glfwVulkanSupported())
{
    // Vulkan is available, at least for compute
}
```

This function returns `GLFW_TRUE` if the Vulkan loader and any minimally functional ICD was found.

If one or both were not found, calling any other Vulkan related GLFW function will generate a GLFW_API_UNAVAILABLE error.

16.3.1 Querying Vulkan function pointers

To load any Vulkan core or extension function from the found loader, call glfwGetInstanceProcAddress. To load functions needed for instance creation, pass `NULL` as the instance.

```
PFN_vkCreateInstance pfnCreateInstance = (PFN_vkCreateInstance)
    glfwGetInstanceProcAddress(NULL, "vkCreateInstance");
```

Once you have created an instance, you can load from it all other Vulkan core functions and functions from any instance extensions you enabled.

```
PFN_vkCreateDevice pfnCreateDevice = (PFN_vkCreateDevice)
    glfwGetInstanceProcAddress(instance, "vkCreateDevice");
```

This function in turn calls `vkGetInstanceProcAddr`. If that fails, the function falls back to a platform-specific query of the Vulkan loader (i.e. `dlsym` or `GetProcAddress`). If that also fails, the function returns `NULL`. For more information about `vkGetInstanceProcAddr`, see the Vulkan documentation.

Vulkan also provides `vkGetDeviceProcAddr` for loading device-specific versions of Vulkan function. This function can be retrieved from an instance with glfwGetInstanceProcAddress.

```
PFN_vkGetDeviceProcAddr pfnGetDeviceProcAddr = (PFN_vkGetDeviceProcAddr)
    glfwGetInstanceProcAddress(instance, "vkGetDeviceProcAddr");
```

Device-specific functions may execute a little bit faster, due to not having to dispatch internally based on the device passed to them. For more information about `vkGetDeviceProcAddr`, see the Vulkan documentation.

16.4 Querying required Vulkan extensions

To do anything useful with Vulkan you need to create an instance. If you want to use Vulkan to render to a window, you must enable the instance extensions GLFW requires to create Vulkan surfaces.

To query the instance extensions required, call glfwGetRequiredInstanceExtensions.

```
uint32_t count;
const char** extensions = glfwGetRequiredInstanceExtensions(&count);
```

These extensions must all be enabled when creating instances that are going to be passed to glfwGetPhysical↩ DevicePresentationSupport and glfwCreateWindowSurface. The set of extensions will vary depending on platform and may also vary depending on graphics drivers and other factors.

If it fails it will return `NULL` and GLFW will not be able to create Vulkan window surfaces. You can still use Vulkan for off-screen rendering and compute work.

If successful the returned array will always include `VK_KHR_surface`, so if you don't require any additional extensions you can pass this list directly to the `VkInstanceCreateInfo` struct.

```
VkInstanceCreateInfo ici;
memset(&ici, 0, sizeof(ici));
ici.enabledExtensionCount = count;
ici.ppEnabledExtensionNames = extensions;
...
```

Additional extensions may be required by future versions of GLFW. You should check whether any extensions you wish to enable are already in the returned array, as it is an error to specify an extension more than once in the `VkInstanceCreateInfo` struct.

16.5 Querying for Vulkan presentation support

Not every queue family of every Vulkan device can present images to surfaces. To check whether a specific queue family of a physical device supports image presentation without first having to create a window and surface, call glfwGetPhysicalDevicePresentationSupport.

```
if (glfwGetPhysicalDevicePresentationSupport(instance, physical_device, queue_family_index))
{
    // Queue family supports image presentation
}
```

The `VK_KHR_surface` extension additionally provides the `vkGetPhysicalDeviceSurfaceSupport↩ KHR` function, which performs the same test on an existing Vulkan surface.

16.6 Creating the window

Unless you will be using OpenGL or OpenGL ES with the same window as Vulkan, there is no need to create a context. You can disable context creation with the GLFW_CLIENT_API hint.

```
glfwWindowHint(GLFW_CLIENT_API, GLFW_NO_API);
GLFWwindow* window = glfwCreateWindow(640, 480, "Window Title", NULL, NULL);
```

See Windows without contexts for more information.

16.7 Creating a Vulkan window surface

You can create a Vulkan surface (as defined by the `VK_KHR_surface` extension) for a GLFW window with glfw↩ CreateWindowSurface.

```
VkSurfaceKHR surface;
VkResult err = glfwCreateWindowSurface(instance, window, NULL, &surface);
if (err)
{
    // Window surface creation failed
}
```

If an OpenGL or OpenGL ES context was created on the window, the context has ownership of the presentation on the window and a Vulkan surface cannot be created.

It is your responsibility to destroy the surface. GLFW does not destroy it for you. Call `vkDestroySurfaceKHR` function from the same extension to destroy it.

Chapter 17

Window guide

This guide introduces the window related functions of GLFW. For details on a specific function in this category, see the Window reference. There are also guides for the other areas of GLFW.

- Introduction to the API
- Context guide
- Vulkan guide
- Monitor guide
- Input guide

17.1 Window objects

The GLFWwindow object encapsulates both a window and a context. They are created with glfwCreateWindow and destroyed with glfwDestroyWindow, or glfwTerminate, if any remain. As the window and context are inseparably linked, the object pointer is used as both a context and window handle.

To see the event stream provided to the various window related callbacks, run the `events` test program.

17.1.1 Window creation

A window and its OpenGL or OpenGL ES context are created with glfwCreateWindow, which returns a handle to the created window object. For example, this creates a 640 by 480 windowed mode window:

```
GLFWwindow* window = glfwCreateWindow(640, 480, "My Title", NULL, NULL);
```

If window creation fails, `NULL` will be returned, so it is necessary to check the return value.

The window handle is passed to all window related functions and is provided to along with all input events, so event handlers can tell which window received the event.

17.1.1.1 Full screen windows

To create a full screen window, you need to specify which monitor the window should use. In most cases, the user's primary monitor is a good choice. For more information about retrieving monitors, see Retrieving monitors.

```
GLFWwindow* window = glfwCreateWindow(640, 480, "My Title", glfwGetPrimaryMonitor(), NULL);
```

Full screen windows cover the entire display area of a monitor, have no border or decorations.

Windowed mode windows can be made full screen by setting a monitor with glfwSetWindowMonitor, and full screen ones can be made windowed by unsetting it with the same function.

Each field of the GLFWvidmode structure corresponds to a function parameter or window hint and combine to form the *desired video mode* for that window. The supported video mode most closely matching the desired video mode will be set for the chosen monitor as long as the window has input focus. For more information about retrieving video modes, see Video modes.

Video mode field	Corresponds to
GLFWvidmode.width	`width` parameter of glfwCreateWindow
GLFWvidmode.height	`height` parameter of glfwCreateWindow
GLFWvidmode.redBits	GLFW_RED_BITS hint
GLFWvidmode.greenBits	GLFW_GREEN_BITS hint
GLFWvidmode.blueBits	GLFW_BLUE_BITS hint
GLFWvidmode.refreshRate	GLFW_REFRESH_RATE hint

Once you have a full screen window, you can change its resolution, refresh rate and monitor with glfwSetWindowMonitor. If you only need change its resolution you can also call glfwSetWindowSize. In all cases, the new video mode will be selected the same way as the video mode chosen by glfwCreateWindow. If the window has an OpenGL or OpenGL ES context, it will be unaffected.

By default, the original video mode of the monitor will be restored and the window iconified if it loses input focus, to allow the user to switch back to the desktop. This behavior can be disabled with the GLFW_AUTO_ICONIFY window hint, for example if you wish to simultaneously cover multiple monitors with full screen windows.

If a monitor is disconnected, all windows that are full screen on that monitor will be switched to windowed mode. See Monitor configuration changes for more information.

17.1.1.2 "Windowed full screen" windows

If the closest match for the desired video mode is the current one, the video mode will not be changed, making window creation faster and application switching much smoother. This is sometimes called *windowed full screen* or *borderless full screen* window and counts as a full screen window. To create such a window, request the current video mode.

```
const GLFWvidmode* mode = glfwGetVideoMode(monitor);
glfwWindowHint(GLFW_RED_BITS, mode->redBits);
glfwWindowHint(GLFW_GREEN_BITS, mode->greenBits);
glfwWindowHint(GLFW_BLUE_BITS, mode->blueBits);
glfwWindowHint(GLFW_REFRESH_RATE, mode->refreshRate);
GLFWwindow* window = glfwCreateWindow(mode->width, mode->height, "My Title", monitor, NULL);
```

This also works for windowed mode windows that are made full screen.

```
const GLFWvidmode* mode = glfwGetVideoMode(monitor);
glfwSetWindowMonitor(window, monitor, 0, 0, mode->width, mode->height, mode->refreshRate);
```

Note that glfwGetVideoMode returns the *current* video mode of a monitor, so if you already have a full screen window on that monitor that you want to make windowed full screen, you need to have saved the desktop resolution before.

17.1.2 Window destruction

When a window is no longer needed, destroy it with glfwDestroyWindow.

```
glfwDestroyWindow(window);
```

Window destruction always succeeds. Before the actual destruction, all callbacks are removed so no further events will be delivered for the window. All windows remaining when glfwTerminate is called are destroyed as well.

When a full screen window is destroyed, the original video mode of its monitor is restored, but the gamma ramp is left untouched.

17.1.3 Window creation hints

There are a number of hints that can be set before the creation of a window and context. Some affect the window itself, others affect the framebuffer or context. These hints are set to their default values each time the library is initialized with glfwInit. Integer value hints can be set individually with glfwWindowHint and string value hints with glfwWindowHintString. You can reset all at once to their defaults with glfwDefaultWindowHints.

Some hints are platform specific. These are always valid to set on any platform but they will only affect their specific platform. Other platforms will ignore them. Setting these hints requires no platform specific headers or calls.

Note

Window hints need to be set before the creation of the window and context you wish to have the specified attributes. They function as additional arguments to glfwCreateWindow.

17.1.3.1 Hard and soft constraints

Some window hints are hard constraints. These must match the available capabilities *exactly* for window and context creation to succeed. Hints that are not hard constraints are matched as closely as possible, but the resulting context and framebuffer may differ from what these hints requested.

The following hints are always hard constraints:

- GLFW_STEREO
- GLFW_DOUBLEBUFFER
- GLFW_CLIENT_API
- GLFW_CONTEXT_CREATION_API

The following additional hints are hard constraints when requesting an OpenGL context, but are ignored when requesting an OpenGL ES context:

- GLFW_OPENGL_FORWARD_COMPAT
- GLFW_OPENGL_PROFILE

17.1.3.2 Window related hints

GLFW_RESIZABLE specifies whether the windowed mode window will be resizable *by the user*. The window will still be resizable using the glfwSetWindowSize function. Possible values are `GLFW_TRUE` and `GLFW_FALSE`. This hint is ignored for full screen and undecorated windows.

GLFW_VISIBLE specifies whether the windowed mode window will be initially visible. Possible values are `GLFW_TRUE` and `GLFW_FALSE`. This hint is ignored for full screen windows.

GLFW_DECORATED specifies whether the windowed mode window will have window decorations such as a border, a close widget, etc. An undecorated window will not be resizable by the user but will still allow the user

to generate close events on some platforms. Possible values are GLFW_TRUE and GLFW_FALSE. This hint is ignored for full screen windows.

GLFW_FOCUSED specifies whether the windowed mode window will be given input focus when created. Possible values are GLFW_TRUE and GLFW_FALSE. This hint is ignored for full screen and initially hidden windows.

GLFW_AUTO_ICONIFY specifies whether the full screen window will automatically iconify and restore the previous video mode on input focus loss. Possible values are GLFW_TRUE and GLFW_FALSE. This hint is ignored for windowed mode windows.

GLFW_FLOATING specifies whether the windowed mode window will be floating above other regular windows, also called topmost or always-on-top. This is intended primarily for debugging purposes and cannot be used to implement proper full screen windows. Possible values are GLFW_TRUE and GLFW_FALSE. This hint is ignored for full screen windows.

GLFW_MAXIMIZED specifies whether the windowed mode window will be maximized when created. Possible values are GLFW_TRUE and GLFW_FALSE. This hint is ignored for full screen windows.

GLFW_CENTER_CURSOR specifies whether the cursor should be centered over newly created full screen windows. Possible values are GLFW_TRUE and GLFW_FALSE. This hint is ignored for windowed mode windows.

GLFW_TRANSPARENT_FRAMEBUFFER specifies whether the window framebuffer will be transparent. If enabled and supported by the system, the window framebuffer alpha channel will be used to combine the framebuffer with the background. This does not affect window decorations. Possible values are GLFW_TRUE and GLFW_FALSE.

GLFW_FOCUS_ON_SHOW specifies whether the window will be given input focus when glfwShowWindow is called. Possible values are GLFW_TRUE and GLFW_FALSE.

GLFW_SCALE_TO_MONITOR specified whether the window content area should be resized based on the monitor content scale of any monitor it is placed on. This includes the initial placement when the window is created. Possible values are GLFW_TRUE and GLFW_FALSE.

This hint only has an effect on platforms where screen coordinates and pixels always map 1:1 such as Windows and X11. On platforms like macOS the resolution of the framebuffer is changed independently of the window size.

GLFW_MOUSE_PASSTHROUGH specifies whether the window is transparent to mouse input, letting any mouse events pass through to whatever window is behind it. This is only supported for undecorated windows. Decorated windows with this enabled will behave differently between platforms. Possible values are GLFW_TRUE and GLFW↩_FALSE.

17.1.3.3 Framebuffer related hints

GLFW_RED_BITS, **GLFW_GREEN_BITS**, **GLFW_BLUE_BITS**, **GLFW_ALPHA_BITS**, **GLFW_DEPTH_BITS** and **GLFW_STENCIL_BITS** specify the desired bit depths of the various components of the default framebuffer. A value of GLFW_DONT_CARE means the application has no preference.

GLFW_ACCUM_RED_BITS, **GLFW_ACCUM_GREEN_BITS**, **GLFW_ACCUM_BLUE_BITS** and **GLFW_↩ACCUM_ALPHA_BITS** specify the desired bit depths of the various components of the accumulation buffer. A value of GLFW_DONT_CARE means the application has no preference.

Accumulation buffers are a legacy OpenGL feature and should not be used in new code.

GLFW_AUX_BUFFERS specifies the desired number of auxiliary buffers. A value of GLFW_DONT_CARE means the application has no preference.

Auxiliary buffers are a legacy OpenGL feature and should not be used in new code.

GLFW_STEREO specifies whether to use OpenGL stereoscopic rendering. Possible values are GLFW_TRUE and GLFW_FALSE. This is a hard constraint.

GLFW_SAMPLES specifies the desired number of samples to use for multisampling. Zero disables multisampling. A value of GLFW_DONT_CARE means the application has no preference.

GLFW_SRGB_CAPABLE specifies whether the framebuffer should be sRGB capable. Possible values are GLFW↩_TRUE and GLFW_FALSE.

Note

OpenGL: If enabled and supported by the system, the GL_FRAMEBUFFER_SRGB enable will control sRGB rendering. By default, sRGB rendering will be disabled.

OpenGL ES: If enabled and supported by the system, the context will always have sRGB rendering enabled.

GLFW_DOUBLEBUFFER specifies whether the framebuffer should be double buffered. You nearly always want to use double buffering. This is a hard constraint. Possible values are GLFW_TRUE and GLFW_FALSE.

17.1.3.4 Monitor related hints

GLFW_REFRESH_RATE specifies the desired refresh rate for full screen windows. A value of GLFW_DONT_CARE means the highest available refresh rate will be used. This hint is ignored for windowed mode windows.

17.1.3.5 Context related hints

GLFW_CLIENT_API specifies which client API to create the context for. Possible values are GLFW_OPENGL_API, GLFW_OPENGL_ES_API and GLFW_NO_API. This is a hard constraint.

GLFW_CONTEXT_CREATION_API specifies which context creation API to use to create the context. Possible values are GLFW_NATIVE_CONTEXT_API, GLFW_EGL_CONTEXT_API and GLFW_OSMESA_CONTEXT_API. This is a hard constraint. If no client API is requested, this hint is ignored.

An extension loader library that assumes it knows which API was used to create the current context may fail if you change this hint. This can be resolved by having it load functions via glfwGetProcAddress.

Note

@wayland The EGL API *is* the native context creation API, so this hint will have no effect.

@x11 On some Linux systems, creating contexts via both the native and EGL APIs in a single process will cause the application to segfault. Stick to one API or the other on Linux for now.

OSMesa: As its name implies, an OpenGL context created with OSMesa does not update the window contents when its buffers are swapped. Use OpenGL functions or the OSMesa native access functions glfwGetOSMesaColorBuffer and glfwGetOSMesaDepthBuffer to retrieve the framebuffer contents.

GLFW_CONTEXT_VERSION_MAJOR and **GLFW_CONTEXT_VERSION_MINOR** specify the client API version that the created context must be compatible with. The exact behavior of these hints depend on the requested client API.

While there is no way to ask the driver for a context of the highest supported version, GLFW will attempt to provide this when you ask for a version 1.0 context, which is the default for these hints.

Do not confuse these hints with GLFW_VERSION_MAJOR and GLFW_VERSION_MINOR, which provide the API version of the GLFW header.

Note

OpenGL: These hints are not hard constraints, but creation will fail if the OpenGL version of the created context is less than the one requested. It is therefore perfectly safe to use the default of version 1.0 for legacy code and you will still get backwards-compatible contexts of version 3.0 and above when available.

OpenGL ES: These hints are not hard constraints, but creation will fail if the OpenGL ES version of the created context is less than the one requested. Additionally, OpenGL ES 1.x cannot be returned if 2.0 or later was requested, and vice versa. This is because OpenGL ES 3.x is backward compatible with 2.0, but OpenGL ES 2.0 is not backward compatible with 1.x.

@macos The OS only supports core profile contexts for OpenGL versions 3.2 and later. Before creating an OpenGL context of version 3.2 or later you must set the GLFW_OPENGL_PROFILE hint accordingly. OpenGL 3.0 and 3.1 contexts are not supported at all on macOS.

GLFW_OPENGL_FORWARD_COMPAT specifies whether the OpenGL context should be forward-compatible, i.e. one where all functionality deprecated in the requested version of OpenGL is removed. This must only be used if the requested OpenGL version is 3.0 or above. If OpenGL ES is requested, this hint is ignored.

Forward-compatibility is described in detail in the OpenGL Reference Manual.

GLFW_CONTEXT_DEBUG specifies whether the context should be created in debug mode, which may provide additional error and diagnostic reporting functionality. Possible values are GLFW_TRUE and GLFW_FALSE.

Debug contexts for OpenGL and OpenGL ES are described in detail by the GL_KHR_debug extension.

Note

GLFW_CONTEXT_DEBUG is the new name introduced in GLFW 3.4. The older GLFW_OPENGL_DEBUG↩ _CONTEXT name is also available for compatibility.

GLFW_OPENGL_PROFILE specifies which OpenGL profile to create the context for. Possible values are one of GLFW_OPENGL_CORE_PROFILE or GLFW_OPENGL_COMPAT_PROFILE, or GLFW_OPENGL_ANY_↩ PROFILE to not request a specific profile. If requesting an OpenGL version below 3.2, GLFW_OPENGL_ANY_↩ PROFILE must be used. If OpenGL ES is requested, this hint is ignored.

OpenGL profiles are described in detail in the OpenGL Reference Manual.

GLFW_CONTEXT_ROBUSTNESS specifies the robustness strategy to be used by the context. This can be one of GLFW_NO_RESET_NOTIFICATION or GLFW_LOSE_CONTEXT_ON_RESET, or GLFW_NO_ROBUSTNESS to not request a robustness strategy.

GLFW_CONTEXT_RELEASE_BEHAVIOR specifies the release behavior to be used by the context. Possible values are one of GLFW_ANY_RELEASE_BEHAVIOR, GLFW_RELEASE_BEHAVIOR_FLUSH or GLFW↩ _RELEASE_BEHAVIOR_NONE. If the behavior is GLFW_ANY_RELEASE_BEHAVIOR, the default behavior of the context creation API will be used. If the behavior is GLFW_RELEASE_BEHAVIOR_FLUSH, the pipeline will be flushed whenever the context is released from being the current one. If the behavior is GLFW_RELEASE_↩ BEHAVIOR_NONE, the pipeline will not be flushed on release.

Context release behaviors are described in detail by the GL_KHR_context_flush_control extension.

GLFW_CONTEXT_NO_ERROR specifies whether errors should be generated by the context. Possible values are GLFW_TRUE and GLFW_FALSE. If enabled, situations that would have generated errors instead cause undefined behavior.

The no error mode for OpenGL and OpenGL ES is described in detail by the GL_KHR_no_error extension.

17.1.3.6 Win32 specific hints

GLFW_WIN32_KEYBOARD_MENU specifies whether to allow access to the window menu via the Alt+Space and Alt-and-then-Space keyboard shortcuts. This is ignored on other platforms.

17.1.3.7 macOS specific hints

GLFW_COCOA_RETINA_FRAMEBUFFER specifies whether to use full resolution framebuffers on Retina displays. Possible values are `GLFW_TRUE` and `GLFW_FALSE`. This is ignored on other platforms.

GLFW_COCOA_FRAME_NAME specifies the UTF-8 encoded name to use for autosaving the window frame, or if empty disables frame autosaving for the window. This is ignored on other platforms. This is set with glfwWindowHintString.

GLFW_COCOA_GRAPHICS_SWITCHING specifies whether to in Automatic Graphics Switching, i.e. to allow the system to choose the integrated GPU for the OpenGL context and move it between GPUs if necessary or whether to force it to always run on the discrete GPU. This only affects systems with both integrated and discrete GPUs. Possible values are `GLFW_TRUE` and `GLFW_FALSE`. This is ignored on other platforms.

Simpler programs and tools may want to enable this to save power, while games and other applications performing advanced rendering will want to leave it disabled.

A bundled application that wishes to participate in Automatic Graphics Switching should also declare this in its `Info.plist` by setting the `NSSupportsAutomaticGraphicsSwitching` key to `true`.

17.1.3.8 X11 specific window hints

GLFW_X11_CLASS_NAME and **GLFW_X11_INSTANCE_NAME** specifies the desired ASCII encoded class and instance parts of the ICCCM `WM_CLASS` window property. These are set with glfwWindowHintString.

17.1.3.9 Supported and default values

Window hint	Default value	Supported values
GLFW_RESIZABLE	`GLFW_TRUE`	`GLFW_TRUE` or `GLFW_FALSE`
GLFW_VISIBLE	`GLFW_TRUE`	`GLFW_TRUE` or `GLFW_FALSE`
GLFW_DECORATED	`GLFW_TRUE`	`GLFW_TRUE` or `GLFW_FALSE`
GLFW_FOCUSED	`GLFW_TRUE`	`GLFW_TRUE` or `GLFW_FALSE`
GLFW_AUTO_ICONIFY	`GLFW_TRUE`	`GLFW_TRUE` or `GLFW_FALSE`
GLFW_FLOATING	`GLFW_FALSE`	`GLFW_TRUE` or `GLFW_FALSE`
GLFW_MAXIMIZED	`GLFW_FALSE`	`GLFW_TRUE` or `GLFW_FALSE`
GLFW_CENTER_CURSOR	`GLFW_TRUE`	`GLFW_TRUE` or `GLFW_FALSE`
GLFW_TRANSPARENT_↩ FRAMEBUFFER	`GLFW_FALSE`	`GLFW_TRUE` or `GLFW_FALSE`
GLFW_FOCUS_ON_SHOW	`GLFW_TRUE`	`GLFW_TRUE` or `GLFW_FALSE`
GLFW_SCALE_TO_MONITOR	`GLFW_FALSE`	`GLFW_TRUE` or `GLFW_FALSE`
GLFW_MOUSE_PASSTHROUGH	`GLFW_FALSE`	`GLFW_TRUE` or `GLFW_FALSE`
GLFW_RED_BITS	8	0 to `INT_MAX` or `GLFW_DONT`↩ `_CARE`
GLFW_GREEN_BITS	8	0 to `INT_MAX` or `GLFW_DONT`↩ `_CARE`

Window hint	Default value	Supported values
GLFW_BLUE_BITS	8	0 to INT_MAX or GLFW_DONT_CARE
GLFW_ALPHA_BITS	8	0 to INT_MAX or GLFW_DONT_CARE
GLFW_DEPTH_BITS	24	0 to INT_MAX or GLFW_DONT_CARE
GLFW_STENCIL_BITS	8	0 to INT_MAX or GLFW_DONT_CARE
GLFW_ACCUM_RED_BITS	0	0 to INT_MAX or GLFW_DONT_CARE
GLFW_ACCUM_GREEN_BITS	0	0 to INT_MAX or GLFW_DONT_CARE
GLFW_ACCUM_BLUE_BITS	0	0 to INT_MAX or GLFW_DONT_CARE
GLFW_ACCUM_ALPHA_BITS	0	0 to INT_MAX or GLFW_DONT_CARE
GLFW_AUX_BUFFERS	0	0 to INT_MAX or GLFW_DONT_CARE
GLFW_SAMPLES	0	0 to INT_MAX or GLFW_DONT_CARE
GLFW_REFRESH_RATE	GLFW_DONT_CARE	0 to INT_MAX or GLFW_DONT_CARE
GLFW_STEREO	GLFW_FALSE	GLFW_TRUE or GLFW_FALSE
GLFW_SRGB_CAPABLE	GLFW_FALSE	GLFW_TRUE or GLFW_FALSE
GLFW_DOUBLEBUFFER	GLFW_TRUE	GLFW_TRUE or GLFW_FALSE
GLFW_CLIENT_API	GLFW_OPENGL_API	GLFW_OPENGL_API, GLFW_OPENGL_ES_API or GLFW_NO_API
GLFW_CONTEXT_CREATION_API	GLFW_NATIVE_CONTEXT_API	GLFW_NATIVE_CONTEXT_API, GLFW_EGL_CONTEXT_API or GLFW_OSMESA_CONTEXT_API
GLFW_CONTEXT_VERSION_MAJOR	1	Any valid major version number of the chosen client API
GLFW_CONTEXT_VERSION_MINOR	0	Any valid minor version number of the chosen client API
GLFW_CONTEXT_ROBUSTNESS	GLFW_NO_ROBUSTNESS	GLFW_NO_ROBUSTNESS, GLFW_NO_RESET_NOTIFICATION or GLFW_LOSE_CONTEXT_ON_RESET
GLFW_CONTEXT_RELEASE_BEHAVIOR	GLFW_ANY_RELEASE_BEHAVIOR	GLFW_ANY_RELEASE_BEHAVIOR, GLFW_RELEASE_BEHAVIOR_FLUSH or GLFW_RELEASE_BEHAVIOR_NONE
GLFW_OPENGL_FORWARD_COMPAT	GLFW_FALSE	GLFW_TRUE or GLFW_FALSE
GLFW_CONTEXT_DEBUG	GLFW_FALSE	GLFW_TRUE or GLFW_FALSE
GLFW_OPENGL_PROFILE	GLFW_OPENGL_ANY_PROFILE	GLFW_OPENGL_ANY_PROFILE, GLFW_OPENGL_COMPAT_PROFILE or GLFW_OPENGL_CORE_PROFILE
GLFW_WIN32_KEYBOARD_MENU	GLFW_FALSE	GLFW_TRUE or GLFW_FALSE

Window hint	Default value	Supported values
GLFW_COCOA_RETINA_↩ FRAMEBUFFER	GLFW_TRUE	GLFW_TRUE or GLFW_FALSE
GLFW_COCOA_FRAME_NAME	""	A UTF-8 encoded frame autosave name
GLFW_COCOA_GRAPHICS_↩ SWITCHING	GLFW_FALSE	GLFW_TRUE or GLFW_FALSE
GLFW_X11_CLASS_NAME	""	An ASCII encoded WM_CLASS class name
GLFW_X11_INSTANCE_NAME	""	An ASCII encoded WM_CLASS instance name

17.2 Window event processing

See Event processing.

17.3 Window properties and events

17.3.1 User pointer

Each window has a user pointer that can be set with glfwSetWindowUserPointer and queried with glfwGetWindowUserPointer. This can be used for any purpose you need and will not be modified by GLFW throughout the life-time of the window.

The initial value of the pointer is NULL.

17.3.2 Window closing and close flag

When the user attempts to close the window, for example by clicking the close widget or using a key chord like Alt+F4, the *close flag* of the window is set. The window is however not actually destroyed and, unless you watch for this state change, nothing further happens.

The current state of the close flag is returned by glfwWindowShouldClose and can be set or cleared directly with glfwSetWindowShouldClose. A common pattern is to use the close flag as a main loop condition.

```
while (!glfwWindowShouldClose(window))
{
    render(window);
    glfwSwapBuffers(window);
    glfwPollEvents();
}
```

If you wish to be notified when the user attempts to close a window, set a close callback.

```
glfwSetWindowCloseCallback(window, window_close_callback);
```

The callback function is called directly *after* the close flag has been set. It can be used for example to filter close requests and clear the close flag again unless certain conditions are met.

```
void window_close_callback(GLFWwindow* window)
{
    if (!time_to_close)
        glfwSetWindowShouldClose(window, GLFW_FALSE);
}
```

17.3.3 Window size

The size of a window can be changed with glfwSetWindowSize. For windowed mode windows, this sets the size, in screen coordinates of the *content area* or *content area* of the window. The window system may impose limits on window size.

```
glfwSetWindowSize(window, 640, 480);
```

For full screen windows, the specified size becomes the new resolution of the window's desired video mode. The video mode most closely matching the new desired video mode is set immediately. The window is resized to fit the resolution of the set video mode.

If you wish to be notified when a window is resized, whether by the user, the system or your own code, set a size callback.

```
glfwSetWindowSizeCallback(window, window_size_callback);
```

The callback function receives the new size, in screen coordinates, of the content area of the window when the window is resized.

```
void window_size_callback(GLFWwindow* window, int width, int height)
{
}
```

There is also glfwGetWindowSize for directly retrieving the current size of a window.

```
int width, height;
glfwGetWindowSize(window, &width, &height);
```

Note

> Do not pass the window size to `glViewport` or other pixel-based OpenGL calls. The window size is in screen coordinates, not pixels. Use the framebuffer size, which is in pixels, for pixel-based calls.

The above functions work with the size of the content area, but decorated windows typically have title bars and window frames around this rectangle. You can retrieve the extents of these with glfwGetWindowFrameSize.

```
int left, top, right, bottom;
glfwGetWindowFrameSize(window, &left, &top, &right, &bottom);
```

The returned values are the distances, in screen coordinates, from the edges of the content area to the corresponding edges of the full window. As they are distances and not coordinates, they are always zero or positive.

17.3.4 Framebuffer size

While the size of a window is measured in screen coordinates, OpenGL works with pixels. The size you pass into `glViewport`, for example, should be in pixels. On some machines screen coordinates and pixels are the same, but on others they will not be. There is a second set of functions to retrieve the size, in pixels, of the framebuffer of a window.

If you wish to be notified when the framebuffer of a window is resized, whether by the user or the system, set a size callback.

```
glfwSetFramebufferSizeCallback(window, framebuffer_size_callback);
```

The callback function receives the new size of the framebuffer when it is resized, which can for example be used to update the OpenGL viewport.

```
void framebuffer_size_callback(GLFWwindow* window, int width, int height)
{
    glViewport(0, 0, width, height);
}
```

There is also glfwGetFramebufferSize for directly retrieving the current size of the framebuffer of a window.

```
int width, height;
glfwGetFramebufferSize(window, &width, &height);
glViewport(0, 0, width, height);
```

The size of a framebuffer may change independently of the size of a window, for example if the window is dragged between a regular monitor and a high-DPI one.

17.3.5 Window content scale

The content scale for a window can be retrieved with glfwGetWindowContentScale.
```
float xscale, yscale;
glfwGetWindowContentScale(window, &xscale, &yscale);
```

The content scale is the ratio between the current DPI and the platform's default DPI. This is especially important for text and any UI elements. If the pixel dimensions of your UI scaled by this look appropriate on your machine then it should appear at a reasonable size on other machines regardless of their DPI and scaling settings. This relies on the system DPI and scaling settings being somewhat correct.

On systems where each monitors can have its own content scale, the window content scale will depend on which monitor the system considers the window to be on.

If you wish to be notified when the content scale of a window changes, whether because of a system setting change or because it was moved to a monitor with a different scale, set a content scale callback.
```
glfwSetWindowContentScaleCallback(window, window_content_scale_callback);
```

The callback function receives the new content scale of the window.
```
void window_content_scale_callback(GLFWwindow* window, float xscale, float yscale)
{
    set_interface_scale(xscale, yscale);
}
```

On platforms where pixels and screen coordinates always map 1:1, the window will need to be resized to appear the same size when it is moved to a monitor with a different content scale. To have this done automatically both when the window is created and when its content scale later changes, set the GLFW_SCALE_TO_MONITOR window hint.

17.3.6 Window size limits

The minimum and maximum size of the content area of a windowed mode window can be enforced with glfwSetWindowSizeLimits. The user may resize the window to any size and aspect ratio within the specified limits, unless the aspect ratio is also set.
```
glfwSetWindowSizeLimits(window, 200, 200, 400, 400);
```

To specify only a minimum size or only a maximum one, set the other pair to GLFW_DONT_CARE.
```
glfwSetWindowSizeLimits(window, 640, 480, GLFW_DONT_CARE, GLFW_DONT_CARE);
```

To disable size limits for a window, set them all to GLFW_DONT_CARE.

The aspect ratio of the content area of a windowed mode window can be enforced with glfwSetWindowAspectRatio. The user may resize the window freely unless size limits are also set, but the size will be constrained to maintain the aspect ratio.
```
glfwSetWindowAspectRatio(window, 16, 9);
```

The aspect ratio is specified as a numerator and denominator, corresponding to the width and height, respectively. If you want a window to maintain its current aspect ratio, use its current size as the ratio.
```
int width, height;
glfwGetWindowSize(window, &width, &height);
glfwSetWindowAspectRatio(window, width, height);
```

To disable the aspect ratio limit for a window, set both terms to GLFW_DONT_CARE.

You can have both size limits and aspect ratio set for a window, but the results are undefined if they conflict.

17.3.7 Window position

The position of a windowed-mode window can be changed with glfwSetWindowPos. This moves the window so that the upper-left corner of its content area has the specified screen coordinates. The window system may put limitations on window placement.

```
glfwSetWindowPos(window, 100, 100);
```

If you wish to be notified when a window is moved, whether by the user, the system or your own code, set a position callback.

```
glfwSetWindowPosCallback(window, window_pos_callback);
```

The callback function receives the new position, in screen coordinates, of the upper-left corner of the content area when the window is moved.

```
void window_pos_callback(GLFWwindow* window, int xpos, int ypos)
{
}
```

There is also glfwGetWindowPos for directly retrieving the current position of the content area of the window.

```
int xpos, ypos;
glfwGetWindowPos(window, &xpos, &ypos);
```

17.3.8 Window title

All GLFW windows have a title, although undecorated or full screen windows may not display it or only display it in a task bar or similar interface. You can set a UTF-8 encoded window title with glfwSetWindowTitle.

```
glfwSetWindowTitle(window, "My Window");
```

The specified string is copied before the function returns, so there is no need to keep it around.

As long as your source file is encoded as UTF-8, you can use any Unicode characters directly in the source.

```
glfwSetWindowTitle(window, "");
```

If you are using C++11 or C11, you can use a UTF-8 string literal.

```
glfwSetWindowTitle(window, u8"This is always a UTF-8 string");
```

17.3.9 Window icon

Decorated windows have icons on some platforms. You can set this icon by specifying a list of candidate images with glfwSetWindowIcon.

```
GLFWimage images[2];
images[0] = load_icon("my_icon.png");
images[1] = load_icon("my_icon_small.png");
glfwSetWindowIcon(window, 2, images);
```

The image data is 32-bit, little-endian, non-premultiplied RGBA, i.e. eight bits per channel with the red channel first. The pixels are arranged canonically as sequential rows, starting from the top-left corner.

To revert to the default window icon, pass in an empty image array.

```
glfwSetWindowIcon(window, 0, NULL);
```

17.3.10 Window monitor

Full screen windows are associated with a specific monitor. You can get the handle for this monitor with glfwGetWindowMonitor.

```
GLFWmonitor* monitor = glfwGetWindowMonitor(window);
```

This monitor handle is one of those returned by glfwGetMonitors.

For windowed mode windows, this function returns `NULL`. This is how to tell full screen windows from windowed mode windows.

You can move windows between monitors or between full screen and windowed mode with glfwSetWindowMonitor. When making a window full screen on the same or on a different monitor, specify the desired monitor, resolution and refresh rate. The position arguments are ignored.

```
const GLFWvidmode* mode = glfwGetVideoMode(monitor);
glfwSetWindowMonitor(window, monitor, 0, 0, mode->width, mode->height, mode->refreshRate);
```

When making the window windowed, specify the desired position and size. The refresh rate argument is ignored.

```
glfwSetWindowMonitor(window, NULL, xpos, ypos, width, height, 0);
```

This restores any previous window settings such as whether it is decorated, floating, resizable, has size or aspect ratio limits, etc.. To restore a window that was originally windowed to its original size and position, save these before making it full screen and then pass them in as above.

17.3.11 Window iconification

Windows can be iconified (i.e. minimized) with glfwIconifyWindow.

```
glfwIconifyWindow(window);
```

When a full screen window is iconified, the original video mode of its monitor is restored until the user or application restores the window.

Iconified windows can be restored with glfwRestoreWindow. This function also restores windows from maximization.

```
glfwRestoreWindow(window);
```

When a full screen window is restored, the desired video mode is restored to its monitor as well.

If you wish to be notified when a window is iconified or restored, whether by the user, system or your own code, set an iconify callback.

```
glfwSetWindowIconifyCallback(window, window_iconify_callback);
```

The callback function receives changes in the iconification state of the window.

```
void window_iconify_callback(GLFWwindow* window, int iconified)
{
    if (iconified)
    {
        // The window was iconified
    }
    else
    {
        // The window was restored
    }
}
```

You can also get the current iconification state with glfwGetWindowAttrib.

```
int iconified = glfwGetWindowAttrib(window, GLFW_ICONIFIED);
```

17.3.12 Window maximization

Windows can be maximized (i.e. zoomed) with glfwMaximizeWindow.
```
glfwMaximizeWindow(window);
```

Full screen windows cannot be maximized and passing a full screen window to this function does nothing.

Maximized windows can be restored with glfwRestoreWindow. This function also restores windows from iconification.
```
glfwRestoreWindow(window);
```

If you wish to be notified when a window is maximized or restored, whether by the user, system or your own code, set a maximize callback.
```
glfwSetWindowMaximizeCallback(window, window_maximize_callback);
```

The callback function receives changes in the maximization state of the window.
```
void window_maximize_callback(GLFWwindow* window, int maximized)
{
    if (maximized)
    {
        // The window was maximized
    }
    else
    {
        // The window was restored
    }
}
```

You can also get the current maximization state with glfwGetWindowAttrib.
```
int maximized = glfwGetWindowAttrib(window, GLFW_MAXIMIZED);
```

By default, newly created windows are not maximized. You can change this behavior by setting the GLFW_MAXIMIZED window hint before creating the window.
```
glfwWindowHint(GLFW_MAXIMIZED, GLFW_TRUE);
```

17.3.13 Window visibility

Windowed mode windows can be hidden with glfwHideWindow.
```
glfwHideWindow(window);
```

This makes the window completely invisible to the user, including removing it from the task bar, dock or window list. Full screen windows cannot be hidden and calling glfwHideWindow on a full screen window does nothing.

Hidden windows can be shown with glfwShowWindow.
```
glfwShowWindow(window);
```

By default, this function will also set the input focus to that window. Set the GLFW_FOCUS_ON_SHOW window hint to change this behavior for all newly created windows, or change the behavior for an existing window with glfwSetWindowAttrib.

You can also get the current visibility state with glfwGetWindowAttrib.
```
int visible = glfwGetWindowAttrib(window, GLFW_VISIBLE);
```

By default, newly created windows are visible. You can change this behavior by setting the GLFW_VISIBLE window hint before creating the window.
```
glfwWindowHint(GLFW_VISIBLE, GLFW_FALSE);
```

Windows created hidden are completely invisible to the user until shown. This can be useful if you need to set up your window further before showing it, for example moving it to a specific location.

17.3.14 Window input focus

Windows can be given input focus and brought to the front with glfwFocusWindow.
```
glfwFocusWindow(window);
```

Keep in mind that it can be very disruptive to the user when a window is forced to the top. For a less disruptive way of getting the user's attention, see attention requests.

If you wish to be notified when a window gains or loses input focus, whether by the user, system or your own code, set a focus callback.
```
glfwSetWindowFocusCallback(window, window_focus_callback);
```

The callback function receives changes in the input focus state of the window.
```
void window_focus_callback(GLFWwindow* window, int focused)
{
    if (focused)
    {
        // The window gained input focus
    }
    else
    {
        // The window lost input focus
    }
}
```

You can also get the current input focus state with glfwGetWindowAttrib.
```
int focused = glfwGetWindowAttrib(window, GLFW_FOCUSED);
```

By default, newly created windows are given input focus. You can change this behavior by setting the GLFW_FOCUSED window hint before creating the window.
```
glfwWindowHint(GLFW_FOCUSED, GLFW_FALSE);
```

17.3.15 Window attention request

If you wish to notify the user of an event without interrupting, you can request attention with glfwRequestWindowAttention.
```
glfwRequestWindowAttention(window);
```

The system will highlight the specified window, or on platforms where this is not supported, the application as a whole. Once the user has given it attention, the system will automatically end the request.

17.3.16 Window damage and refresh

If you wish to be notified when the contents of a window is damaged and needs to be refreshed, set a window refresh callback.
```
glfwSetWindowRefreshCallback(m_handle, window_refresh_callback);
```

The callback function is called when the contents of the window needs to be refreshed.
```
void window_refresh_callback(GLFWwindow* window)
{
    draw_editor_ui(window);
    glfwSwapBuffers(window);
}
```

Note

On compositing window systems such as Aero, Compiz or Aqua, where the window contents are saved off-screen, this callback might only be called when the window or framebuffer is resized.

17.3.17 Window transparency

GLFW supports two kinds of transparency for windows; framebuffer transparency and whole window transparency. A single window may not use both methods. The results of doing this are undefined.

Both methods require the platform to support it and not every version of every platform GLFW supports does this, so there are mechanisms to check whether the window really is transparent.

Window framebuffers can be made transparent on a per-pixel per-frame basis with the GLFW_TRANSPARENT_FRAMEBUFFER window hint.

```
glfwWindowHint(GLFW_TRANSPARENT_FRAMEBUFFER, GLFW_TRUE);
```

If supported by the system, the window content area will be composited with the background using the framebuffer per-pixel alpha channel. This requires desktop compositing to be enabled on the system. It does not affect window decorations.

You can check whether the window framebuffer was successfully made transparent with the GLFW_TRANSPARENT_FRAMEBUFFER window attribute.

```
if (glfwGetWindowAttrib(window, GLFW_TRANSPARENT_FRAMEBUFFER))
{
    // window framebuffer is currently transparent
}
```

GLFW comes with an example that enabled framebuffer transparency called `gears`.

The opacity of the whole window, including any decorations, can be set with glfwSetWindowOpacity.

```
glfwSetWindowOpacity(window, 0.5f);
```

The opacity (or alpha) value is a positive finite number between zero and one, where 0 (zero) is fully transparent and 1 (one) is fully opaque. The initial opacity value for newly created windows is 1.

The current opacity of a window can be queried with glfwGetWindowOpacity.

```
float opacity = glfwGetWindowOpacity(window);
```

If the system does not support whole window transparency, this function always returns one.

GLFW comes with a test program that lets you control whole window transparency at run-time called `window`.

If you want to use either of these transparency methods to display a temporary overlay like for example a notification, the GLFW_FLOATING and GLFW_MOUSE_PASSTHROUGH window hints and attributes may be useful.

17.3.18 Window attributes

Windows have a number of attributes that can be returned using glfwGetWindowAttrib. Some reflect state that may change as a result of user interaction, (e.g. whether it has input focus), while others reflect inherent properties of the window (e.g. what kind of border it has). Some are related to the window and others to its OpenGL or OpenGL ES context.

```
if (glfwGetWindowAttrib(window, GLFW_FOCUSED))
{
    // window has input focus
}
```

The GLFW_DECORATED, GLFW_RESIZABLE, GLFW_FLOATING, GLFW_AUTO_ICONIFY and GLFW_FOCUS_ON_SHOW window attributes can be changed with glfwSetWindowAttrib.

```
glfwSetWindowAttrib(window, GLFW_RESIZABLE, GLFW_FALSE);
```

17.3.18.1 Window related attributes

GLFW_FOCUSED indicates whether the specified window has input focus. See Window input focus for details.

GLFW_ICONIFIED indicates whether the specified window is iconified. See Window iconification for details.

GLFW_MAXIMIZED indicates whether the specified window is maximized. See Window maximization for details.

GLFW_HOVERED indicates whether the cursor is currently directly over the content area of the window, with no other windows between. See Cursor enter/leave events for details.

GLFW_VISIBLE indicates whether the specified window is visible. See Window visibility for details.

GLFW_RESIZABLE indicates whether the specified window is resizable *by the user*. This can be set before creation with the GLFW_RESIZABLE window hint or after with glfwSetWindowAttrib.

GLFW_DECORATED indicates whether the specified window has decorations such as a border, a close widget, etc. This can be set before creation with the GLFW_DECORATED window hint or after with glfwSetWindowAttrib.

GLFW_AUTO_ICONIFY indicates whether the specified full screen window is iconified on focus loss, a close widget, etc. This can be set before creation with the GLFW_AUTO_ICONIFY window hint or after with glfwSetWindowAttrib.

GLFW_FLOATING indicates whether the specified window is floating, also called topmost or always-on-top. This can be set before creation with the GLFW_FLOATING window hint or after with glfwSetWindowAttrib.

GLFW_TRANSPARENT_FRAMEBUFFER indicates whether the specified window has a transparent framebuffer, i.e. the window contents is composited with the background using the window framebuffer alpha channel. See Window transparency for details.

GLFW_FOCUS_ON_SHOW specifies whether the window will be given input focus when glfwShowWindow is called. This can be set before creation with the GLFW_FOCUS_ON_SHOW window hint or after with glfwSetWindowAttrib.

GLFW_MOUSE_PASSTHROUGH specifies whether the window is transparent to mouse input, letting any mouse events pass through to whatever window is behind it. This can be set before creation with the GLFW_MOUSE_PASSTHROUGH window hint or after with glfwSetWindowAttrib. This is only supported for undecorated windows. Decorated windows with this enabled will behave differently between platforms.

17.3.18.2 Context related attributes

GLFW_CLIENT_API indicates the client API provided by the window's context; either `GLFW_OPENGL_API`, `GLFW_OPENGL_ES_API` or `GLFW_NO_API`.

GLFW_CONTEXT_CREATION_API indicates the context creation API used to create the window's context; either `GLFW_NATIVE_CONTEXT_API`, `GLFW_EGL_CONTEXT_API` or `GLFW_OSMESA_CONTEXT_API`.

GLFW_CONTEXT_VERSION_MAJOR, **GLFW_CONTEXT_VERSION_MINOR** and **GLFW_CONTEXT_↩ REVISION** indicate the client API version of the window's context.

Note

Do not confuse these attributes with `GLFW_VERSION_MAJOR`, `GLFW_VERSION_MINOR` and `GLFW_↩ VERSION_REVISION` which provide the API version of the GLFW header.

GLFW_OPENGL_FORWARD_COMPAT is `GLFW_TRUE` if the window's context is an OpenGL forward-compatible one, or `GLFW_FALSE` otherwise.

GLFW_CONTEXT_DEBUG is `GLFW_TRUE` if the window's context is in debug mode, or `GLFW_FALSE` otherwise.

This is the new name, introduced in GLFW 3.4. The older GLFW_OPENGL_DEBUG_CONTEXT name is also available for compatibility.

GLFW_OPENGL_PROFILE indicates the OpenGL profile used by the context. This is GLFW_OPENGL_CORE_PROFILE or GLFW_OPENGL_COMPAT_PROFILE if the context uses a known profile, or GLFW_OPENGL_ANY_PROFILE if the OpenGL profile is unknown or the context is an OpenGL ES context. Note that the returned profile may not match the profile bits of the context flags, as GLFW will try other means of detecting the profile when no bits are set.

GLFW_CONTEXT_RELEASE_BEHAVIOR indicates the release used by the context. Possible values are one of GLFW_ANY_RELEASE_BEHAVIOR, GLFW_RELEASE_BEHAVIOR_FLUSH or GLFW_RELEASE_BEHAVIOR_NONE. If the behavior is GLFW_ANY_RELEASE_BEHAVIOR, the default behavior of the context creation API will be used. If the behavior is GLFW_RELEASE_BEHAVIOR_FLUSH, the pipeline will be flushed whenever the context is released from being the current one. If the behavior is GLFW_RELEASE_BEHAVIOR_NONE, the pipeline will not be flushed on release.

GLFW_CONTEXT_NO_ERROR indicates whether errors are generated by the context. Possible values are GLFW_TRUE and GLFW_FALSE. If enabled, situations that would have generated errors instead cause undefined behavior.

GLFW_CONTEXT_ROBUSTNESS indicates the robustness strategy used by the context. This is GLFW_LOSE_CONTEXT_ON_RESET or GLFW_NO_RESET_NOTIFICATION if the window's context supports robustness, or GLFW_NO_ROBUSTNESS otherwise.

17.3.18.3 Framebuffer related attributes

GLFW does not expose most attributes of the default framebuffer (i.e. the framebuffer attached to the window) as these can be queried directly with either OpenGL, OpenGL ES or Vulkan. The one exception is GLFW_DOUBLEBUFFER, as this is not provided by OpenGL ES.

If you are using version 3.0 or later of OpenGL or OpenGL ES, the glGetFramebufferAttachmentParameteriv function can be used to retrieve the number of bits for the red, green, blue, alpha, depth and stencil buffer channels. Otherwise, the glGetIntegerv function can be used.

The number of MSAA samples are always retrieved with glGetIntegerv. For contexts supporting framebuffer objects, the number of samples of the currently bound framebuffer is returned.

Attribute	glGetIntegerv	glGetFramebufferAttachmentParameteriv
Red bits	GL_RED_BITS	GL_FRAMEBUFFER_ATTACHMENT_RED_SIZE
Green bits	GL_GREEN_BITS	GL_FRAMEBUFFER_ATTACHMENT_GREEN_SIZE
Blue bits	GL_BLUE_BITS	GL_FRAMEBUFFER_ATTACHMENT_BLUE_SIZE
Alpha bits	GL_ALPHA_BITS	GL_FRAMEBUFFER_ATTACHMENT_ALPHA_SIZE
Depth bits	GL_DEPTH_BITS	GL_FRAMEBUFFER_ATTACHMENT_DEPTH_SIZE
Stencil bits	GL_STENCIL_BITS	GL_FRAMEBUFFER_ATTACHMENT_STENCIL_SIZE
MSAA samples	GL_SAMPLES	*Not provided by this function*

When calling glGetFramebufferAttachmentParameteriv, the red, green, blue and alpha sizes are queried from the GL_BACK_LEFT, while the depth and stencil sizes are queried from the GL_DEPTH and GL_STENCIL attachments, respectively.

GLFW_DOUBLEBUFFER indicates whether the specified window is double-buffered when rendering with OpenGL or OpenGL ES. This can be set before creation with the GLFW_DOUBLEBUFFER window hint.

17.4 Buffer swapping

GLFW windows are by default double buffered. That means that you have two rendering buffers; a front buffer and a back buffer. The front buffer is the one being displayed and the back buffer the one you render to.

When the entire frame has been rendered, it is time to swap the back and the front buffers in order to display what has been rendered and begin rendering a new frame. This is done with glfwSwapBuffers.

```
glfwSwapBuffers(window);
```

Sometimes it can be useful to select when the buffer swap will occur. With the function glfwSwapInterval it is possible to select the minimum number of monitor refreshes the driver should wait from the time glfwSwapBuffers was called before swapping the buffers:

```
glfwSwapInterval(1);
```

If the interval is zero, the swap will take place immediately when glfwSwapBuffers is called without waiting for a refresh. Otherwise at least interval retraces will pass between each buffer swap. Using a swap interval of zero can be useful for benchmarking purposes, when it is not desirable to measure the time it takes to wait for the vertical retrace. However, a swap interval of one lets you avoid tearing.

Note that this may not work on all machines, as some drivers have user-controlled settings that override any swap interval the application requests.

A context that supports either the `WGL_EXT_swap_control_tear` or the `GLX_EXT_swap_control_tear` extension also accepts *negative* swap intervals, which allows the driver to swap immediately even if a frame arrives a little bit late. This trades the risk of visible tears for greater framerate stability. You can check for these extensions with glfwExtensionSupported.

Chapter 18

LICENSE

Chapter 19

GLFW

19.1 Introduction

GLFW is an Open Source, multi-platform library for OpenGL, OpenGL ES and Vulkan application development. It provides a simple, platform-independent API for creating windows, contexts and surfaces, reading input, handling events, etc.

GLFW natively supports Windows, macOS and Linux and other Unix-like systems. On Linux both X11 and Wayland are supported.

GLFW is licensed under the zlib/libpng license.

You can download the latest stable release as source or Windows binaries, or fetch the latest branch from GitHub. Each release starting with 3.0 also has a corresponding annotated tag with source and binary archives.

The documentation is available online and is included in all source and binary archives. See the release notes for new features, caveats and deprecations in the latest release. For more details see the version history.

The master branch is the stable integration branch and *should* always compile and run on all supported platforms, although details of newly added features may change until they have been included in a release. New features and many bug fixes live in other branches until they are stable enough to merge.

If you are new to GLFW, you may find the tutorial for GLFW 3 useful. If you have used GLFW 2 in the past, there is a transition guide for moving to the GLFW 3 API.

GLFW exists because of the contributions of many people around the world, whether by reporting bugs, providing community support, adding features, reviewing or testing code, debugging, proofreading docs, suggesting features or fixing bugs.

19.2 Compiling GLFW

GLFW itself requires only the headers and libraries for your OS and window system. It does not need the headers for any context creation API (WGL, GLX, EGL, NSGL, OSMesa) or rendering API (OpenGL, OpenGL ES, Vulkan) to enable support for them.

GLFW supports compilation on Windows with Visual C++ 2010 and later, MinGW and MinGW-w64, on macOS with Clang and on Linux and other Unix-like systems with GCC and Clang. It will likely compile in other environments as well, but this is not regularly tested.

There are `pre-compiled Windows binaries` available for all supported compilers.

See the `compilation guide` for more information about how to compile GLFW yourself.

19.3 Using GLFW

See the `documentation` for tutorials, guides and the API reference.

19.4 Contributing to GLFW

See the `contribution guide` for more information.

19.5 System requirements

GLFW supports Windows XP and later and macOS 10.8 and later. Linux and other Unix-like systems running the X Window System are supported even without a desktop environment or modern extensions, although some features require a running window or clipboard manager. The OSMesa backend requires Mesa 6.3.

See the `compatibility guide` in the documentation for more information.

19.6 Dependencies

GLFW itself needs only CMake 3.1 or later and the headers and libraries for your OS and window system.

The examples and test programs depend on a number of tiny libraries. These are located in the `deps/` directory.

- `getopt_port` for examples with command-line options
- `TinyCThread` for threaded examples
- `glad2` for loading OpenGL and Vulkan functions
- `linmath.h` for linear algebra in examples
- `Nuklear` for test and example UI
- `stb_image_write` for writing images to disk

The documentation is generated with `Doxygen` if CMake can find that tool.

19.7 Reporting bugs

Bugs are reported to our issue tracker. Please check the contribution guide for information on what to include when reporting a bug.

19.8 Changelog

- Added GLFW_PLATFORM init hint for runtime platform selection (#1958)
- Added GLFW_ANY_PLATFORM, GLFW_PLATFORM_WIN32, GLFW_PLATFORM_COCOA, GLFW_PLATFORM_WAYLAND, GLFW_PLATFORM_X11 and GLFW_PLATFORM_NULL symbols to specify the desired platform (#1958)
- Added glfwGetPlatform function to query what platform was selected (#1655,#1958)
- Added glfwPlatformSupported function to query if a platform is supported (#1655,#1958)
- Added glfwInitAllocator for setting a custom memory allocator (#544,#1628,#1947)
- Added GLFWallocator struct and GLFWallocatefun, GLFWreallocatefun and GLFWdeallocatefun types (#544,#1628,#1947)
- Added glfwInitVulkanLoader for using a non-default Vulkan loader (#1374,#1890)
- Added GLFW_RESIZE_NWSE_CURSOR, GLFW_RESIZE_NESW_CURSOR, GLFW_RESIZE_ALL_CURSOR and GLFW_NOT_ALLOWED_CURSOR cursor shapes (#427)
- Added GLFW_RESIZE_EW_CURSOR alias for GLFW_HRESIZE_CURSOR (#427)
- Added GLFW_RESIZE_NS_CURSOR alias for GLFW_VRESIZE_CURSOR (#427)
- Added GLFW_POINTING_HAND_CURSOR alias for GLFW_HAND_CURSOR (#427)
- Added GLFW_MOUSE_PASSTHROUGH window hint for letting mouse input pass through the window (#1236,#1568)
- Added GLFW_PLATFORM_UNAVAILABLE error for platform detection failures (#1958)
- Added GLFW_FEATURE_UNAVAILABLE error for platform limitations (#1692)
- Added GLFW_FEATURE_UNIMPLEMENTED error for incomplete backends (#1692)
- Added GLFW_ANGLE_PLATFORM_TYPE init hint and GLFW_ANGLE_PLATFORM_TYPE_* values to select ANGLE backend (#1380)
- Added GLFW_X11_XCB_VULKAN_SURFACE init hint for selecting X11 Vulkan surface extension (#1793)
- Added GLFW_BUILD_WIN32 CMake option for enabling Win32 support (#1958)
- Added GLFW_BUILD_COCOA CMake option for enabling Cocoa support (#1958)
- Added GLFW_BUILD_X11 CMake option for enabling X11 support (#1958)
- Added GLFW_LIBRARY_TYPE CMake variable for overriding the library type (#279,#1307,#1497,#1574,#1928)
- Added GLFW_PKG_CONFIG_REQUIRES_PRIVATE and GLFW_PKG_CONFIG_LIBS_PRIVATE CMake variables exposing pkg-config dependencies (#1307)
- Made joystick subsystem initialize at first use (#1284,#1646)
- Made GLFW_DOUBLEBUFFER a read-only window attribute
- Updated the minimum required CMake version to 3.1

- Updated gamepad mappings from upstream
- Disabled tests and examples by default when built as a CMake subdirectory
- Renamed `GLFW_USE_WAYLAND` CMake option to `GLFW_BUILD_WAYLAND` (#1958)
- Removed `GLFW_USE_OSMESA` CMake option enabling the Null platform (#1958)
- Removed CMake generated configuration header
- Bugfix: The CMake config-file package used an absolute path and was not relocatable (#1470)
- Bugfix: Video modes with a duplicate screen area were discarded (#1555,#1556)
- Bugfix: Compiling with -Wextra-semi caused warnings (#1440)
- Bugfix: Built-in mappings failed because some OEMs re-used VID/PID (#1583)
- Bugfix: Some extension loader headers did not prevent default OpenGL header inclusion (#1695)
- Bugfix: Buffers were swapped at creation on single-buffered windows (#1873)
- Bugfix: Gamepad mapping updates could spam `GLFW_INVALID_VALUE` due to incompatible controllers sharing hardware ID (#1763)
- Bugfix: Native access functions for context handles did not check that the API matched
- [Win32] Added the `GLFW_WIN32_KEYBOARD_MENU` window hint for enabling access to the window menu
- [Win32] Added a version info resource to the GLFW DLL
- [Win32] Disabled framebuffer transparency on Windows 7 when DWM windows are opaque (#1512)
- [Win32] Bugfix: `GLFW_INCLUDE_VULKAN` plus `VK_USE_PLATFORM_WIN32_KHR` caused symbol redefinition (#1524)
- [Win32] Bugfix: The cursor position event was emitted before its cursor enter event (#1490)
- [Win32] Bugfix: The window hint `GLFW_MAXIMIZED` did not move or resize the window (#1499)
- [Win32] Bugfix: Disabled cursor mode interfered with some non-client actions
- [Win32] Bugfix: Super key was not released after Win+V hotkey (#1622)
- [Win32] Bugfix: `glfwGetKeyName` could access out of bounds and return an invalid pointer
- [Win32] Bugfix: Some synthetic key events were reported as `GLFW_KEY_UNKNOWN` (#1623)
- [Win32] Bugfix: Non-BMP Unicode codepoint input was reported as UTF-16
- [Win32] Bugfix: Monitor functions could return invalid values after configuration change (#1761)
- [Win32] Bugfix: Initialization would segfault on Windows 8 (not 8.1) (#1775)
- [Win32] Bugfix: Duplicate size events were not filtered (#1610)
- [Win32] Bugfix: Full screen windows were incorrectly resized by DPI changes (#1582)
- [Win32] Bugfix: `GLFW_SCALE_TO_MONITOR` had no effect on systems older than Windows 10 version 1703 (#1511)
- [Win32] Bugfix: `USE_MSVC_RUNTIME_LIBRARY_DLL` had no effect on CMake 3.15 or later (#1783,#1796)
- [Win32] Bugfix: Compilation with LLVM for Windows failed (#1807,#1824,#1874)
- [Win32] Bugfix: The foreground lock timeout was overridden, ignoring the user
- [Win32] Bugfix: Content scale queries could fail silently (#1615)

- [Win32] Bugfix: Content scales could have garbage values if monitor was recently disconnected (#1615)
- [Win32] Bugfix: A window created maximized and undecorated would cover the whole monitor (#1806)
- [Win32] Bugfix: The default restored window position was lost when creating a maximized window
- [Win32] Bugfix: `glfwMaximizeWindow` would make a hidden window visible
- [Win32] Bugfix: `Alt+PrtSc` would emit `GLFW_KEY_UNKNOWN` and a different scancode than `PrtSc` (#1993)
- [Win32] Bugfix: `GLFW_KEY_PAUSE` scancode from `glfwGetKeyScancode` did not match event scancode (#1993)
- [Win32] Bugfix: Instance-local operations used executable instance (#469,#1296,#1395)
- [Cocoa] Added support for `VK_EXT_metal_surface` (#1619)
- [Cocoa] Added locating the Vulkan loader at runtime in an application bundle
- [Cocoa] Moved main menu creation to GLFW initialization time (#1649)
- [Cocoa] Changed `EGLNativeWindowType` from `NSView` to `CALayer` (#1169)
- [Cocoa] Changed F13 key to report Print Screen for cross-platform consistency (#1786)
- [Cocoa] Removed dependency on the CoreVideo framework
- [Cocoa] Bugfix: `glfwSetWindowSize` used a bottom-left anchor point (#1553)
- [Cocoa] Bugfix: Window remained on screen after destruction until event poll (#1412)
- [Cocoa] Bugfix: Event processing before window creation would assert (#1543)
- [Cocoa] Bugfix: Undecorated windows could not be iconified on recent macOS
- [Cocoa] Bugfix: Touching event queue from secondary thread before main thread would abort (#1649)
- [Cocoa] Bugfix: Non-BMP Unicode codepoint input was reported as UTF-16 (#1635)
- [Cocoa] Bugfix: Failing to retrieve the refresh rate of built-in displays could leak memory
- [Cocoa] Bugfix: Objective-C files were compiled as C with CMake 3.19 (#1787)
- [Cocoa] Bugfix: Duplicate video modes were not filtered out (#1830)
- [Cocoa] Bugfix: Menu bar was not clickable on macOS 10.15+ until it lost and regained focus (#1648,#1802)
- [Cocoa] Bugfix: Monitor name query could segfault on macOS 11 (#1809,#1833)
- [Cocoa] Bugfix: The install name of the installed dylib was relative (#1504)
- [Cocoa] Bugfix: The MoltenVK layer contents scale was updated only after related events were emitted
- [Cocoa] Bugfix: Moving the cursor programmatically would freeze it for a fraction of a second (#1962)
- [Cocoa] Bugfix: `kIOMasterPortDefault` was deprecated in macOS 12.0 (#1980)
- [Cocoa] Bugfix: `kUTTypeURL` was deprecated in macOS 12.0 (#2003)
- [Cocoa] Bugfix: A connected Apple AirPlay would emit a useless error (#1791)
- [X11] Bugfix: The CMake files did not check for the XInput headers (#1480)
- [X11] Bugfix: Key names were not updated when the keyboard layout changed (#1462,#1528)
- [X11] Bugfix: Decorations could not be enabled after window creation (#1566)
- [X11] Bugfix: Content scale fallback value could be inconsistent (#1578)
- [X11] Bugfix: `glfwMaximizeWindow` had no effect on hidden windows

- [X11] Bugfix: Clearing `GLFW_FLOATING` on a hidden window caused invalid read
- [X11] Bugfix: Changing `GLFW_FLOATING` on a hidden window could silently fail
- [X11] Bugfix: Disabled cursor mode was interrupted by indicator windows
- [X11] Bugfix: Monitor physical dimensions could be reported as zero mm
- [X11] Bugfix: Window position events were not emitted during resizing (#1613)
- [X11] Bugfix: `glfwFocusWindow` could terminate on older WMs or without a WM
- [X11] Bugfix: Querying a disconnected monitor could segfault (#1602)
- [X11] Bugfix: IME input of CJK was broken for "C" locale (#1587,#1636)
- [X11] Bugfix: Termination would segfault if the IM had been destroyed
- [X11] Bugfix: Any IM started after initialization would not be detected
- [X11] Bugfix: Xlib errors caused by other parts of the application could be reported as GLFW errors
- [X11] Bugfix: A handle race condition could cause a `BadWindow` error (#1633)
- [X11] Bugfix: XKB path used keysyms instead of physical locations for non-printable keys (#1598)
- [X11] Bugfix: Function keys were mapped to `GLFW_KEY_UNKNOWN` for some layout combinations (#1598)
- [X11] Bugfix: Keys pressed simultaneously with others were not always reported (#1112,#1415,#1472,#1616)
- [X11] Bugfix: Some window attributes were not applied on leaving fullscreen (#1863)
- [X11] Bugfix: Changing `GLFW_FLOATING` could leak memory
- [X11] Bugfix: Icon pixel format conversion worked only by accident, relying on undefined behavior (#1986)
- [X11] Bugfix: Dynamic loading on OpenBSD failed due to soname differences
- [X11] Bugfix: Waiting for events would fail if file descriptor was too large (#2024)
- [X11] Bugfix: Joystick events could lead to busy-waiting (#1872)
- [X11] Bugfix: `glfwWaitEvents*` did not continue for joystick events
- [X11] Bugfix: `glfwPostEmptyEvent` could be ignored due to race condition (#379,#1281,#1285,#2033)
- [X11] Bugfix: Dynamic loading on NetBSD failed due to soname differences
- [X11] Bugfix: Left shift of int constant relied on undefined behavior (#1951)
- [Wayland] Added dynamic loading of all Wayland libraries
- [Wayland] Added support for key names via xkbcommon
- [Wayland] Removed support for `wl_shell` (#1443)
- [Wayland] Bugfix: The `GLFW_HAND_CURSOR` shape used the wrong image (#1432)
- [Wayland] Bugfix: `CLOCK_MONOTONIC` was not correctly enabled
- [Wayland] Bugfix: Repeated keys could be reported with `NULL` window (#1704)
- [Wayland] Bugfix: Retrieving partial framebuffer size would segfault
- [Wayland] Bugfix: Scrolling offsets were inverted compared to other platforms (#1463)
- [Wayland] Bugfix: Client-Side Decorations were destroyed in the wrong order (#1798)
- [Wayland] Bugfix: Monitors physical size could report zero (#1784,#1792)
- [Wayland] Bugfix: Some keys were not repeating in Wayland (#1908)

- [Wayland] Bugfix: Non-arrow cursors are offset from the hotspot (#1706,#1899)
- [Wayland] Bugfix: The `O_CLOEXEC` flag was not defined on FreeBSD
- [Wayland] Bugfix: Key repeat could lead to a race condition (#1710)
- [Wayland] Bugfix: Activating a window would emit two input focus events
- [Wayland] Bugfix: Disable key repeat mechanism when window loses input focus
- [Wayland] Bugfix: Window hiding and showing did not work (#1492,#1731)
- [Wayland] Bugfix: A key being repeated was not released when window lost focus
- [Wayland] Bugfix: Showing a hidden window did not emit a window refresh event
- [Wayland] Bugfix: Full screen window creation did not ignore `GLFW_VISIBLE`
- [Wayland] Bugfix: Some keys were reported as wrong key or `GLFW_KEY_UNKNOWN`
- [Wayland] Bugfix: Text input did not repeat along with key repeat
- [Wayland] Bugfix: `glfwPostEmptyEvent` sometimes had no effect (#1520,#1521)
- [POSIX] Removed use of deprecated function `gettimeofday`
- [POSIX] Bugfix: `CLOCK_MONOTONIC` was not correctly tested for or enabled
- [WGL] Disabled the DWM swap interval hack for Windows 8 and later (#1072)
- [NSGL] Removed enforcement of forward-compatible flag for core contexts
- [NSGL] Bugfix: `GLFW_COCOA_RETINA_FRAMEBUFFER` had no effect on newer macOS versions (#1442)
- [NSGL] Bugfix: Workaround for swap interval on 10.14 broke on 10.12 (#1483)
- [NSGL] Bugfix: Defining `GL_SILENCE_DEPRECATION` externally caused a duplicate definition warning (#1840)
- [EGL] Added platform selection via the `EGL_EXT_platform_base` extension (#442)
- [EGL] Added ANGLE backend selection via `EGL_ANGLE_platform_angle` extension (#1380)
- [EGL] Bugfix: The `GLFW_DOUBLEBUFFER` context attribute was ignored (#1843)
- [GLX] Bugfix: Context creation failed if GLX 1.4 was not exported by GLX library

19.9 Contact

On `glfw.org` you can find the latest version of GLFW, as well as news, documentation and other information about the project.

If you have questions related to the use of GLFW, we have a `forum`, and the `#glfw` IRC channel on `Libera.Chat`.

If you have a bug to report, a patch to submit or a feature you'd like to request, please file it in the `issue tracker` on GitHub.

Finally, if you're interested in helping out with the development of GLFW or porting it to your favorite platform, join us on the forum, GitHub or IRC.

Chapter 20

Deprecated List

Member GLFWcharmodsfun)(GLFWwindow *window, unsigned int codepoint, int mods)

Scheduled for removal in version 4.0.

Member glfwSetCharModsCallback (GLFWwindow *window, GLFWcharmodsfun callback)

Scheduled for removal in version 4.0.

Chapter 21

Module Index

21.1 Modules

Here is a list of all modules:

Chapter 22

Hierarchical Index

22.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 23

Class Index

23.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 24

File Index

24.1 File List

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

Chapter 25

Module Documentation

25.1 Context reference

Functions and types related to OpenGL and OpenGL ES contexts.

Typedefs

- typedef void(∗ GLFWglproc) (void)

  Client API function pointer type.

Functions

- GLFWAPI void glfwMakeContextCurrent (GLFWwindow ∗window)

  Makes the context of the specified window current for the calling thread.
- GLFWAPI GLFWwindow ∗ glfwGetCurrentContext (void)

  Returns the window whose context is current on the calling thread.
- GLFWAPI void glfwSwapInterval (int interval)

  Sets the swap interval for the current context.
- GLFWAPI int glfwExtensionSupported (const char ∗extension)

  Returns whether the specified extension is available.
- GLFWAPI GLFWglproc glfwGetProcAddress (const char ∗procname)

  Returns the address of the specified function for the current context.

25.1.1 Detailed Description

Functions and types related to OpenGL and OpenGL ES contexts.

This is the reference documentation for OpenGL and OpenGL ES context related functions. For more task-oriented information, see the Context guide.

25.1.2 Typedef Documentation

25.1.2.1 GLFWglproc

```
typedef void(* GLFWglproc) (void)
```

Client API function pointer type.

Generic function pointer used for returning client API function pointers without forcing a cast from a regular pointer.

See also

OpenGL and OpenGL ES extensions

glfwGetProcAddress

Since

Added in version 3.0.

25.1.3 Function Documentation

25.1.3.1 glfwExtensionSupported()

```
GLFWAPI int glfwExtensionSupported (
            const char * extension )
```

Returns whether the specified extension is available.

This function returns whether the specified API extension is supported by the current OpenGL or OpenGL ES context. It searches both for client API extension and context creation API extensions.

A context must be current on the calling thread. Calling this function without a current context will cause a GLFW_NO_CURRENT_CONTEXT error.

As this functions retrieves and searches one or more extension strings each call, it is recommended that you cache its results if it is going to be used frequently. The extension strings will not change during the lifetime of a context, so there is no danger in doing this.

This function does not apply to Vulkan. If you are using Vulkan, see glfwGetRequiredInstanceExtensions, `vk↩EnumerateInstanceExtensionProperties` and `vkEnumerateDeviceExtensionProperties` instead.

Parameters

in	*extension*	The ASCII encoded name of the extension.

Returns

`GLFW_TRUE` if the extension is available, or `GLFW_FALSE` otherwise.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_NO_CURRENT_CONTEXT, GLFW_INVALID_VALUE and GLFW_PLATFORM_ERROR.

@thread_safety This function may be called from any thread.

See also

OpenGL and OpenGL ES extensions

glfwGetProcAddress

Since

Added in version 1.0.

25.1.3.2 glfwGetCurrentContext()

```
GLFWAPI GLFWwindow * glfwGetCurrentContext (
            void  )
```

Returns the window whose context is current on the calling thread.

This function returns the window whose OpenGL or OpenGL ES context is current on the calling thread.

Returns

The window whose context is current, or `NULL` if no window's context is current.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function may be called from any thread.

See also

Current context

glfwMakeContextCurrent

Since

Added in version 3.0.

25.1.3.3 glfwGetProcAddress()

```
GLFWAPI GLFWglproc glfwGetProcAddress (
            const char * procname )
```

Returns the address of the specified function for the current context.

This function returns the address of the specified OpenGL or OpenGL ES core or extension function, if it is supported by the current context.

A context must be current on the calling thread. Calling this function without a current context will cause a GLFW_NO_CURRENT_CONTEXT error.

This function does not apply to Vulkan. If you are rendering with Vulkan, see glfwGetInstanceProcAddress, `vk↩GetInstanceProcAddr` and `vkGetDeviceProcAddr` instead.

Parameters

in	*procname*	The ASCII encoded name of the function.

Returns

The address of the function, or `NULL` if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_NO_CURRENT_CONTEXT and GLFW_PLATFORM_ERROR.

Remarks

The address of a given function is not guaranteed to be the same between contexts.

This function may return a non-`NULL` address despite the associated version or extension not being available. Always check the context version or extension string first.

@pointer_lifetime The returned function pointer is valid until the context is destroyed or the library is terminated.

@thread_safety This function may be called from any thread.

See also

OpenGL and OpenGL ES extensions

glfwExtensionSupported

Since

Added in version 1.0.

25.1.3.4 glfwMakeContextCurrent()

```
GLFWAPI void glfwMakeContextCurrent (
            GLFWwindow * window )
```

Makes the context of the specified window current for the calling thread.

This function makes the OpenGL or OpenGL ES context of the specified window current on the calling thread. A context must only be made current on a single thread at a time and each thread can have only a single current context at a time.

When moving a context between threads, you must make it non-current on the old thread before making it current on the new one.

By default, making a context non-current implicitly forces a pipeline flush. On machines that support `GL↩_KHR_context_flush_control`, you can control whether a context performs this flush by setting the GLFW_CONTEXT_RELEASE_BEHAVIOR hint.

The specified window must have an OpenGL or OpenGL ES context. Specifying a window without a context will generate a GLFW_NO_WINDOW_CONTEXT error.

Parameters

in	*window*	The window whose context to make current, or `NULL` to detach the current context.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_NO_WINDOW_CONTEXT and GLFW_PLATFORM_ERROR.

@thread_safety This function may be called from any thread.

See also

Current context

glfwGetCurrentContext

Since

Added in version 3.0.

25.1.3.5 glfwSwapInterval()

```
GLFWAPI void glfwSwapInterval (
            int interval )
```

Sets the swap interval for the current context.

This function sets the swap interval for the current OpenGL or OpenGL ES context, i.e. the number of screen updates to wait from the time glfwSwapBuffers was called before swapping the buffers and returning. This is sometimes called *vertical synchronization*, *vertical retrace synchronization* or just *vsync*.

A context that supports either of the `WGL_EXT_swap_control_tear` and `GLX_EXT_swap_control_tear` extensions also accepts *negative* swap intervals, which allows the driver to swap immediately even if a frame arrives a little bit late. You can check for these extensions with glfwExtensionSupported.

A context must be current on the calling thread. Calling this function without a current context will cause a GLFW_NO_CURRENT_CONTEXT error.

This function does not apply to Vulkan. If you are rendering with Vulkan, see the present mode of your swapchain instead.

Parameters

in	*interval*	The minimum number of screen updates to wait for until the buffers are swapped by glfwSwapBuffers.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_NO_CURRENT_CONTEXT and GLFW_PLATFORM_ERROR.

Remarks

This function is not called during context creation, leaving the swap interval set to whatever is the default for that API. This is done because some swap interval extensions used by GLFW do not allow the swap interval to be reset to zero once it has been set to a non-zero value.

Some GPU drivers do not honor the requested swap interval, either because of a user setting that overrides the application's request or due to bugs in the driver.

@thread_safety This function may be called from any thread.

See also

Buffer swapping

glfwSwapBuffers

Since

Added in version 1.0.

25.2 Vulkan support reference

Functions and types related to Vulkan.

Typedefs

- typedef void(∗ GLFWvkproc) (void)

  Vulkan API function pointer type.

Functions

- GLFWAPI int glfwVulkanSupported (void)

  Returns whether the Vulkan loader and an ICD have been found.
- GLFWAPI const char ∗∗ glfwGetRequiredInstanceExtensions (uint32_t ∗count)

  Returns the Vulkan instance extensions required by GLFW.

25.2.1 Detailed Description

Functions and types related to Vulkan.

This is the reference documentation for Vulkan related functions and types. For more task-oriented information, see the Vulkan guide.

25.2.2 Typedef Documentation

25.2.2.1 GLFWvkproc

```
typedef void(* GLFWvkproc) (void)
```

Vulkan API function pointer type.

Generic function pointer used for returning Vulkan API function pointers without forcing a cast from a regular pointer.

See also

Querying Vulkan function pointers

glfwGetInstanceProcAddress

Since

Added in version 3.2.

25.2.3 Function Documentation

25.2.3.1 glfwGetRequiredInstanceExtensions()

```
GLFWAPI const char ** glfwGetRequiredInstanceExtensions (
            uint32_t * count )
```

Returns the Vulkan instance extensions required by GLFW.

This function returns an array of names of Vulkan instance extensions required by GLFW for creating Vulkan surfaces for GLFW windows. If successful, the list will always contain `VK_KHR_surface`, so if you don't require any additional extensions you can pass this list directly to the `VkInstanceCreateInfo` struct.

If Vulkan is not available on the machine, this function returns `NULL` and generates a GLFW_API_UNAVAILABLE error. Call glfwVulkanSupported to check whether Vulkan is at least minimally available.

If Vulkan is available but no set of extensions allowing window surface creation was found, this function returns `NULL`. You may still use Vulkan for off-screen rendering and compute work.

Parameters

`out`	*count*	Where to store the number of extensions in the returned array. This is set to zero if an error occurred.

Returns

An array of ASCII encoded extension names, or `NULL` if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_API_UNAVAILABLE.

Remarks

Additional extensions may be required by future versions of GLFW. You should check if any extensions you wish to enable are already in the returned array, as it is an error to specify an extension more than once in the `VkInstanceCreateInfo` struct.

@pointer_lifetime The returned array is allocated and freed by GLFW. You should not free it yourself. It is guaranteed to be valid only until the library is terminated.

@thread_safety This function may be called from any thread.

See also

Querying required Vulkan extensions

glfwCreateWindowSurface

Since

Added in version 3.2.

25.2.3.2 glfwVulkanSupported()

```
GLFWAPI int glfwVulkanSupported (
            void  )
```

Returns whether the Vulkan loader and an ICD have been found.

This function returns whether the Vulkan loader and any minimally functional ICD have been found.

The availability of a Vulkan loader and even an ICD does not by itself guarantee that surface creation or even instance creation is possible. Call glfwGetRequiredInstanceExtensions to check whether the extensions necessary for Vulkan surface creation are available and glfwGetPhysicalDevicePresentationSupport to check whether a queue family of a physical device supports image presentation.

Returns

`GLFW_TRUE` if Vulkan is minimally available, or `GLFW_FALSE` otherwise.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function may be called from any thread.

See also

Querying for Vulkan support

Since

Added in version 3.2.

25.3 Initialization, version and error reference

Functions and types related to initialization and error handling.

Modules

- Error codes

  Error codes.

Classes

- struct GLFWallocator

Macros

- #define GLFW_TRUE 1

  One.
- #define GLFW_FALSE 0

  Zero.
- #define GLFW_JOYSTICK_HAT_BUTTONS 0x00050001

  Joystick hat buttons init hint.
- #define GLFW_ANGLE_PLATFORM_TYPE 0x00050002

  ANGLE rendering backend init hint.
- #define GLFW_PLATFORM 0x00050003

  Platform selection init hint.
- #define GLFW_COCOA_CHDIR_RESOURCES 0x00051001

  macOS specific init hint.
- #define GLFW_COCOA_MENUBAR 0x00051002

  macOS specific init hint.
- #define GLFW_X11_XCB_VULKAN_SURFACE 0x00052001

  X11 specific init hint.
- #define GLFW_ANY_PLATFORM 0x00060000

  Hint value that enables automatic platform selection.
- #define **GLFW_PLATFORM_WIN32** 0x00060001
- #define **GLFW_PLATFORM_COCOA** 0x00060002
- #define **GLFW_PLATFORM_WAYLAND** 0x00060003
- #define **GLFW_PLATFORM_X11** 0x00060004
- #define **GLFW_PLATFORM_NULL** 0x00060005

Typedefs

- typedef void *(* GLFWallocatefun) (size_t size, void *user)

  The function pointer type for memory allocation callbacks.
- typedef void *(* GLFWreallocatefun) (void *block, size_t size, void *user)

  The function pointer type for memory reallocation callbacks.
- typedef void(* GLFWdeallocatefun) (void *block, void *user)

  The function pointer type for memory deallocation callbacks.
- typedef void(* GLFWerrorfun) (int error_code, const char *description)

  The function pointer type for error callbacks.
- typedef struct GLFWallocator GLFWallocator

Functions

- GLFWAPI int glfwInit (void)

  Initializes the GLFW library.
- GLFWAPI void glfwTerminate (void)

  Terminates the GLFW library.
- GLFWAPI void glfwInitHint (int hint, int value)

  Sets the specified init hint to the desired value.
- GLFWAPI void glfwInitAllocator (const GLFWallocator *allocator)

  Sets the init allocator to the desired value.
- GLFWAPI void glfwGetVersion (int *major, int *minor, int *rev)

  Retrieves the version of the GLFW library.
- GLFWAPI const char * glfwGetVersionString (void)

  Returns a string describing the compile-time configuration.
- GLFWAPI int glfwGetError (const char **description)

  Returns and clears the last error for the calling thread.
- GLFWAPI GLFWerrorfun glfwSetErrorCallback (GLFWerrorfun callback)

  Sets the error callback.
- GLFWAPI int glfwGetPlatform (void)

  Returns the currently selected platform.
- GLFWAPI int glfwPlatformSupported (int platform)

  Returns whether the library includes support for the specified platform.

GLFW version macros

- #define GLFW_VERSION_MAJOR 3

  The major version number of the GLFW header.
- #define GLFW_VERSION_MINOR 4

  The minor version number of the GLFW header.
- #define GLFW_VERSION_REVISION 0

  The revision number of the GLFW header.

25.3.1 Detailed Description

Functions and types related to initialization and error handling.

This is the reference documentation for initialization and termination of the library, version management and error handling. For more task-oriented information, see the Introduction to the API.

25.3.2 Macro Definition Documentation

25.3.2.1 GLFW_ANGLE_PLATFORM_TYPE

```
#define GLFW_ANGLE_PLATFORM_TYPE 0x00050002
```

ANGLE rendering backend init hint.

ANGLE rendering backend init hint.

25.3.2.2 GLFW_ANY_PLATFORM

```
#define GLFW_ANY_PLATFORM 0x00060000
```

Hint value that enables automatic platform selection.

Hint value for GLFW_PLATFORM that enables automatic platform selection.

25.3.2.3 GLFW_COCOA_CHDIR_RESOURCES

```
#define GLFW_COCOA_CHDIR_RESOURCES 0x00051001
```

macOS specific init hint.

macOS specific init hint.

25.3.2.4 GLFW_COCOA_MENUBAR

```
#define GLFW_COCOA_MENUBAR 0x00051002
```

macOS specific init hint.

macOS specific init hint.

25.3.2.5 GLFW_FALSE

```
#define GLFW_FALSE 0
```

Zero.

This is only semantic sugar for the number 0. You can instead use `0` or `false` or `_False` or `GL_FALSE` or `VK_FALSE` or anything else that is equal to zero.

25.3.2.6 GLFW_JOYSTICK_HAT_BUTTONS

```
#define GLFW_JOYSTICK_HAT_BUTTONS 0x00050001
```

Joystick hat buttons init hint.

Joystick hat buttons init hint.

25.3.2.7 GLFW_PLATFORM

```
#define GLFW_PLATFORM 0x00050003
```

Platform selection init hint.

Platform selection init hint.

25.3.2.8 GLFW_TRUE

```
#define GLFW_TRUE 1
```

One.

This is only semantic sugar for the number 1. You can instead use `1` or `true` or `_True` or `GL_TRUE` or `VK_TRUE` or anything else that is equal to one.

25.3.2.9 GLFW_VERSION_MAJOR

```
#define GLFW_VERSION_MAJOR 3
```

The major version number of the GLFW header.

The major version number of the GLFW header. This is incremented when the API is changed in non-compatible ways.

25.3.2.10 GLFW_VERSION_MINOR

```
#define GLFW_VERSION_MINOR 4
```

The minor version number of the GLFW header.

The minor version number of the GLFW header. This is incremented when features are added to the API but it remains backward-compatible.

25.3.2.11 GLFW_VERSION_REVISION

```
#define GLFW_VERSION_REVISION 0
```

The revision number of the GLFW header.

The revision number of the GLFW header. This is incremented when a bug fix release is made that does not contain any API changes.

25.3.2.12 GLFW_X11_XCB_VULKAN_SURFACE

```
#define GLFW_X11_XCB_VULKAN_SURFACE 0x00052001
```

X11 specific init hint.

X11 specific init hint.

25.3.3 Typedef Documentation

25.3.3.1 GLFWallocatefun

```
typedef void *(* GLFWallocatefun) (size_t size, void *user)
```

The function pointer type for memory allocation callbacks.

This is the function pointer type for memory allocation callbacks. A memory allocation callback function has the following signature:

```
void* function_name(size_t size, void* user)
```

This function must return either a memory block at least `size` bytes long, or `NULL` if allocation failed. Note that not all parts of GLFW handle allocation failures gracefully yet.

This function may be called during glfwInit but before the library is flagged as initialized, as well as during glfwTerminate after the library is no longer flagged as initialized.

Any memory allocated by this function will be deallocated during library termination or earlier.

The size will always be greater than zero. Allocations of size zero are filtered out before reaching the custom allocator.

Parameters

in	*size*	The minimum size, in bytes, of the memory block.
in	*user*	The user-defined pointer from the allocator.

Returns

The address of the newly allocated memory block, or `NULL` if an error occurred.

@pointer_lifetime The returned memory block must be valid at least until it is deallocated.

@reentrancy This function should not call any GLFW function.

@thread_safety This function may be called from any thread that calls GLFW functions.

See also

Custom heap memory allocator

GLFWallocator

Since

Added in version 3.4.

25.3.3.2 GLFWallocator

```
typedef struct GLFWallocator GLFWallocator
```

See also

Custom heap memory allocator

glfwInitAllocator

Since

Added in version 3.4.

25.3.3.3 GLFWdeallocatefun

```
typedef void(* GLFWdeallocatefun) (void *block, void *user)
```

The function pointer type for memory deallocation callbacks.

This is the function pointer type for memory deallocation callbacks. A memory deallocation callback function has the following signature:
```
void function_name(void* block, void* user)
```

This function may deallocate the specified memory block. This memory block will have been allocated with the same allocator.

This function may be called during glfwInit but before the library is flagged as initialized, as well as during glfwTerminate after the library is no longer flagged as initialized.

The block address will never be `NULL`. Deallocations of `NULL` are filtered out before reaching the custom allocator.

Parameters

in	*block*	The address of the memory block to deallocate.
in	*user*	The user-defined pointer from the allocator.

@pointer_lifetime The specified memory block will not be accessed by GLFW after this function is called.

@reentrancy This function should not call any GLFW function.

@thread_safety This function may be called from any thread that calls GLFW functions.

See also

Custom heap memory allocator

GLFWallocator

Since

Added in version 3.4.

25.3.3.4 GLFWerrorfun

```
typedef void(* GLFWerrorfun) (int error_code, const char *description)
```

The function pointer type for error callbacks.

This is the function pointer type for error callbacks. An error callback function has the following signature:
```
void callback_name(int error_code, const char* description)
```

Parameters

in	*error_code*	An error code. Future releases may add more error codes.
in	*description*	A UTF-8 encoded string describing the error.

@pointer_lifetime The error description string is valid until the callback function returns.

See also

Error handling

glfwSetErrorCallback

Since

Added in version 3.0.

25.3.3.5 GLFWreallocatefun

```
typedef void *(* GLFWreallocatefun) (void *block, size_t size, void *user)
```

The function pointer type for memory reallocation callbacks.

This is the function pointer type for memory reallocation callbacks. A memory reallocation callback function has the following signature:

```
void* function_name(void* block, size_t size, void* user)
```

This function must return a memory block at least `size` bytes long, or `NULL` if allocation failed. Note that not all parts of GLFW handle allocation failures gracefully yet.

This function may be called during glfwInit but before the library is flagged as initialized, as well as during glfwTerminate after the library is no longer flagged as initialized.

Any memory allocated by this function will be deallocated during library termination or earlier.

The block address will never be `NULL` and the size will always be greater than zero. Reallocations of a block to size zero are converted into deallocations. Reallocations of `NULL` to a non-zero size are converted into regular allocations.

Parameters

`in`	*block*	The address of the memory block to reallocate.
`in`	*size*	The new minimum size, in bytes, of the memory block.
`in`	*user*	The user-defined pointer from the allocator.

Returns

The address of the newly allocated or resized memory block, or `NULL` if an error occurred.

@pointer_lifetime The returned memory block must be valid at least until it is deallocated.

@reentrancy This function should not call any GLFW function.

@thread_safety This function may be called from any thread that calls GLFW functions.

See also

Custom heap memory allocator

GLFWallocator

Since

Added in version 3.4.

25.3.4 Function Documentation

25.3.4.1 glfwGetError()

```
GLFWAPI int glfwGetError (
            const char ** description )
```

Returns and clears the last error for the calling thread.

This function returns and clears the error code of the last error that occurred on the calling thread, and optionally a UTF-8 encoded human-readable description of it. If no error has occurred since the last call, it returns GLFW_NO_ERROR (zero) and the description pointer is set to `NULL`.

Parameters

in	*description*	Where to store the error description pointer, or `NULL`.

Returns

The last error code for the calling thread, or GLFW_NO_ERROR (zero).

@errors None.

@pointer_lifetime The returned string is allocated and freed by GLFW. You should not free it yourself. It is guaranteed to be valid only until the next error occurs or the library is terminated.

Remarks

This function may be called before glfwInit.

@thread_safety This function may be called from any thread.

See also

Error handling

glfwSetErrorCallback

Since

Added in version 3.3.

25.3.4.2 glfwGetPlatform()

```
GLFWAPI int glfwGetPlatform (
            void  )
```

Returns the currently selected platform.

This function returns the platform that was selected during initialization. The returned value will be one of `GLFW_PLATFORM_WIN32`, `GLFW_PLATFORM_COCOA`, `GLFW_PLATFORM_WAYLAND`, `GLFW_PLATFORM_X11` or `GLFW_PLATFORM_NULL`.

Returns

The currently selected platform, or zero if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function may be called from any thread.

See also

Runtime platform selection

glfwPlatformSupported

Since

Added in version 3.4.

25.3.4.3 glfwGetVersion()

```
GLFWAPI void glfwGetVersion (
             int * major,
             int * minor,
             int * rev )
```

Retrieves the version of the GLFW library.

This function retrieves the major, minor and revision numbers of the GLFW library. It is intended for when you are using GLFW as a shared library and want to ensure that you are using the minimum required version.

Any or all of the version arguments may be `NULL`.

Parameters

out	*major*	Where to store the major version number, or `NULL`.
out	*minor*	Where to store the minor version number, or `NULL`.
out	*rev*	Where to store the revision number, or `NULL`.

@errors None.

Remarks

This function may be called before glfwInit.

@thread_safety This function may be called from any thread.

See also

Version management

glfwGetVersionString

Since

Added in version 1.0.

25.3.4.4 glfwGetVersionString()

```
GLFWAPI const char * glfwGetVersionString (
            void  )
```

Returns a string describing the compile-time configuration.

This function returns the compile-time generated version string of the GLFW library binary. It describes the version, platforms, compiler and any platform or operating system specific compile-time options. It should not be confused with the OpenGL or OpenGL ES version string, queried with `glGetString`.

Do not use the version string to parse the GLFW library version. The glfwGetVersion function provides the version of the running library binary in numerical format.

Do not use the version string to parse what platforms are supported. The glfwPlatformSupported function lets you query platform support.

Returns

The ASCII encoded GLFW version string.

@errors None.

Remarks

This function may be called before glfwInit.

@pointer_lifetime The returned string is static and compile-time generated.

@thread_safety This function may be called from any thread.

See also

Version management

glfwGetVersion

Since

Added in version 3.0.

25.3.4.5 glfwInit()

```
GLFWAPI int glfwInit (
            void  )
```

Initializes the GLFW library.

This function initializes the GLFW library. Before most GLFW functions can be used, GLFW must be initialized, and before an application terminates GLFW should be terminated in order to free any resources allocated during or after initialization.

If this function fails, it calls glfwTerminate before returning. If it succeeds, you should call glfwTerminate before the application exits.

Additional calls to this function after successful initialization but before termination will return `GLFW_TRUE` immediately.

The GLFW_PLATFORM init hint controls which platforms are considered during initialization. This also depends on which platforms the library was compiled to support.

Returns

`GLFW_TRUE` if successful, or `GLFW_FALSE` if an error occurred.

@errors Possible errors include GLFW_PLATFORM_UNAVAILABLE and GLFW_PLATFORM_ERROR.

Remarks

@macos This function will change the current directory of the application to the `Contents/Resources` subdirectory of the application's bundle, if present. This can be disabled with the GLFW_COCOA_CHDIR_RESOURCES init hint.

@macos This function will create the main menu and dock icon for the application. If GLFW finds a `MainMenu.nib` it is loaded and assumed to contain a menu bar. Otherwise a minimal menu bar is created manually with common commands like Hide, Quit and About. The About entry opens a minimal about dialog with information from the application's bundle. The menu bar and dock icon can be disabled entirely with the GLFW_COCOA_MENUBAR init hint.

@x11 This function will set the `LC_CTYPE` category of the application locale according to the current environment if that category is still "C". This is because the "C" locale breaks Unicode text input.

@thread_safety This function must only be called from the main thread.

See also

Initialization and termination

glfwInitHint

glfwInitAllocator

glfwTerminate

Since

Added in version 1.0.

25.3.4.6 glfwInitAllocator()

```
GLFWAPI void glfwInitAllocator (
            const GLFWallocator * allocator )
```

Sets the init allocator to the desired value.

To use the default allocator, call this function with a `NULL` argument.

If you specify an allocator struct, every member must be a valid function pointer. If any member is `NULL`, this function emits GLFW_INVALID_VALUE and the init allocator is unchanged.

Parameters

in	*allocator*	The allocator to use at the next initialization, or `NULL` to use the default one.

@errors Possible errors include GLFW_INVALID_VALUE.

@pointer_lifetime The specified allocator is copied before this function returns.

@thread_safety This function must only be called from the main thread.

See also

Custom heap memory allocator

glfwInit

Since

Added in version 3.4.

25.3.4.7 glfwInitHint()

```
GLFWAPI void glfwInitHint (
            int hint,
            int value )
```

Sets the specified init hint to the desired value.

This function sets hints for the next initialization of GLFW.

The values you set hints to are never reset by GLFW, but they only take effect during initialization. Once GLFW has been initialized, any values you set will be ignored until the library is terminated and initialized again.

Some hints are platform specific. These may be set on any platform but they will only affect their specific platform. Other platforms will ignore them. Setting these hints requires no platform specific headers or functions.

Parameters

in	*hint*	The init hint to set.
in	*value*	The new value of the init hint.

@errors Possible errors include GLFW_INVALID_ENUM and GLFW_INVALID_VALUE.

Remarks

This function may be called before glfwInit.

@thread_safety This function must only be called from the main thread.

See also

init_hints

glfwInit

Since

Added in version 3.3.

25.3.4.8 glfwPlatformSupported()

```
GLFWAPI int glfwPlatformSupported (
            int platform )
```

Returns whether the library includes support for the specified platform.

This function returns whether the library was compiled with support for the specified platform. The platform must be one of `GLFW_PLATFORM_WIN32`, `GLFW_PLATFORM_COCOA`, `GLFW_PLATFORM_WAYLAND`, `GLFW_PLATFORM_X11` or `GLFW_PLATFORM_NULL`.

Parameters

in	*platform*	The platform to query.

Returns

`GLFW_TRUE` if the platform is supported, or `GLFW_FALSE` otherwise.

@errors Possible errors include GLFW_INVALID_ENUM.

Remarks

This function may be called before glfwInit.

@thread_safety This function may be called from any thread.

See also

Runtime platform selection

glfwGetPlatform

Since

Added in version 3.4.

25.3.4.9 glfwSetErrorCallback()

```
GLFWAPI GLFWerrorfun glfwSetErrorCallback (
            GLFWerrorfun callback )
```

Sets the error callback.

This function sets the error callback, which is called with an error code and a human-readable description each time a GLFW error occurs.

The error code is set before the callback is called. Calling glfwGetError from the error callback will return the same value as the error code argument.

The error callback is called on the thread where the error occurred. If you are using GLFW from multiple threads, your error callback needs to be written accordingly.

Because the description string may have been generated specifically for that error, it is not guaranteed to be valid after the callback has returned. If you wish to use it after the callback returns, you need to make a copy.

Once set, the error callback remains set even after the library has been terminated.

Parameters

in	*callback*	The new callback, or NULL to remove the currently set callback.

Returns

The previously set callback, or NULL if no callback was set.

@callback_signature
```
void callback_name(int error_code, const char* description)
```

For more information about the callback parameters, see the callback pointer type.

@errors None.

Remarks

This function may be called before glfwInit.

@thread_safety This function must only be called from the main thread.

See also

Error handling

glfwGetError

Since

Added in version 3.0.

25.3.4.10 glfwTerminate()

```
GLFWAPI void glfwTerminate (
            void  )
```

Terminates the GLFW library.

This function destroys all remaining windows and cursors, restores any modified gamma ramps and frees any other allocated resources. Once this function is called, you must again call glfwInit successfully before you will be able to use most GLFW functions.

If GLFW has been successfully initialized, this function should be called before the application exits. If initialization fails, there is no need to call this function, as it is called by glfwInit before it returns failure.

This function has no effect if GLFW is not initialized.

@errors Possible errors include GLFW_PLATFORM_ERROR.

Remarks

This function may be called before glfwInit.

Warning

The contexts of any remaining windows must not be current on any other thread when this function is called.

@reentrancy This function must not be called from a callback.

@thread_safety This function must only be called from the main thread.

See also

Initialization and termination

glfwInit

Since

Added in version 1.0.

25.4 Input reference

Functions and types related to input handling.

Modules

- Joystick hat states

  Joystick hat states.
- Keyboard keys

  Keyboard key IDs.
- Modifier key flags

  Modifier key flags.
- Mouse buttons

  Mouse button IDs.
- Joysticks

  Joystick IDs.
- Gamepad buttons

  Gamepad buttons.
- Gamepad axes

  Gamepad axes.
- Standard cursor shapes

  Standard system cursor shapes.

Classes

- struct GLFWgamepadstate

  Gamepad input state.

Typedefs

- typedef struct GLFWcursor GLFWcursor

  Opaque cursor object.
- typedef void(∗ GLFWmousebuttonfun) (GLFWwindow ∗window, int button, int action, int mods)

  The function pointer type for mouse button callbacks.
- typedef void(∗ GLFWcursorposfun) (GLFWwindow ∗window, double xpos, double ypos)

  The function pointer type for cursor position callbacks.
- typedef void(∗ GLFWcursorenterfun) (GLFWwindow ∗window, int entered)

  The function pointer type for cursor enter/leave callbacks.
- typedef void(∗ GLFWscrollfun) (GLFWwindow ∗window, double xoffset, double yoffset)

  The function pointer type for scroll callbacks.
- typedef void(∗ GLFWkeyfun) (GLFWwindow ∗window, int key, int scancode, int action, int mods)

  The function pointer type for keyboard key callbacks.
- typedef void(∗ GLFWcharfun) (GLFWwindow ∗window, unsigned int codepoint)

  The function pointer type for Unicode character callbacks.
- typedef void(∗ GLFWcharmodsfun) (GLFWwindow ∗window, unsigned int codepoint, int mods)

  The function pointer type for Unicode character with modifiers callbacks.
- typedef void(∗ GLFWdropfun) (GLFWwindow ∗window, int path_count, const char ∗paths[ ])

  The function pointer type for path drop callbacks.
- typedef void(∗ GLFWjoystickfun) (int jid, int event)

  The function pointer type for joystick configuration callbacks.
- typedef struct GLFWgamepadstate GLFWgamepadstate

  Gamepad input state.

Functions

- GLFWAPI int glfwGetInputMode (GLFWwindow *window, int mode)

  Returns the value of an input option for the specified window.
- GLFWAPI void glfwSetInputMode (GLFWwindow *window, int mode, int value)

  Sets an input option for the specified window.
- GLFWAPI int glfwRawMouseMotionSupported (void)

  Returns whether raw mouse motion is supported.
- GLFWAPI const char * glfwGetKeyName (int key, int scancode)

  Returns the layout-specific name of the specified printable key.
- GLFWAPI int glfwGetKeyScancode (int key)

  Returns the platform-specific scancode of the specified key.
- GLFWAPI int glfwGetKey (GLFWwindow *window, int key)

  Returns the last reported state of a keyboard key for the specified window.
- GLFWAPI int glfwGetMouseButton (GLFWwindow *window, int button)

  Returns the last reported state of a mouse button for the specified window.
- GLFWAPI void glfwGetCursorPos (GLFWwindow *window, double *xpos, double *ypos)

  Retrieves the position of the cursor relative to the content area of the window.
- GLFWAPI void glfwSetCursorPos (GLFWwindow *window, double xpos, double ypos)

  Sets the position of the cursor, relative to the content area of the window.
- GLFWAPI GLFWcursor * glfwCreateCursor (const GLFWimage *image, int xhot, int yhot)

  Creates a custom cursor.
- GLFWAPI GLFWcursor * glfwCreateStandardCursor (int shape)

  Creates a cursor with a standard shape.
- GLFWAPI void glfwDestroyCursor (GLFWcursor *cursor)

  Destroys a cursor.
- GLFWAPI void glfwSetCursor (GLFWwindow *window, GLFWcursor *cursor)

  Sets the cursor for the window.
- GLFWAPI GLFWkeyfun glfwSetKeyCallback (GLFWwindow *window, GLFWkeyfun callback)

  Sets the key callback.
- GLFWAPI GLFWcharfun glfwSetCharCallback (GLFWwindow *window, GLFWcharfun callback)

  Sets the Unicode character callback.
- GLFWAPI GLFWcharmodsfun glfwSetCharModsCallback (GLFWwindow *window, GLFWcharmodsfun callback)

  Sets the Unicode character with modifiers callback.
- GLFWAPI GLFWmousebuttonfun glfwSetMouseButtonCallback (GLFWwindow *window, GLFWmousebuttonfun callback)

  Sets the mouse button callback.
- GLFWAPI GLFWcursorposfun glfwSetCursorPosCallback (GLFWwindow *window, GLFWcursorposfun callback)

  Sets the cursor position callback.
- GLFWAPI GLFWcursorenterfun glfwSetCursorEnterCallback (GLFWwindow *window, GLFWcursorenterfun callback)

  Sets the cursor enter/leave callback.
- GLFWAPI GLFWscrollfun glfwSetScrollCallback (GLFWwindow *window, GLFWscrollfun callback)

  Sets the scroll callback.
- GLFWAPI GLFWdropfun glfwSetDropCallback (GLFWwindow *window, GLFWdropfun callback)

  Sets the path drop callback.
- GLFWAPI int glfwJoystickPresent (int jid)

  Returns whether the specified joystick is present.
- GLFWAPI const float * glfwGetJoystickAxes (int jid, int *count)

Returns the values of all axes of the specified joystick.

- GLFWAPI const unsigned char * glfwGetJoystickButtons (int jid, int *count)

  Returns the state of all buttons of the specified joystick.
- GLFWAPI const unsigned char * glfwGetJoystickHats (int jid, int *count)

  Returns the state of all hats of the specified joystick.
- GLFWAPI const char * glfwGetJoystickName (int jid)

  Returns the name of the specified joystick.
- GLFWAPI const char * glfwGetJoystickGUID (int jid)

  Returns the SDL compatible GUID of the specified joystick.
- GLFWAPI void glfwSetJoystickUserPointer (int jid, void *pointer)

  Sets the user pointer of the specified joystick.
- GLFWAPI void * glfwGetJoystickUserPointer (int jid)

  Returns the user pointer of the specified joystick.
- GLFWAPI int glfwJoystickIsGamepad (int jid)

  Returns whether the specified joystick has a gamepad mapping.
- GLFWAPI GLFWjoystickfun glfwSetJoystickCallback (GLFWjoystickfun callback)

  Sets the joystick configuration callback.
- GLFWAPI int glfwUpdateGamepadMappings (const char *string)

  Adds the specified SDL_GameControllerDB gamepad mappings.
- GLFWAPI const char * glfwGetGamepadName (int jid)

  Returns the human-readable gamepad name for the specified joystick.
- GLFWAPI int glfwGetGamepadState (int jid, GLFWgamepadstate *state)

  Retrieves the state of the specified joystick remapped as a gamepad.
- GLFWAPI void glfwSetClipboardString (GLFWwindow *window, const char *string)

  Sets the clipboard to the specified string.
- GLFWAPI const char * glfwGetClipboardString (GLFWwindow *window)

  Returns the contents of the clipboard as a string.
- GLFWAPI double glfwGetTime (void)

  Returns the GLFW time.
- GLFWAPI void glfwSetTime (double time)

  Sets the GLFW time.
- GLFWAPI uint64_t glfwGetTimerValue (void)

  Returns the current value of the raw timer.
- GLFWAPI uint64_t glfwGetTimerFrequency (void)

  Returns the frequency, in Hz, of the raw timer.

Key and button actions

- #define GLFW_RELEASE 0

  The key or mouse button was released.
- #define GLFW_PRESS 1

  The key or mouse button was pressed.
- #define GLFW_REPEAT 2

  The key was held down until it repeated.

25.4.1 Detailed Description

Functions and types related to input handling.

This is the reference documentation for input related functions and types. For more task-oriented information, see the Input guide.

25.4.2 Macro Definition Documentation

25.4.2.1 GLFW_PRESS

```
#define GLFW_PRESS 1
```

The key or mouse button was pressed.

The key or mouse button was pressed.

25.4.2.2 GLFW_RELEASE

```
#define GLFW_RELEASE 0
```

The key or mouse button was released.

The key or mouse button was released.

25.4.2.3 GLFW_REPEAT

```
#define GLFW_REPEAT 2
```

The key was held down until it repeated.

The key was held down until it repeated.

25.4.3 Typedef Documentation

25.4.3.1 GLFWcharfun

```
typedef void(* GLFWcharfun) (GLFWwindow *window, unsigned int codepoint)
```

The function pointer type for Unicode character callbacks.

This is the function pointer type for Unicode character callbacks. A Unicode character callback function has the following signature:

```
void function_name(GLFWwindow* window, unsigned int codepoint)
```

Parameters

in	*window*	The window that received the event.
in	*codepoint*	The Unicode code point of the character.

See also

Text input

glfwSetCharCallback

Since

Added in version 2.4. @glfw3 Added window handle parameter.

25.4.3.2 GLFWcharmodsfun

```
typedef void(* GLFWcharmodsfun) (GLFWwindow *window, unsigned int codepoint, int mods)
```

The function pointer type for Unicode character with modifiers callbacks.

This is the function pointer type for Unicode character with modifiers callbacks. It is called for each input character, regardless of what modifier keys are held down. A Unicode character with modifiers callback function has the following signature:

```
void function_name(GLFWwindow* window, unsigned int codepoint, int mods)
```

Parameters

in	*window*	The window that received the event.
in	*codepoint*	The Unicode code point of the character.
in	*mods*	Bit field describing which modifier keys were held down.

See also

Text input

glfwSetCharModsCallback

Deprecated Scheduled for removal in version 4.0.

Since

Added in version 3.1.

25.4.3.3 GLFWcursor

```
typedef struct GLFWcursor GLFWcursor
```

Opaque cursor object.

Opaque cursor object.

See also

Cursor objects

Since

Added in version 3.1.

25.4.3.4 GLFWcursorenterfun

```
typedef void(* GLFWcursorenterfun) (GLFWwindow *window, int entered)
```

The function pointer type for cursor enter/leave callbacks.

This is the function pointer type for cursor enter/leave callbacks. A cursor enter/leave callback function has the following signature:
```
void function_name(GLFWwindow* window, int entered)
```

Parameters

in	*window*	The window that received the event.
in	*entered*	GLFW_TRUE if the cursor entered the window's content area, or GLFW_FALSE if it left it.

See also

Cursor enter/leave events

glfwSetCursorEnterCallback

Since

Added in version 3.0.

25.4.3.5 GLFWcursorposfun

```
typedef void(* GLFWcursorposfun) (GLFWwindow *window, double xpos, double ypos)
```

The function pointer type for cursor position callbacks.

This is the function pointer type for cursor position callbacks. A cursor position callback function has the following signature:
```
void function_name(GLFWwindow* window, double xpos, double ypos);
```

Parameters

in	*window*	The window that received the event.
in	*xpos*	The new cursor x-coordinate, relative to the left edge of the content area.
in	*ypos*	The new cursor y-coordinate, relative to the top edge of the content area.

See also

Cursor position

glfwSetCursorPosCallback

Since

Added in version 3.0. Replaces GLFWmouseposfun.

25.4.3.6 GLFWdropfun

```
typedef void(* GLFWdropfun) (GLFWwindow *window, int path_count, const char *paths[])
```

The function pointer type for path drop callbacks.

This is the function pointer type for path drop callbacks. A path drop callback function has the following signature:
```
void function_name(GLFWwindow* window, int path_count, const char* paths[])
```

Parameters

in	*window*	The window that received the event.
in	*path_count*	The number of dropped paths.
in	*paths*	The UTF-8 encoded file and/or directory path names.

@pointer_lifetime The path array and its strings are valid until the callback function returns.

See also

Path drop input

glfwSetDropCallback

Since

Added in version 3.1.

25.4.3.7 GLFWgamepadstate

```
typedef struct GLFWgamepadstate GLFWgamepadstate
```

Gamepad input state.

This describes the input state of a gamepad.

See also

Gamepad input

glfwGetGamepadState

Since

Added in version 3.3.

25.4.3.8 GLFWjoystickfun

```
typedef void(* GLFWjoystickfun) (int jid, int event)
```

The function pointer type for joystick configuration callbacks.

This is the function pointer type for joystick configuration callbacks. A joystick configuration callback function has the following signature:
```
void function_name(int jid, int event)
```

Parameters

in	*jid*	The joystick that was connected or disconnected.
in	*event*	One of GLFW_CONNECTED or GLFW_DISCONNECTED. Future releases may add more events.

See also

Joystick configuration changes

glfwSetJoystickCallback

Since

Added in version 3.2.

25.4.3.9 GLFWkeyfun

```
typedef void(* GLFWkeyfun) (GLFWwindow *window, int key, int scancode, int action, int mods)
```

The function pointer type for keyboard key callbacks.

This is the function pointer type for keyboard key callbacks. A keyboard key callback function has the following signature:

```
void function_name(GLFWwindow* window, int key, int scancode, int action, int mods)
```

Parameters

in	*window*	The window that received the event.
in	*key*	The keyboard key that was pressed or released.
in	*scancode*	The platform-specific scancode of the key.
in	*action*	GLFW_PRESS, GLFW_RELEASE or GLFW_REPEAT. Future releases may add more actions.
in	*mods*	Bit field describing which modifier keys were held down.

See also

Key input

glfwSetKeyCallback

Since

Added in version 1.0. @glfw3 Added window handle, scancode and modifier mask parameters.

25.4.3.10 GLFWmousebuttonfun

```
typedef void(* GLFWmousebuttonfun) (GLFWwindow *window, int button, int action, int mods)
```

The function pointer type for mouse button callbacks.

This is the function pointer type for mouse button callback functions. A mouse button callback function has the following signature:
```
void function_name(GLFWwindow* window, int button, int action, int mods)
```

Parameters

in	*window*	The window that received the event.
in	*button*	The mouse button that was pressed or released.
in	*action*	One of `GLFW_PRESS` or `GLFW_RELEASE`. Future releases may add more actions.
in	*mods*	Bit field describing which modifier keys were held down.

See also

Mouse button input

glfwSetMouseButtonCallback

Since

Added in version 1.0. @glfw3 Added window handle and modifier mask parameters.

25.4.3.11 GLFWscrollfun

```
typedef void(* GLFWscrollfun) (GLFWwindow *window, double xoffset, double yoffset)
```

The function pointer type for scroll callbacks.

This is the function pointer type for scroll callbacks. A scroll callback function has the following signature:
```
void function_name(GLFWwindow* window, double xoffset, double yoffset)
```

Parameters

in	*window*	The window that received the event.
in	*xoffset*	The scroll offset along the x-axis.
in	*yoffset*	The scroll offset along the y-axis.

See also

Scroll input

glfwSetScrollCallback

Since

Added in version 3.0. Replaces `GLFWmousewheelfun`.

25.4.4 Function Documentation

25.4.4.1 glfwCreateCursor()

```
GLFWAPI GLFWcursor * glfwCreateCursor (
            const GLFWimage * image,
            int xhot,
            int yhot )
```

Creates a custom cursor.

Creates a new custom cursor image that can be set for a window with glfwSetCursor. The cursor can be destroyed with glfwDestroyCursor. Any remaining cursors are destroyed by glfwTerminate.

The pixels are 32-bit, little-endian, non-premultiplied RGBA, i.e. eight bits per channel with the red channel first. They are arranged canonically as packed sequential rows, starting from the top-left corner.

The cursor hotspot is specified in pixels, relative to the upper-left corner of the cursor image. Like all other coordinate systems in GLFW, the X-axis points to the right and the Y-axis points down.

Parameters

in	*image*	The desired cursor image.
in	*xhot*	The desired x-coordinate, in pixels, of the cursor hotspot.
in	*yhot*	The desired y-coordinate, in pixels, of the cursor hotspot.

Returns

The handle of the created cursor, or `NULL` if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_INVALID_VALUE and GLFW_PLATFORM_ERROR.

@pointer_lifetime The specified image data is copied before this function returns.

@thread_safety This function must only be called from the main thread.

See also

Cursor objects

glfwDestroyCursor

glfwCreateStandardCursor

Since

Added in version 3.1.

25.4.4.2 glfwCreateStandardCursor()

```
GLFWAPI GLFWcursor * glfwCreateStandardCursor (
            int shape )
```

Creates a cursor with a standard shape.

Returns a cursor with a standard shape, that can be set for a window with glfwSetCursor. The images for these cursors come from the system cursor theme and their exact appearance will vary between platforms.

Most of these shapes are guaranteed to exist on every supported platform but a few may not be present. See the table below for details.

Cursor shape	Windows	macOS	X11	Wayland
GLFW_ARROW_CURSOR	Yes	Yes	Yes	Yes
GLFW_IBEAM_CURSOR	Yes	Yes	Yes	Yes
GLFW_CROSSHAIR_CURSOR	Yes	Yes	Yes	Yes
GLFW_POINTING_HAND_CURSOR	Yes	Yes	Yes	Yes
GLFW_RESIZE_EW_CURSOR	Yes	Yes	Yes	Yes
GLFW_RESIZE_NS_CURSOR	Yes	Yes	Yes	Yes
GLFW_RESIZE_NWSE_CURSOR	Yes	Yes[1]	Maybe[2]	Maybe[2]
GLFW_RESIZE_NESW_CURSOR	Yes	Yes[1]	Maybe[2]	Maybe[2]
GLFW_RESIZE_ALL_CURSOR	Yes	Yes	Yes	Yes
GLFW_NOT_ALLOWED_CURSOR	Yes	Yes	Maybe[2]	Maybe[2]

1) This uses a private system API and may fail in the future.

2) This uses a newer standard that not all cursor themes support.

If the requested shape is not available, this function emits a GLFW_CURSOR_UNAVAILABLE error and returns NULL.

Parameters

in	*shape*	One of the standard shapes.

Returns

A new cursor ready to use or NULL if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_INVALID_ENUM, GLFW_CURSOR_UNAVAILABLE and GLFW_PLATFORM_ERROR.

@thread_safety This function must only be called from the main thread.

See also

Standard cursor creation

glfwCreateCursor

Since

Added in version 3.1.

25.4.4.3 glfwDestroyCursor()

```
GLFWAPI void glfwDestroyCursor (
            GLFWcursor * cursor )
```

Destroys a cursor.

This function destroys a cursor previously created with glfwCreateCursor. Any remaining cursors will be destroyed by glfwTerminate.

If the specified cursor is current for any window, that window will be reverted to the default cursor. This does not affect the cursor mode.

Parameters

in	*cursor*	The cursor object to destroy.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@reentrancy This function must not be called from a callback.

@thread_safety This function must only be called from the main thread.

See also

Cursor objects

glfwCreateCursor

Since

Added in version 3.1.

25.4.4.4 glfwGetClipboardString()

```
GLFWAPI const char * glfwGetClipboardString (
            GLFWwindow * window )
```

Returns the contents of the clipboard as a string.

This function returns the contents of the system clipboard, if it contains or is convertible to a UTF-8 encoded string. If the clipboard is empty or if its contents cannot be converted, `NULL` is returned and a GLFW_FORMAT_UNAVAILABLE error is generated.

Parameters

in	*window*	Deprecated. Any valid window or `NULL`.

Returns

The contents of the clipboard as a UTF-8 encoded string, or NULL if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_FORMAT_UNAVAILABLE and GLFW_PLATFORM_ERROR.

@pointer_lifetime The returned string is allocated and freed by GLFW. You should not free it yourself. It is valid until the next call to glfwGetClipboardString or glfwSetClipboardString, or until the library is terminated.

@thread_safety This function must only be called from the main thread.

See also

Clipboard input and output

glfwSetClipboardString

Since

Added in version 3.0.

25.4.4.5 glfwGetCursorPos()

```
GLFWAPI void glfwGetCursorPos (
            GLFWwindow * window,
            double * xpos,
            double * ypos )
```

Retrieves the position of the cursor relative to the content area of the window.

This function returns the position of the cursor, in screen coordinates, relative to the upper-left corner of the content area of the specified window.

If the cursor is disabled (with GLFW_CURSOR_DISABLED) then the cursor position is unbounded and limited only by the minimum and maximum values of a double.

The coordinate can be converted to their integer equivalents with the floor function. Casting directly to an integer type works for positive coordinates, but fails for negative ones.

Any or all of the position arguments may be NULL. If an error occurs, all non-NULL position arguments will be set to zero.

Parameters

in	*window*	The desired window.
out	*xpos*	Where to store the cursor x-coordinate, relative to the left edge of the content area, or NULL.
out	*ypos*	Where to store the cursor y-coordinate, relative to the to top edge of the content area, or NULL.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@thread_safety This function must only be called from the main thread.

See also

Cursor position

glfwSetCursorPos

Since

Added in version 3.0. Replaces `glfwGetMousePos`.

25.4.4.6 glfwGetGamepadName()

```
GLFWAPI const char * glfwGetGamepadName (
            int jid )
```

Returns the human-readable gamepad name for the specified joystick.

This function returns the human-readable name of the gamepad from the gamepad mapping assigned to the specified joystick.

If the specified joystick is not present or does not have a gamepad mapping this function will return `NULL` but will not generate an error. Call glfwJoystickPresent to check whether it is present regardless of whether it has a mapping.

Parameters

in	*jid*	The joystick to query.

Returns

The UTF-8 encoded name of the gamepad, or `NULL` if the joystick is not present, does not have a mapping or an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_INVALID_ENUM.

@pointer_lifetime The returned string is allocated and freed by GLFW. You should not free it yourself. It is valid until the specified joystick is disconnected, the gamepad mappings are updated or the library is terminated.

@thread_safety This function must only be called from the main thread.

See also

Gamepad input

glfwJoystickIsGamepad

Since

Added in version 3.3.

25.4.4.7 glfwGetGamepadState()

```
GLFWAPI int glfwGetGamepadState (
            int jid,
            GLFWgamepadstate * state )
```

Retrieves the state of the specified joystick remapped as a gamepad.

This function retrieves the state of the specified joystick remapped to an Xbox-like gamepad.

If the specified joystick is not present or does not have a gamepad mapping this function will return `GLFW_FALSE` but will not generate an error. Call glfwJoystickPresent to check whether it is present regardless of whether it has a mapping.

The Guide button may not be available for input as it is often hooked by the system or the Steam client.

Not all devices have all the buttons or axes provided by GLFWgamepadstate. Unavailable buttons and axes will always report `GLFW_RELEASE` and 0.0 respectively.

Parameters

in	*jid*	The joystick to query.
out	*state*	The gamepad input state of the joystick.

Returns

`GLFW_TRUE` if successful, or `GLFW_FALSE` if no joystick is connected, it has no gamepad mapping or an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_INVALID_ENUM.

@thread_safety This function must only be called from the main thread.

See also

Gamepad input

glfwUpdateGamepadMappings

glfwJoystickIsGamepad

Since

Added in version 3.3.

25.4.4.8 glfwGetInputMode()

```
GLFWAPI int glfwGetInputMode (
            GLFWwindow * window,
            int mode )
```

Returns the value of an input option for the specified window.

This function returns the value of an input option for the specified window. The mode must be one of GLFW_CURSOR, GLFW_STICKY_KEYS, GLFW_STICKY_MOUSE_BUTTONS, GLFW_LOCK_KEY_MODS or GLFW_RAW_MOUSE_MOTION.

Parameters

in	*window*	The window to query.
in	*mode*	One of GLFW_CURSOR, GLFW_STICKY_KEYS, GLFW_STICKY_MOUSE_BUTTONS, GLFW_LOCK_KEY_MODS or GLFW_RAW_MOUSE_MOTION.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_INVALID_ENUM.

@thread_safety This function must only be called from the main thread.

See also

glfwSetInputMode

Since

Added in version 3.0.

25.4.4.9 glfwGetJoystickAxes()

```
GLFWAPI const float * glfwGetJoystickAxes (
            int jid,
            int * count )
```

Returns the values of all axes of the specified joystick.

This function returns the values of all axes of the specified joystick. Each element in the array is a value between -1.0 and 1.0.

If the specified joystick is not present this function will return NULL but will not generate an error. This can be used instead of first calling glfwJoystickPresent.

Parameters

in	*jid*	The joystick to query.
out	*count*	Where to store the number of axis values in the returned array. This is set to zero if the joystick is not present or an error occurred.

Returns

An array of axis values, or NULL if the joystick is not present or an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_INVALID_ENUM and GLFW_PLATFORM_ERROR.

@pointer_lifetime The returned array is allocated and freed by GLFW. You should not free it yourself. It is valid until the specified joystick is disconnected or the library is terminated.

@thread_safety This function must only be called from the main thread.

See also

Joystick axis states

Since

Added in version 3.0. Replaces `glfwGetJoystickPos`.

25.4.4.10 glfwGetJoystickButtons()

```
GLFWAPI const unsigned char * glfwGetJoystickButtons (
            int jid,
            int * count )
```

Returns the state of all buttons of the specified joystick.

This function returns the state of all buttons of the specified joystick. Each element in the array is either `GLFW_PRESS` or `GLFW_RELEASE`.

For backward compatibility with earlier versions that did not have glfwGetJoystickHats, the button array also includes all hats, each represented as four buttons. The hats are in the same order as returned by **glfwGetJoystickHats** and are in the order *up*, *right*, *down* and *left*. To disable these extra buttons, set the GLFW_JOYSTICK_HAT_BUTTONS init hint before initialization.

If the specified joystick is not present this function will return `NULL` but will not generate an error. This can be used instead of first calling glfwJoystickPresent.

Parameters

`in`	*jid*	The joystick to query.
`out`	*count*	Where to store the number of button states in the returned array. This is set to zero if the joystick is not present or an error occurred.

Returns

An array of button states, or `NULL` if the joystick is not present or an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_INVALID_ENUM and GLFW_PLATFORM_ERROR.

@pointer_lifetime The returned array is allocated and freed by GLFW. You should not free it yourself. It is valid until the specified joystick is disconnected or the library is terminated.

@thread_safety This function must only be called from the main thread.

See also

Joystick button states

Since

Added in version 2.2. @glfw3 Changed to return a dynamic array.

25.4.4.11 glfwGetJoystickGUID()

```
GLFWAPI const char * glfwGetJoystickGUID (
            int jid )
```

Returns the SDL compatible GUID of the specified joystick.

This function returns the SDL compatible GUID, as a UTF-8 encoded hexadecimal string, of the specified joystick. The returned string is allocated and freed by GLFW. You should not free it yourself.

The GUID is what connects a joystick to a gamepad mapping. A connected joystick will always have a GUID even if there is no gamepad mapping assigned to it.

If the specified joystick is not present this function will return `NULL` but will not generate an error. This can be used instead of first calling glfwJoystickPresent.

The GUID uses the format introduced in SDL 2.0.5. This GUID tries to uniquely identify the make and model of a joystick but does not identify a specific unit, e.g. all wired Xbox 360 controllers will have the same GUID on that platform. The GUID for a unit may vary between platforms depending on what hardware information the platform specific APIs provide.

Parameters

in	*jid*	The joystick to query.

Returns

The UTF-8 encoded GUID of the joystick, or `NULL` if the joystick is not present or an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_INVALID_ENUM and GLFW_PLATFORM_ERROR.

@pointer_lifetime The returned string is allocated and freed by GLFW. You should not free it yourself. It is valid until the specified joystick is disconnected or the library is terminated.

@thread_safety This function must only be called from the main thread.

See also

Gamepad input

Since

Added in version 3.3.

25.4.4.12 glfwGetJoystickHats()

```
GLFWAPI const unsigned char * glfwGetJoystickHats (
            int jid,
            int * count )
```

Returns the state of all hats of the specified joystick.

This function returns the state of all hats of the specified joystick. Each element in the array is one of the following values:

Name	Value
GLFW_HAT_CENTERED	0
GLFW_HAT_UP	1
GLFW_HAT_RIGHT	2
GLFW_HAT_DOWN	4
GLFW_HAT_LEFT	8
GLFW_HAT_RIGHT_UP	GLFW_HAT_RIGHT | GLFW_HAT_UP
GLFW_HAT_RIGHT_DOWN	GLFW_HAT_RIGHT | GLFW_HAT_DOWN
GLFW_HAT_LEFT_UP	GLFW_HAT_LEFT | GLFW_HAT_UP
GLFW_HAT_LEFT_DOWN	GLFW_HAT_LEFT | GLFW_HAT_DOWN

The diagonal directions are bitwise combinations of the primary (up, right, down and left) directions and you can test for these individually by ANDing it with the corresponding direction.

```
if (hats[2] & GLFW_HAT_RIGHT)
{
    // State of hat 2 could be right-up, right or right-down
}
```

If the specified joystick is not present this function will return NULL but will not generate an error. This can be used instead of first calling glfwJoystickPresent.

Parameters

in	*jid*	The joystick to query.
out	*count*	Where to store the number of hat states in the returned array. This is set to zero if the joystick is not present or an error occurred.

Returns

An array of hat states, or NULL if the joystick is not present or an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_INVALID_ENUM and GLFW_PLATFORM_ERROR.

@pointer_lifetime The returned array is allocated and freed by GLFW. You should not free it yourself. It is valid until the specified joystick is disconnected, this function is called again for that joystick or the library is terminated.

@thread_safety This function must only be called from the main thread.

See also

Joystick hat states

Since

Added in version 3.3.

25.4.4.13 glfwGetJoystickName()

```
GLFWAPI const char * glfwGetJoystickName (
            int jid )
```

Returns the name of the specified joystick.

This function returns the name, encoded as UTF-8, of the specified joystick. The returned string is allocated and freed by GLFW. You should not free it yourself.

If the specified joystick is not present this function will return NULL but will not generate an error. This can be used instead of first calling glfwJoystickPresent.

Parameters

in	*jid*	The joystick to query.

Returns

The UTF-8 encoded name of the joystick, or `NULL` if the joystick is not present or an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_INVALID_ENUM and GLFW_PLATFORM_ERROR.

@pointer_lifetime The returned string is allocated and freed by GLFW. You should not free it yourself. It is valid until the specified joystick is disconnected or the library is terminated.

@thread_safety This function must only be called from the main thread.

See also

Joystick name

Since

Added in version 3.0.

25.4.4.14 glfwGetJoystickUserPointer()

```
GLFWAPI void * glfwGetJoystickUserPointer (
            int jid )
```

Returns the user pointer of the specified joystick.

This function returns the current value of the user-defined pointer of the specified joystick. The initial value is `NULL`.

This function may be called from the joystick callback, even for a joystick that is being disconnected.

Parameters

in	*jid*	The joystick whose pointer to return.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function may be called from any thread. Access is not synchronized.

See also

Joystick user pointer

glfwSetJoystickUserPointer

Since

Added in version 3.3.

25.4.4.15 glfwGetKey()

```
GLFWAPI int glfwGetKey (
            GLFWwindow * window,
            int key )
```

Returns the last reported state of a keyboard key for the specified window.

This function returns the last state reported for the specified key to the specified window. The returned state is one of GLFW_PRESS or GLFW_RELEASE. The higher-level action GLFW_REPEAT is only reported to the key callback.

If the GLFW_STICKY_KEYS input mode is enabled, this function returns GLFW_PRESS the first time you call it for a key that was pressed, even if that key has already been released.

The key functions deal with physical keys, with key tokens named after their use on the standard US keyboard layout. If you want to input text, use the Unicode character callback instead.

The modifier key bit masks are not key tokens and cannot be used with this function.

Do not use this function to implement text input.

Parameters

in	*window*	The desired window.
in	*key*	The desired keyboard key. GLFW_KEY_UNKNOWN is not a valid key for this function.

Returns

One of GLFW_PRESS or GLFW_RELEASE.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_INVALID_ENUM.

@thread_safety This function must only be called from the main thread.

See also

Key input

Since

Added in version 1.0. @glfw3 Added window handle parameter.

25.4.4.16 glfwGetKeyName()

```
GLFWAPI const char * glfwGetKeyName (
            int key,
            int scancode )
```

Returns the layout-specific name of the specified printable key.

This function returns the name of the specified printable key, encoded as UTF-8. This is typically the character that key would produce without any modifier keys, intended for displaying key bindings to the user. For dead keys, it is typically the diacritic it would add to a character.

Do not use this function for text input. You will break text input for many languages even if it happens to work for yours.

If the key is `GLFW_KEY_UNKNOWN`, the scancode is used to identify the key, otherwise the scancode is ignored. If you specify a non-printable key, or `GLFW_KEY_UNKNOWN` and a scancode that maps to a non-printable key, this function returns `NULL` but does not emit an error.

This behavior allows you to always pass in the arguments in the key callback without modification.

The printable keys are:

- `GLFW_KEY_APOSTROPHE`
- `GLFW_KEY_COMMA`
- `GLFW_KEY_MINUS`
- `GLFW_KEY_PERIOD`
- `GLFW_KEY_SLASH`
- `GLFW_KEY_SEMICOLON`
- `GLFW_KEY_EQUAL`
- `GLFW_KEY_LEFT_BRACKET`
- `GLFW_KEY_RIGHT_BRACKET`
- `GLFW_KEY_BACKSLASH`
- `GLFW_KEY_WORLD_1`
- `GLFW_KEY_WORLD_2`
- `GLFW_KEY_0` to `GLFW_KEY_9`
- `GLFW_KEY_A` to `GLFW_KEY_Z`
- `GLFW_KEY_KP_0` to `GLFW_KEY_KP_9`
- `GLFW_KEY_KP_DECIMAL`
- `GLFW_KEY_KP_DIVIDE`
- `GLFW_KEY_KP_MULTIPLY`
- `GLFW_KEY_KP_SUBTRACT`
- `GLFW_KEY_KP_ADD`
- `GLFW_KEY_KP_EQUAL`

Names for printable keys depend on keyboard layout, while names for non-printable keys are the same across layouts but depend on the application language and should be localized along with other user interface text.

Parameters

in	*key*	The key to query, or `GLFW_KEY_UNKNOWN`.
in	*scancode*	The scancode of the key to query.

Returns

The UTF-8 encoded, layout-specific name of the key, or `NULL`.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

Remarks

The contents of the returned string may change when a keyboard layout change event is received.

@pointer_lifetime The returned string is allocated and freed by GLFW. You should not free it yourself. It is valid until the library is terminated.

@thread_safety This function must only be called from the main thread.

See also

Key names

Since

Added in version 3.2.

25.4.4.17 glfwGetKeyScancode()

```
GLFWAPI int glfwGetKeyScancode (
            int key )
```

Returns the platform-specific scancode of the specified key.

This function returns the platform-specific scancode of the specified key.

If the key is `GLFW_KEY_UNKNOWN` or does not exist on the keyboard this method will return `-1`.

Parameters

in	*key*	Any named key.

Returns

The platform-specific scancode for the key, or `-1` if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_INVALID_ENUM and GLFW_PLATFORM_ERROR.

@thread_safety This function may be called from any thread.

See also

Key input

Since

Added in version 3.3.

25.4.4.18 glfwGetMouseButton()

```
GLFWAPI int glfwGetMouseButton (
            GLFWwindow * window,
            int button )
```

Returns the last reported state of a mouse button for the specified window.

This function returns the last state reported for the specified mouse button to the specified window. The returned state is one of GLFW_PRESS or GLFW_RELEASE.

If the GLFW_STICKY_MOUSE_BUTTONS input mode is enabled, this function returns GLFW_PRESS the first time you call it for a mouse button that was pressed, even if that mouse button has already been released.

Parameters

in	*window*	The desired window.
in	*button*	The desired mouse button.

Returns

One of GLFW_PRESS or GLFW_RELEASE.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_INVALID_ENUM.

@thread_safety This function must only be called from the main thread.

See also

Mouse button input

Since

Added in version 1.0. @glfw3 Added window handle parameter.

25.4.4.19 glfwGetTime()

```
GLFWAPI double glfwGetTime (
            void  )
```

Returns the GLFW time.

This function returns the current GLFW time, in seconds. Unless the time has been set using glfwSetTime it measures time elapsed since GLFW was initialized.

This function and glfwSetTime are helper functions on top of glfwGetTimerFrequency and glfwGetTimerValue.

The resolution of the timer is system dependent, but is usually on the order of a few micro- or nanoseconds. It uses the highest-resolution monotonic time source on each operating system.

Returns

The current time, in seconds, or zero if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function may be called from any thread. Reading and writing of the internal base time is not atomic, so it needs to be externally synchronized with calls to glfwSetTime.

See also

Time input

Since

Added in version 1.0.

25.4.4.20 glfwGetTimerFrequency()

```
GLFWAPI uint64_t glfwGetTimerFrequency (
            void  )
```

Returns the frequency, in Hz, of the raw timer.

This function returns the frequency, in Hz, of the raw timer.

Returns

The frequency of the timer, in Hz, or zero if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function may be called from any thread.

See also

Time input

glfwGetTimerValue

Since

Added in version 3.2.

25.4.4.21 glfwGetTimerValue()

```
GLFWAPI uint64_t glfwGetTimerValue (
            void  )
```

Returns the current value of the raw timer.

This function returns the current value of the raw timer, measured in 1 / frequency seconds. To get the frequency, call glfwGetTimerFrequency.

Returns

The value of the timer, or zero if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function may be called from any thread.

See also

Time input

glfwGetTimerFrequency

Since

Added in version 3.2.

25.4.4.22 glfwJoystickIsGamepad()

```
GLFWAPI int glfwJoystickIsGamepad (
            int jid )
```

Returns whether the specified joystick has a gamepad mapping.

This function returns whether the specified joystick is both present and has a gamepad mapping.

If the specified joystick is present but does not have a gamepad mapping this function will return `GLFW_FALSE` but will not generate an error. Call glfwJoystickPresent to check if a joystick is present regardless of whether it has a mapping.

Parameters

in	*jid*	The joystick to query.

Returns

`GLFW_TRUE` if a joystick is both present and has a gamepad mapping, or `GLFW_FALSE` otherwise.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_INVALID_ENUM.

@thread_safety This function must only be called from the main thread.

See also

Gamepad input

glfwGetGamepadState

Since

Added in version 3.3.

25.4.4.23 glfwJoystickPresent()

```
GLFWAPI int glfwJoystickPresent (
            int jid )
```

Returns whether the specified joystick is present.

This function returns whether the specified joystick is present.

There is no need to call this function before other functions that accept a joystick ID, as they all check for presence before performing any other work.

Parameters

in	*jid*	The joystick to query.

Returns

`GLFW_TRUE` if the joystick is present, or `GLFW_FALSE` otherwise.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_INVALID_ENUM and GLFW_PLATFORM_ERROR.

@thread_safety This function must only be called from the main thread.

See also

Joystick input

Since

Added in version 3.0. Replaces `glfwGetJoystickParam`.

25.4.4.24 glfwRawMouseMotionSupported()

```
GLFWAPI int glfwRawMouseMotionSupported (
            void  )
```

Returns whether raw mouse motion is supported.

This function returns whether raw mouse motion is supported on the current system. This status does not change after GLFW has been initialized so you only need to check this once. If you attempt to enable raw motion on a system that does not support it, GLFW_PLATFORM_ERROR will be emitted.

Raw mouse motion is closer to the actual motion of the mouse across a surface. It is not affected by the scaling and acceleration applied to the motion of the desktop cursor. That processing is suitable for a cursor while raw motion is better for controlling for example a 3D camera. Because of this, raw mouse motion is only provided when the cursor is disabled.

Returns

GLFW_TRUE if raw mouse motion is supported on the current machine, or GLFW_FALSE otherwise.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Raw mouse motion

glfwSetInputMode

Since

Added in version 3.3.

25.4.4.25 glfwSetCharCallback()

```
GLFWAPI GLFWcharfun glfwSetCharCallback (
            GLFWwindow * window,
            GLFWcharfun callback )
```

Sets the Unicode character callback.

This function sets the character callback of the specified window, which is called when a Unicode character is input.

The character callback is intended for Unicode text input. As it deals with characters, it is keyboard layout dependent, whereas the key callback is not. Characters do not map 1:1 to physical keys, as a key may produce zero, one or more characters. If you want to know whether a specific physical key was pressed or released, see the key callback instead.

The character callback behaves as system text input normally does and will not be called if modifier keys are held down that would prevent normal text input on that platform, for example a Super (Command) key on macOS or Alt key on Windows.

Parameters

in	*window*	The window whose callback to set.
in	*callback*	The new callback, or NULL to remove the currently set callback.

Returns

The previously set callback, or NULL if no callback was set or the library had not been initialized.

@callback_signature
```
void function_name(GLFWwindow* window, unsigned int codepoint)
```

For more information about the callback parameters, see the function pointer type.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Text input

Since

Added in version 2.4. @glfw3 Added window handle parameter and return value.

25.4.4.26 glfwSetCharModsCallback()

```
GLFWAPI GLFWcharmodsfun glfwSetCharModsCallback (
            GLFWwindow * window,
            GLFWcharmodsfun callback )
```

Sets the Unicode character with modifiers callback.

This function sets the character with modifiers callback of the specified window, which is called when a Unicode character is input regardless of what modifier keys are used.

The character with modifiers callback is intended for implementing custom Unicode character input. For regular Unicode text input, see the character callback. Like the character callback, the character with modifiers callback deals with characters and is keyboard layout dependent. Characters do not map 1:1 to physical keys, as a key may produce zero, one or more characters. If you want to know whether a specific physical key was pressed or released, see the key callback instead.

Parameters

in	*window*	The window whose callback to set.
in	*callback*	The new callback, or NULL to remove the currently set callback.

Returns

The previously set callback, or `NULL` if no callback was set or an error occurred.

@callback_signature
```
void function_name(GLFWwindow* window, unsigned int codepoint, int mods)
```

For more information about the callback parameters, see the function pointer type.

Deprecated Scheduled for removal in version 4.0.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Text input

Since

Added in version 3.1.

25.4.4.27 glfwSetClipboardString()

```
GLFWAPI void glfwSetClipboardString (
            GLFWwindow * window,
            const char * string )
```

Sets the clipboard to the specified string.

This function sets the system clipboard to the specified, UTF-8 encoded string.

Parameters

in	*window*	Deprecated. Any valid window or `NULL`.
in	*string*	A UTF-8 encoded string.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@pointer_lifetime The specified string is copied before this function returns.

@thread_safety This function must only be called from the main thread.

See also

Clipboard input and output

glfwGetClipboardString

Since

Added in version 3.0.

25.4.4.28 glfwSetCursor()

```
GLFWAPI void glfwSetCursor (
            GLFWwindow * window,
            GLFWcursor * cursor )
```

Sets the cursor for the window.

This function sets the cursor image to be used when the cursor is over the content area of the specified window. The set cursor will only be visible when the cursor mode of the window is GLFW_CURSOR_NORMAL.

On some platforms, the set cursor may not be visible unless the window also has input focus.

Parameters

in	*window*	The window to set the cursor for.
in	*cursor*	The cursor to set, or NULL to switch back to the default arrow cursor.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@thread_safety This function must only be called from the main thread.

See also

Cursor objects

Since

Added in version 3.1.

25.4.4.29 glfwSetCursorEnterCallback()

```
GLFWAPI GLFWcursorenterfun glfwSetCursorEnterCallback (
            GLFWwindow * window,
            GLFWcursorenterfun callback )
```

Sets the cursor enter/leave callback.

This function sets the cursor boundary crossing callback of the specified window, which is called when the cursor enters or leaves the content area of the window.

Parameters

in	*window*	The window whose callback to set.
in	*callback*	The new callback, or NULL to remove the currently set callback.

Returns

The previously set callback, or NULL if no callback was set or the library had not been initialized.

@callback_signature
```
void function_name(GLFWwindow* window, int entered)
```

For more information about the callback parameters, see the function pointer type.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Cursor enter/leave events

Since

Added in version 3.0.

25.4.4.30 glfwSetCursorPos()

```
GLFWAPI void glfwSetCursorPos (
            GLFWwindow * window,
            double xpos,
            double ypos )
```

Sets the position of the cursor, relative to the content area of the window.

This function sets the position, in screen coordinates, of the cursor relative to the upper-left corner of the content area of the specified window. The window must have input focus. If the window does not have input focus when this function is called, it fails silently.

Do not use this function to implement things like camera controls. GLFW already provides the GLFW_CURSOR↩ _DISABLED cursor mode that hides the cursor, transparently re-centers it and provides unconstrained cursor motion. See glfwSetInputMode for more information.

If the cursor mode is GLFW_CURSOR_DISABLED then the cursor position is unconstrained and limited only by the minimum and maximum values of a double.

Parameters

in	*window*	The desired window.
in	*xpos*	The desired x-coordinate, relative to the left edge of the content area.
in	*ypos*	The desired y-coordinate, relative to the top edge of the content area.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

Remarks

@wayland This function will only work when the cursor mode is `GLFW_CURSOR_DISABLED`, otherwise it will do nothing.

@thread_safety This function must only be called from the main thread.

See also

Cursor position

glfwGetCursorPos

Since

Added in version 3.0. Replaces `glfwSetMousePos`.

25.4.4.31 glfwSetCursorPosCallback()

```
GLFWAPI GLFWcursorposfun glfwSetCursorPosCallback (
            GLFWwindow * window,
            GLFWcursorposfun callback )
```

Sets the cursor position callback.

This function sets the cursor position callback of the specified window, which is called when the cursor is moved. The callback is provided with the position, in screen coordinates, relative to the upper-left corner of the content area of the window.

Parameters

in	*window*	The window whose callback to set.
in	*callback*	The new callback, or `NULL` to remove the currently set callback.

Returns

The previously set callback, or `NULL` if no callback was set or the library had not been initialized.

@callback_signature
```
void function_name(GLFWwindow* window, double xpos, double ypos);
```

For more information about the callback parameters, see the function pointer type.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Cursor position

Since

Added in version 3.0. Replaces `glfwSetMousePosCallback`.

25.4.4.32 glfwSetDropCallback()

```
GLFWAPI GLFWdropfun glfwSetDropCallback (
            GLFWwindow * window,
            GLFWdropfun callback )
```

Sets the path drop callback.

This function sets the path drop callback of the specified window, which is called when one or more dragged paths are dropped on the window.

Because the path array and its strings may have been generated specifically for that event, they are not guaranteed to be valid after the callback has returned. If you wish to use them after the callback returns, you need to make a deep copy.

Parameters

in	*window*	The window whose callback to set.
in	*callback*	The new file drop callback, or `NULL` to remove the currently set callback.

Returns

The previously set callback, or `NULL` if no callback was set or the library had not been initialized.

@callback_signature
```
void function_name(GLFWwindow* window, int path_count, const char* paths[])
```

For more information about the callback parameters, see the function pointer type.

@errors Possible errors include GLFW_NOT_INITIALIZED.

Remarks

@wayland File drop is currently unimplemented.

@thread_safety This function must only be called from the main thread.

See also

Path drop input

Since

Added in version 3.1.

25.4.4.33 glfwSetInputMode()

```
GLFWAPI void glfwSetInputMode (
            GLFWwindow * window,
            int mode,
            int value )
```

Sets an input option for the specified window.

This function sets an input mode option for the specified window. The mode must be one of GLFW_CURSOR, GLFW_STICKY_KEYS, GLFW_STICKY_MOUSE_BUTTONS, GLFW_LOCK_KEY_MODS or GLFW_RAW_MOUSE_MOTION.

If the mode is GLFW_CURSOR, the value must be one of the following cursor modes:

- GLFW_CURSOR_NORMAL makes the cursor visible and behaving normally.
- GLFW_CURSOR_HIDDEN makes the cursor invisible when it is over the content area of the window but does not restrict the cursor from leaving.
- GLFW_CURSOR_DISABLED hides and grabs the cursor, providing virtual and unlimited cursor movement. This is useful for implementing for example 3D camera controls.

If the mode is GLFW_STICKY_KEYS, the value must be either GLFW_TRUE to enable sticky keys, or GLFW_↩ FALSE to disable it. If sticky keys are enabled, a key press will ensure that glfwGetKey returns GLFW_PRESS the next time it is called even if the key had been released before the call. This is useful when you are only interested in whether keys have been pressed but not when or in which order.

If the mode is GLFW_STICKY_MOUSE_BUTTONS, the value must be either GLFW_TRUE to enable sticky mouse buttons, or GLFW_FALSE to disable it. If sticky mouse buttons are enabled, a mouse button press will ensure that glfwGetMouseButton returns GLFW_PRESS the next time it is called even if the mouse button had been released before the call. This is useful when you are only interested in whether mouse buttons have been pressed but not when or in which order.

If the mode is GLFW_LOCK_KEY_MODS, the value must be either GLFW_TRUE to enable lock key modifier bits, or GLFW_FALSE to disable them. If enabled, callbacks that receive modifier bits will also have the GLFW_MOD_CAPS_LOCK bit set when the event was generated with Caps Lock on, and the GLFW_MOD_NUM_LOCK bit when Num Lock was on.

If the mode is GLFW_RAW_MOUSE_MOTION, the value must be either GLFW_TRUE to enable raw (unscaled and unaccelerated) mouse motion when the cursor is disabled, or GLFW_FALSE to disable it. If raw motion is not supported, attempting to set this will emit GLFW_FEATURE_UNAVAILABLE. Call glfwRawMouseMotionSupported to check for support.

Parameters

in	*window*	The window whose input mode to set.
in	*mode*	One of GLFW_CURSOR, GLFW_STICKY_KEYS, GLFW_STICKY_MOUSE_BUTTONS, GLFW_LOCK_KEY_MODS or GLFW_RAW_MOUSE_MOTION.
in	*value*	The new value of the specified input mode.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_INVALID_ENUM, GLFW_PLATFORM_ERROR and GLFW_FEATURE_UNAVAILABLE (see above).

@thread_safety This function must only be called from the main thread.

See also

glfwGetInputMode

Since

Added in version 3.0. Replaces `glfwEnable` and `glfwDisable`.

25.4.4.34 glfwSetJoystickCallback()

```
GLFWAPI GLFWjoystickfun glfwSetJoystickCallback (
            GLFWjoystickfun callback )
```

Sets the joystick configuration callback.

This function sets the joystick configuration callback, or removes the currently set callback. This is called when a joystick is connected to or disconnected from the system.

For joystick connection and disconnection events to be delivered on all platforms, you need to call one of the event processing functions. Joystick disconnection may also be detected and the callback called by joystick functions. The function will then return whatever it returns if the joystick is not present.

Parameters

in	*callback*	The new callback, or `NULL` to remove the currently set callback.

Returns

The previously set callback, or `NULL` if no callback was set or the library had not been initialized.

@callback_signature
```
void function_name(int jid, int event)
```

For more information about the callback parameters, see the function pointer type.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Joystick configuration changes

Since

Added in version 3.2.

25.4.4.35 glfwSetJoystickUserPointer()

```
GLFWAPI void glfwSetJoystickUserPointer (
            int jid,
            void * pointer )
```

Sets the user pointer of the specified joystick.

This function sets the user-defined pointer of the specified joystick. The current value is retained until the joystick is disconnected. The initial value is `NULL`.

This function may be called from the joystick callback, even for a joystick that is being disconnected.

Parameters

in	*jid*	The joystick whose pointer to set.
in	*pointer*	The new value.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function may be called from any thread. Access is not synchronized.

See also

Joystick user pointer

glfwGetJoystickUserPointer

Since

Added in version 3.3.

25.4.4.36 glfwSetKeyCallback()

```
GLFWAPI GLFWkeyfun glfwSetKeyCallback (
            GLFWwindow * window,
            GLFWkeyfun callback )
```

Sets the key callback.

This function sets the key callback of the specified window, which is called when a key is pressed, repeated or released.

The key functions deal with physical keys, with layout independent key tokens named after their values in the standard US keyboard layout. If you want to input text, use the character callback instead.

When a window loses input focus, it will generate synthetic key release events for all pressed keys. You can tell these events from user-generated events by the fact that the synthetic ones are generated after the focus loss event has been processed, i.e. after the window focus callback has been called.

The scancode of a key is specific to that platform or sometimes even to that machine. Scancodes are intended to allow users to bind keys that don't have a GLFW key token. Such keys have `key` set to `GLFW_KEY_UNKNOWN`, their state is not saved and so it cannot be queried with glfwGetKey.

Sometimes GLFW needs to generate synthetic key events, in which case the scancode may be zero.

Parameters

in	*window*	The window whose callback to set.
in	*callback*	The new key callback, or NULL to remove the currently set callback.

Returns

The previously set callback, or NULL if no callback was set or the library had not been initialized.

@callback_signature
```
void function_name(GLFWwindow* window, int key, int scancode, int action, int mods)
```

For more information about the callback parameters, see the function pointer type.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Key input

Since

Added in version 1.0. @glfw3 Added window handle parameter and return value.

25.4.4.37 glfwSetMouseButtonCallback()

```
GLFWAPI GLFWmousebuttonfun glfwSetMouseButtonCallback (
            GLFWwindow * window,
            GLFWmousebuttonfun callback )
```

Sets the mouse button callback.

This function sets the mouse button callback of the specified window, which is called when a mouse button is pressed or released.

When a window loses input focus, it will generate synthetic mouse button release events for all pressed mouse buttons. You can tell these events from user-generated events by the fact that the synthetic ones are generated after the focus loss event has been processed, i.e. after the window focus callback has been called.

Parameters

in	*window*	The window whose callback to set.
in	*callback*	The new callback, or NULL to remove the currently set callback.

Returns

The previously set callback, or NULL if no callback was set or the library had not been initialized.

@callback_signature
```
void function_name(GLFWwindow* window, int button, int action, int mods)
```

For more information about the callback parameters, see the function pointer type.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Mouse button input

Since

Added in version 1.0. @glfw3 Added window handle parameter and return value.

25.4.4.38 glfwSetScrollCallback()

```
GLFWAPI GLFWscrollfun glfwSetScrollCallback (
            GLFWwindow * window,
            GLFWscrollfun callback )
```

Sets the scroll callback.

This function sets the scroll callback of the specified window, which is called when a scrolling device is used, such as a mouse wheel or scrolling area of a touchpad.

The scroll callback receives all scrolling input, like that from a mouse wheel or a touchpad scrolling area.

Parameters

in	*window*	The window whose callback to set.
in	*callback*	The new scroll callback, or NULL to remove the currently set callback.

Returns

The previously set callback, or NULL if no callback was set or the library had not been initialized.

@callback_signature
```
void function_name(GLFWwindow* window, double xoffset, double yoffset)
```

For more information about the callback parameters, see the function pointer type.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Scroll input

Since

Added in version 3.0. Replaces `glfwSetMouseWheelCallback`.

25.4.4.39 glfwSetTime()

```
GLFWAPI void glfwSetTime (
            double time )
```

Sets the GLFW time.

This function sets the current GLFW time, in seconds. The value must be a positive finite number less than or equal to 18446744073.0, which is approximately 584.5 years.

This function and glfwGetTime are helper functions on top of glfwGetTimerFrequency and glfwGetTimerValue.

Parameters

in	*time*	The new value, in seconds.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_INVALID_VALUE.

Remarks

The upper limit of GLFW time is calculated as floor((2^{64} - 1) / 10^{9}) and is due to implementations storing nanoseconds in 64 bits. The limit may be increased in the future.

@thread_safety This function may be called from any thread. Reading and writing of the internal base time is not atomic, so it needs to be externally synchronized with calls to glfwGetTime.

See also

Time input

Since

Added in version 2.2.

25.4.4.40 glfwUpdateGamepadMappings()

```
GLFWAPI int glfwUpdateGamepadMappings (
            const char * string )
```

Adds the specified SDL_GameControllerDB gamepad mappings.

This function parses the specified ASCII encoded string and updates the internal list with any gamepad mappings it finds. This string may contain either a single gamepad mapping or many mappings separated by newlines. The parser supports the full format of the `gamecontrollerdb.txt` source file including empty lines and comments.

See Gamepad mappings for a description of the format.

If there is already a gamepad mapping for a given GUID in the internal list, it will be replaced by the one passed to this function. If the library is terminated and re-initialized the internal list will revert to the built-in default.

Parameters

in	*string*	The string containing the gamepad mappings.

Returns

`GLFW_TRUE` if successful, or `GLFW_FALSE` if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_INVALID_VALUE.

@thread_safety This function must only be called from the main thread.

See also

Gamepad input

glfwJoystickIsGamepad

glfwGetGamepadName

Since

Added in version 3.3.

25.5 Monitor reference

Functions and types related to monitors.

Classes

- struct GLFWvidmode

  Video mode type.
- struct GLFWgammaramp

  Gamma ramp.

Typedefs

- typedef struct GLFWmonitor GLFWmonitor

  Opaque monitor object.
- typedef void(∗ GLFWmonitorfun) (GLFWmonitor ∗monitor, int event)

  The function pointer type for monitor configuration callbacks.
- typedef struct GLFWvidmode GLFWvidmode

  Video mode type.
- typedef struct GLFWgammaramp GLFWgammaramp

  Gamma ramp.

Functions

- GLFWAPI GLFWmonitor ∗∗ glfwGetMonitors (int ∗count)

  Returns the currently connected monitors.
- GLFWAPI GLFWmonitor ∗ glfwGetPrimaryMonitor (void)

  Returns the primary monitor.
- GLFWAPI void glfwGetMonitorPos (GLFWmonitor ∗monitor, int ∗xpos, int ∗ypos)

  Returns the position of the monitor's viewport on the virtual screen.
- GLFWAPI void glfwGetMonitorWorkarea (GLFWmonitor ∗monitor, int ∗xpos, int ∗ypos, int ∗width, int ∗height)

  Retrieves the work area of the monitor.
- GLFWAPI void glfwGetMonitorPhysicalSize (GLFWmonitor ∗monitor, int ∗widthMM, int ∗heightMM)

  Returns the physical size of the monitor.
- GLFWAPI void glfwGetMonitorContentScale (GLFWmonitor ∗monitor, float ∗xscale, float ∗yscale)

  Retrieves the content scale for the specified monitor.
- GLFWAPI const char ∗ glfwGetMonitorName (GLFWmonitor ∗monitor)

  Returns the name of the specified monitor.
- GLFWAPI void glfwSetMonitorUserPointer (GLFWmonitor ∗monitor, void ∗pointer)

  Sets the user pointer of the specified monitor.
- GLFWAPI void ∗ glfwGetMonitorUserPointer (GLFWmonitor ∗monitor)

  Returns the user pointer of the specified monitor.
- GLFWAPI GLFWmonitorfun glfwSetMonitorCallback (GLFWmonitorfun callback)

  Sets the monitor configuration callback.
- GLFWAPI const GLFWvidmode ∗ glfwGetVideoModes (GLFWmonitor ∗monitor, int ∗count)

  Returns the available video modes for the specified monitor.
- GLFWAPI const GLFWvidmode ∗ glfwGetVideoMode (GLFWmonitor ∗monitor)

  Returns the current mode of the specified monitor.
- GLFWAPI void glfwSetGamma (GLFWmonitor ∗monitor, float gamma)

  Generates a gamma ramp and sets it for the specified monitor.
- GLFWAPI const GLFWgammaramp ∗ glfwGetGammaRamp (GLFWmonitor ∗monitor)

  Returns the current gamma ramp for the specified monitor.
- GLFWAPI void glfwSetGammaRamp (GLFWmonitor ∗monitor, const GLFWgammaramp ∗ramp)

  Sets the current gamma ramp for the specified monitor.

25.5.1 Detailed Description

Functions and types related to monitors.

This is the reference documentation for monitor related functions and types. For more task-oriented information, see the Monitor guide.

25.5.2 Typedef Documentation

25.5.2.1 GLFWgammaramp

```
typedef struct GLFWgammaramp GLFWgammaramp
```

Gamma ramp.

This describes the gamma ramp for a monitor.

See also

Gamma ramp
glfwGetGammaRamp
glfwSetGammaRamp

Since

Added in version 3.0.

25.5.2.2 GLFWmonitor

```
typedef struct GLFWmonitor GLFWmonitor
```

Opaque monitor object.

Opaque monitor object.

See also

Monitor objects

Since

Added in version 3.0.

25.5.2.3 GLFWmonitorfun

```
typedef void(* GLFWmonitorfun) (GLFWmonitor *monitor, int event)
```

The function pointer type for monitor configuration callbacks.

This is the function pointer type for monitor configuration callbacks. A monitor callback function has the following signature:

```
void function_name(GLFWmonitor* monitor, int event)
```

Parameters

in	*monitor*	The monitor that was connected or disconnected.
in	*event*	One of GLFW_CONNECTED or GLFW_DISCONNECTED. Future releases may add more events.

See also

Monitor configuration changes

glfwSetMonitorCallback

Since

Added in version 3.0.

25.5.2.4 GLFWvidmode

```
typedef struct GLFWvidmode GLFWvidmode
```

Video mode type.

This describes a single video mode.

See also

Video modes

glfwGetVideoMode

glfwGetVideoModes

Since

Added in version 1.0. @glfw3 Added refresh rate member.

25.5.3 Function Documentation

25.5.3.1 glfwGetGammaRamp()

```
GLFWAPI const GLFWgammaramp * glfwGetGammaRamp (
            GLFWmonitor * monitor )
```

Returns the current gamma ramp for the specified monitor.

This function returns the current gamma ramp of the specified monitor.

Parameters

in	*monitor*	The monitor to query.

Returns

The current gamma ramp, or `NULL` if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

Remarks

@wayland Gamma handling is a privileged protocol, this function will thus never be implemented and emits GLFW_PLATFORM_ERROR while returning `NULL`.

@pointer_lifetime The returned structure and its arrays are allocated and freed by GLFW. You should not free them yourself. They are valid until the specified monitor is disconnected, this function is called again for that monitor or the library is terminated.

@thread_safety This function must only be called from the main thread.

See also

Gamma ramp

Since

Added in version 3.0.

25.5.3.2 glfwGetMonitorContentScale()

```
GLFWAPI void glfwGetMonitorContentScale (
             GLFWmonitor * monitor,
             float * xscale,
             float * yscale )
```

Retrieves the content scale for the specified monitor.

This function retrieves the content scale for the specified monitor. The content scale is the ratio between the current DPI and the platform's default DPI. This is especially important for text and any UI elements. If the pixel dimensions of your UI scaled by this look appropriate on your machine then it should appear at a reasonable size on other machines regardless of their DPI and scaling settings. This relies on the system DPI and scaling settings being somewhat correct.

The content scale may depend on both the monitor resolution and pixel density and on user settings. It may be very different from the raw DPI calculated from the physical size and current resolution.

Parameters

in	*monitor*	The monitor to query.
out	*xscale*	Where to store the x-axis content scale, or NULL.
out	*yscale*	Where to store the y-axis content scale, or NULL.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@thread_safety This function must only be called from the main thread.

See also

Content scale

glfwGetWindowContentScale

Since

Added in version 3.3.

25.5.3.3 glfwGetMonitorName()

```
GLFWAPI const char * glfwGetMonitorName (
            GLFWmonitor * monitor )
```

Returns the name of the specified monitor.

This function returns a human-readable name, encoded as UTF-8, of the specified monitor. The name typically reflects the make and model of the monitor and is not guaranteed to be unique among the connected monitors.

Parameters

in	*monitor*	The monitor to query.

Returns

The UTF-8 encoded name of the monitor, or NULL if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@pointer_lifetime The returned string is allocated and freed by GLFW. You should not free it yourself. It is valid until the specified monitor is disconnected or the library is terminated.

@thread_safety This function must only be called from the main thread.

See also

Monitor properties

Since

Added in version 3.0.

25.5.3.4 glfwGetMonitorPhysicalSize()

```
GLFWAPI void glfwGetMonitorPhysicalSize (
            GLFWmonitor * monitor,
            int * widthMM,
            int * heightMM )
```

Returns the physical size of the monitor.

This function returns the size, in millimetres, of the display area of the specified monitor.

Some platforms do not provide accurate monitor size information, either because the monitor EDID data is incorrect or because the driver does not report it accurately.

Any or all of the size arguments may be NULL. If an error occurs, all non-NULL size arguments will be set to zero.

Parameters

in	*monitor*	The monitor to query.
out	*widthMM*	Where to store the width, in millimetres, of the monitor's display area, or NULL.
out	*heightMM*	Where to store the height, in millimetres, of the monitor's display area, or NULL.

@errors Possible errors include GLFW_NOT_INITIALIZED.

Remarks

@win32 On Windows 8 and earlier the physical size is calculated from the current resolution and system DPI instead of querying the monitor EDID data.

@thread_safety This function must only be called from the main thread.

See also

Monitor properties

Since

Added in version 3.0.

25.5.3.5 glfwGetMonitorPos()

```
GLFWAPI void glfwGetMonitorPos (
            GLFWmonitor * monitor,
            int * xpos,
            int * ypos )
```

Returns the position of the monitor's viewport on the virtual screen.

This function returns the position, in screen coordinates, of the upper-left corner of the specified monitor.

Any or all of the position arguments may be NULL. If an error occurs, all non-NULL position arguments will be set to zero.

Parameters

in	*monitor*	The monitor to query.
out	*xpos*	Where to store the monitor x-coordinate, or NULL.
out	*ypos*	Where to store the monitor y-coordinate, or NULL.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@thread_safety This function must only be called from the main thread.

See also

Monitor properties

Since

Added in version 3.0.

25.5.3.6 glfwGetMonitors()

```
GLFWAPI GLFWmonitor ** glfwGetMonitors (
            int * count )
```

Returns the currently connected monitors.

This function returns an array of handles for all currently connected monitors. The primary monitor is always first in the returned array. If no monitors were found, this function returns NULL.

Parameters

out	*count*	Where to store the number of monitors in the returned array. This is set to zero if an error occurred.

Returns

An array of monitor handles, or NULL if no monitors were found or if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@pointer_lifetime The returned array is allocated and freed by GLFW. You should not free it yourself. It is guaranteed to be valid only until the monitor configuration changes or the library is terminated.

@thread_safety This function must only be called from the main thread.

See also

Retrieving monitors

Monitor configuration changes

glfwGetPrimaryMonitor

Since

Added in version 3.0.

25.5.3.7 glfwGetMonitorUserPointer()

```
GLFWAPI void * glfwGetMonitorUserPointer (
            GLFWmonitor * monitor )
```

Returns the user pointer of the specified monitor.

This function returns the current value of the user-defined pointer of the specified monitor. The initial value is NULL.

This function may be called from the monitor callback, even for a monitor that is being disconnected.

Parameters

in	*monitor*	The monitor whose pointer to return.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function may be called from any thread. Access is not synchronized.

See also

User pointer

glfwSetMonitorUserPointer

Since

Added in version 3.3.

25.5.3.8 glfwGetMonitorWorkarea()

```
GLFWAPI void glfwGetMonitorWorkarea (
            GLFWmonitor * monitor,
            int * xpos,
            int * ypos,
            int * width,
            int * height )
```

Retrieves the work area of the monitor.

This function returns the position, in screen coordinates, of the upper-left corner of the work area of the specified monitor along with the work area size in screen coordinates. The work area is defined as the area of the monitor not occluded by the window system task bar where present. If no task bar exists then the work area is the monitor resolution in screen coordinates.

Any or all of the position and size arguments may be NULL. If an error occurs, all non-NULL position and size arguments will be set to zero.

Parameters

in	*monitor*	The monitor to query.
out	*xpos*	Where to store the monitor x-coordinate, or NULL.
out	*ypos*	Where to store the monitor y-coordinate, or NULL.
out	*width*	Where to store the monitor width, or NULL.
out	*height*	Where to store the monitor height, or NULL.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@thread_safety This function must only be called from the main thread.

See also

Work area

Since

Added in version 3.3.

25.5.3.9 glfwGetPrimaryMonitor()

```
GLFWAPI GLFWmonitor * glfwGetPrimaryMonitor (
            void  )
```

Returns the primary monitor.

This function returns the primary monitor. This is usually the monitor where elements like the task bar or global menu bar are located.

Returns

The primary monitor, or NULL if no monitors were found or if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

Remarks

The primary monitor is always first in the array returned by glfwGetMonitors.

See also

Retrieving monitors

glfwGetMonitors

Since

Added in version 3.0.

25.5.3.10 glfwGetVideoMode()

```
GLFWAPI const GLFWvidmode * glfwGetVideoMode (
            GLFWmonitor * monitor )
```

Returns the current mode of the specified monitor.

This function returns the current video mode of the specified monitor. If you have created a full screen window for that monitor, the return value will depend on whether that window is iconified.

Parameters

in	*monitor*	The monitor to query.

Returns

The current mode of the monitor, or NULL if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@pointer_lifetime The returned array is allocated and freed by GLFW. You should not free it yourself. It is valid until the specified monitor is disconnected or the library is terminated.

@thread_safety This function must only be called from the main thread.

See also

Video modes

glfwGetVideoModes

Since

Added in version 3.0. Replaces glfwGetDesktopMode.

25.5.3.11 glfwGetVideoModes()

```
GLFWAPI const GLFWvidmode * glfwGetVideoModes (
            GLFWmonitor * monitor,
            int * count )
```

Returns the available video modes for the specified monitor.

This function returns an array of all video modes supported by the specified monitor. The returned array is sorted in ascending order, first by color bit depth (the sum of all channel depths), then by resolution area (the product of width and height), then resolution width and finally by refresh rate.

Parameters

in	*monitor*	The monitor to query.
out	*count*	Where to store the number of video modes in the returned array. This is set to zero if an error occurred.

Returns

An array of video modes, or NULL if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@pointer_lifetime The returned array is allocated and freed by GLFW. You should not free it yourself. It is valid until the specified monitor is disconnected, this function is called again for that monitor or the library is terminated.

@thread_safety This function must only be called from the main thread.

See also

Video modes

glfwGetVideoMode

Since

Added in version 1.0. @glfw3 Changed to return an array of modes for a specific monitor.

25.5.3.12 glfwSetGamma()

```
GLFWAPI void glfwSetGamma (
            GLFWmonitor * monitor,
            float gamma )
```

Generates a gamma ramp and sets it for the specified monitor.

This function generates an appropriately sized gamma ramp from the specified exponent and then calls glfwSetGammaRamp with it. The value must be a finite number greater than zero.

The software controlled gamma ramp is applied *in addition* to the hardware gamma correction, which today is usually an approximation of sRGB gamma. This means that setting a perfectly linear ramp, or gamma 1.0, will produce the default (usually sRGB-like) behavior.

For gamma correct rendering with OpenGL or OpenGL ES, see the GLFW_SRGB_CAPABLE hint.

Parameters

in	*monitor*	The monitor whose gamma ramp to set.
in	*gamma*	The desired exponent.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_INVALID_VALUE and GLFW_PLATFORM_ERROR.

Remarks

@wayland Gamma handling is a privileged protocol, this function will thus never be implemented and emits GLFW_PLATFORM_ERROR.

@thread_safety This function must only be called from the main thread.

See also

Gamma ramp

Since

Added in version 3.0.

25.5.3.13 glfwSetGammaRamp()

```
GLFWAPI void glfwSetGammaRamp (
            GLFWmonitor * monitor,
            const GLFWgammaramp * ramp )
```

Sets the current gamma ramp for the specified monitor.

This function sets the current gamma ramp for the specified monitor. The original gamma ramp for that monitor is saved by GLFW the first time this function is called and is restored by glfwTerminate.

The software controlled gamma ramp is applied *in addition* to the hardware gamma correction, which today is usually an approximation of sRGB gamma. This means that setting a perfectly linear ramp, or gamma 1.0, will produce the default (usually sRGB-like) behavior.

For gamma correct rendering with OpenGL or OpenGL ES, see the GLFW_SRGB_CAPABLE hint.

Parameters

in	*monitor*	The monitor whose gamma ramp to set.
in	*ramp*	The gamma ramp to use.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

Remarks

The size of the specified gamma ramp should match the size of the current ramp for that monitor.

@win32 The gamma ramp size must be 256.

@wayland Gamma handling is a privileged protocol, this function will thus never be implemented and emits GLFW_PLATFORM_ERROR.

@pointer_lifetime The specified gamma ramp is copied before this function returns.

@thread_safety This function must only be called from the main thread.

See also

Gamma ramp

Since

Added in version 3.0.

25.5.3.14 glfwSetMonitorCallback()

```
GLFWAPI GLFWmonitorfun glfwSetMonitorCallback (
            GLFWmonitorfun callback )
```

Sets the monitor configuration callback.

This function sets the monitor configuration callback, or removes the currently set callback. This is called when a monitor is connected to or disconnected from the system.

Parameters

in	*callback*	The new callback, or NULL to remove the currently set callback.

Returns

The previously set callback, or NULL if no callback was set or the library had not been initialized.

@callback_signature
```
void function_name(GLFWmonitor* monitor, int event)
```

For more information about the callback parameters, see the function pointer type.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Monitor configuration changes

Since

Added in version 3.0.

25.5.3.15 glfwSetMonitorUserPointer()

```
GLFWAPI void glfwSetMonitorUserPointer (
            GLFWmonitor * monitor,
            void * pointer )
```

Sets the user pointer of the specified monitor.

This function sets the user-defined pointer of the specified monitor. The current value is retained until the monitor is disconnected. The initial value is NULL.

This function may be called from the monitor callback, even for a monitor that is being disconnected.

Parameters

in	*monitor*	The monitor whose pointer to set.
in	*pointer*	The new value.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function may be called from any thread. Access is not synchronized.

See also

User pointer

glfwGetMonitorUserPointer

Since

Added in version 3.3.

25.6 Window reference

Functions and types related to windows.

Classes

- struct GLFWimage

  Image data.

Macros

- #define GLFW_FOCUSED 0x00020001

  Input focus window hint and attribute.
- #define GLFW_ICONIFIED 0x00020002

  Window iconification window attribute.
- #define GLFW_RESIZABLE 0x00020003

  Window resize-ability window hint and attribute.
- #define GLFW_VISIBLE 0x00020004

  Window visibility window hint and attribute.
- #define GLFW_DECORATED 0x00020005

  Window decoration window hint and attribute.
- #define GLFW_AUTO_ICONIFY 0x00020006

  Window auto-iconification window hint and attribute.
- #define GLFW_FLOATING 0x00020007

  Window decoration window hint and attribute.
- #define GLFW_MAXIMIZED 0x00020008

  Window maximization window hint and attribute.
- #define GLFW_CENTER_CURSOR 0x00020009

  Cursor centering window hint.
- #define GLFW_TRANSPARENT_FRAMEBUFFER 0x0002000A

  Window framebuffer transparency hint and attribute.
- #define GLFW_HOVERED 0x0002000B

  Mouse cursor hover window attribute.
- #define GLFW_FOCUS_ON_SHOW 0x0002000C

  Input focus on calling show window hint and attribute.
- #define GLFW_MOUSE_PASSTHROUGH 0x0002000D

  Mouse input transparency window hint and attribute.
- #define GLFW_RED_BITS 0x00021001

  Framebuffer bit depth hint.

- #define GLFW_GREEN_BITS 0x00021002

  Framebuffer bit depth hint.
- #define GLFW_BLUE_BITS 0x00021003

  Framebuffer bit depth hint.
- #define GLFW_ALPHA_BITS 0x00021004

  Framebuffer bit depth hint.
- #define GLFW_DEPTH_BITS 0x00021005

  Framebuffer bit depth hint.
- #define GLFW_STENCIL_BITS 0x00021006

  Framebuffer bit depth hint.
- #define GLFW_ACCUM_RED_BITS 0x00021007

  Framebuffer bit depth hint.
- #define GLFW_ACCUM_GREEN_BITS 0x00021008

  Framebuffer bit depth hint.
- #define GLFW_ACCUM_BLUE_BITS 0x00021009

  Framebuffer bit depth hint.
- #define GLFW_ACCUM_ALPHA_BITS 0x0002100A

  Framebuffer bit depth hint.
- #define GLFW_AUX_BUFFERS 0x0002100B

  Framebuffer auxiliary buffer hint.
- #define GLFW_STEREO 0x0002100C

  OpenGL stereoscopic rendering hint.
- #define GLFW_SAMPLES 0x0002100D

  Framebuffer MSAA samples hint.
- #define GLFW_SRGB_CAPABLE 0x0002100E

  Framebuffer sRGB hint.
- #define GLFW_REFRESH_RATE 0x0002100F

  Monitor refresh rate hint.
- #define GLFW_DOUBLEBUFFER 0x00021010

  Framebuffer double buffering hint and attribute.
- #define GLFW_CLIENT_API 0x00022001

  Context client API hint and attribute.
- #define GLFW_CONTEXT_VERSION_MAJOR 0x00022002

  Context client API major version hint and attribute.
- #define GLFW_CONTEXT_VERSION_MINOR 0x00022003

  Context client API minor version hint and attribute.
- #define GLFW_CONTEXT_REVISION 0x00022004

  Context client API revision number attribute.
- #define GLFW_CONTEXT_ROBUSTNESS 0x00022005

  Context robustness hint and attribute.
- #define GLFW_OPENGL_FORWARD_COMPAT 0x00022006

  OpenGL forward-compatibility hint and attribute.
- #define GLFW_CONTEXT_DEBUG 0x00022007

  Debug mode context hint and attribute.
- #define GLFW_OPENGL_DEBUG_CONTEXT GLFW_CONTEXT_DEBUG

  Legacy name for compatibility.
- #define GLFW_OPENGL_PROFILE 0x00022008

  OpenGL profile hint and attribute.
- #define GLFW_CONTEXT_RELEASE_BEHAVIOR 0x00022009

  Context flush-on-release hint and attribute.
- #define GLFW_CONTEXT_NO_ERROR 0x0002200A

Context error suppression hint and attribute.

- #define GLFW_CONTEXT_CREATION_API 0x0002200B

  Context creation API hint and attribute.
- #define **GLFW_SCALE_TO_MONITOR** 0x0002200C

  Window content area scaling window window hint.
- #define **GLFW_COCOA_RETINA_FRAMEBUFFER** 0x00023001

  macOS specific window hint.
- #define **GLFW_COCOA_FRAME_NAME** 0x00023002

  macOS specific window hint.
- #define **GLFW_COCOA_GRAPHICS_SWITCHING** 0x00023003

  macOS specific window hint.
- #define **GLFW_X11_CLASS_NAME** 0x00024001

  X11 specific window hint.
- #define **GLFW_X11_INSTANCE_NAME** 0x00024002

  X11 specific window hint.
- #define **GLFW_WIN32_KEYBOARD_MENU** 0x00025001

Typedefs

- typedef struct GLFWwindow GLFWwindow

  Opaque window object.
- typedef void(∗ GLFWwindowposfun) (GLFWwindow ∗window, int xpos, int ypos)

  The function pointer type for window position callbacks.
- typedef void(∗ GLFWwindowsizefun) (GLFWwindow ∗window, int width, int height)

  The function pointer type for window size callbacks.
- typedef void(∗ GLFWwindowclosefun) (GLFWwindow ∗window)

  The function pointer type for window close callbacks.
- typedef void(∗ GLFWwindowrefreshfun) (GLFWwindow ∗window)

  The function pointer type for window content refresh callbacks.
- typedef void(∗ GLFWwindowfocusfun) (GLFWwindow ∗window, int focused)

  The function pointer type for window focus callbacks.
- typedef void(∗ GLFWwindowiconifyfun) (GLFWwindow ∗window, int iconified)

  The function pointer type for window iconify callbacks.
- typedef void(∗ GLFWwindowmaximizefun) (GLFWwindow ∗window, int maximized)

  The function pointer type for window maximize callbacks.
- typedef void(∗ GLFWframebuffersizefun) (GLFWwindow ∗window, int width, int height)

  The function pointer type for framebuffer size callbacks.
- typedef void(∗ GLFWwindowcontentscalefun) (GLFWwindow ∗window, float xscale, float yscale)

  The function pointer type for window content scale callbacks.
- typedef struct GLFWimage GLFWimage

  Image data.

Functions

- GLFWAPI void glfwDefaultWindowHints (void)

  Resets all window hints to their default values.
- GLFWAPI void glfwWindowHint (int hint, int value)

  Sets the specified window hint to the desired value.
- GLFWAPI void glfwWindowHintString (int hint, const char *value)

  Sets the specified window hint to the desired value.
- GLFWAPI GLFWwindow * glfwCreateWindow (int width, int height, const char *title, GLFWmonitor *monitor, GLFWwindow *share)

  Creates a window and its associated context.
- GLFWAPI void glfwDestroyWindow (GLFWwindow *window)

  Destroys the specified window and its context.
- GLFWAPI int glfwWindowShouldClose (GLFWwindow *window)

  Checks the close flag of the specified window.
- GLFWAPI void glfwSetWindowShouldClose (GLFWwindow *window, int value)

  Sets the close flag of the specified window.
- GLFWAPI void glfwSetWindowTitle (GLFWwindow *window, const char *title)

  Sets the title of the specified window.
- GLFWAPI void glfwSetWindowIcon (GLFWwindow *window, int count, const GLFWimage *images)

  Sets the icon for the specified window.
- GLFWAPI void glfwGetWindowPos (GLFWwindow *window, int *xpos, int *ypos)

  Retrieves the position of the content area of the specified window.
- GLFWAPI void glfwSetWindowPos (GLFWwindow *window, int xpos, int ypos)

  Sets the position of the content area of the specified window.
- GLFWAPI void glfwGetWindowSize (GLFWwindow *window, int *width, int *height)

  Retrieves the size of the content area of the specified window.
- GLFWAPI void glfwSetWindowSizeLimits (GLFWwindow *window, int minwidth, int minheight, int maxwidth, int maxheight)

  Sets the size limits of the specified window.
- GLFWAPI void glfwSetWindowAspectRatio (GLFWwindow *window, int numer, int denom)

  Sets the aspect ratio of the specified window.
- GLFWAPI void glfwSetWindowSize (GLFWwindow *window, int width, int height)

  Sets the size of the content area of the specified window.
- GLFWAPI void glfwGetFramebufferSize (GLFWwindow *window, int *width, int *height)

  Retrieves the size of the framebuffer of the specified window.
- GLFWAPI void glfwGetWindowFrameSize (GLFWwindow *window, int *left, int *top, int *right, int *bottom)

  Retrieves the size of the frame of the window.
- GLFWAPI void glfwGetWindowContentScale (GLFWwindow *window, float *xscale, float *yscale)

  Retrieves the content scale for the specified window.
- GLFWAPI float glfwGetWindowOpacity (GLFWwindow *window)

  Returns the opacity of the whole window.
- GLFWAPI void glfwSetWindowOpacity (GLFWwindow *window, float opacity)

  Sets the opacity of the whole window.
- GLFWAPI void glfwIconifyWindow (GLFWwindow *window)

  Iconifies the specified window.
- GLFWAPI void glfwRestoreWindow (GLFWwindow *window)

  Restores the specified window.
- GLFWAPI void glfwMaximizeWindow (GLFWwindow *window)

  Maximizes the specified window.
- GLFWAPI void glfwShowWindow (GLFWwindow *window)

Makes the specified window visible.

- GLFWAPI void glfwHideWindow (GLFWwindow *window)

  Hides the specified window.

- GLFWAPI void glfwFocusWindow (GLFWwindow *window)

  Brings the specified window to front and sets input focus.

- GLFWAPI void glfwRequestWindowAttention (GLFWwindow *window)

  Requests user attention to the specified window.

- GLFWAPI GLFWmonitor * glfwGetWindowMonitor (GLFWwindow *window)

  Returns the monitor that the window uses for full screen mode.

- GLFWAPI void glfwSetWindowMonitor (GLFWwindow *window, GLFWmonitor *monitor, int xpos, int ypos, int width, int height, int refreshRate)

  Sets the mode, monitor, video mode and placement of a window.

- GLFWAPI int glfwGetWindowAttrib (GLFWwindow *window, int attrib)

  Returns an attribute of the specified window.

- GLFWAPI void glfwSetWindowAttrib (GLFWwindow *window, int attrib, int value)

  Sets an attribute of the specified window.

- GLFWAPI void glfwSetWindowUserPointer (GLFWwindow *window, void *pointer)

  Sets the user pointer of the specified window.

- GLFWAPI void * glfwGetWindowUserPointer (GLFWwindow *window)

  Returns the user pointer of the specified window.

- GLFWAPI GLFWwindowposfun glfwSetWindowPosCallback (GLFWwindow *window, GLFWwindowposfun callback)

  Sets the position callback for the specified window.

- GLFWAPI GLFWwindowsizefun glfwSetWindowSizeCallback (GLFWwindow *window, GLFWwindowsizefun callback)

  Sets the size callback for the specified window.

- GLFWAPI GLFWwindowclosefun glfwSetWindowCloseCallback (GLFWwindow *window, GLFWwindowclosefun callback)

  Sets the close callback for the specified window.

- GLFWAPI GLFWwindowrefreshfun glfwSetWindowRefreshCallback (GLFWwindow *window, GLFWwindowrefreshfun callback)

  Sets the refresh callback for the specified window.

- GLFWAPI GLFWwindowfocusfun glfwSetWindowFocusCallback (GLFWwindow *window, GLFWwindowfocusfun callback)

  Sets the focus callback for the specified window.

- GLFWAPI GLFWwindowiconifyfun glfwSetWindowIconifyCallback (GLFWwindow *window, GLFWwindowiconifyfun callback)

  Sets the iconify callback for the specified window.

- GLFWAPI GLFWwindowmaximizefun glfwSetWindowMaximizeCallback (GLFWwindow *window, GLFWwindowmaximizefun callback)

  Sets the maximize callback for the specified window.

- GLFWAPI GLFWframebuffersizefun glfwSetFramebufferSizeCallback (GLFWwindow *window, GLFWframebuffersizefun callback)

  Sets the framebuffer resize callback for the specified window.

- GLFWAPI GLFWwindowcontentscalefun glfwSetWindowContentScaleCallback (GLFWwindow *window, GLFWwindowcontentscalefun callback)

  Sets the window content scale callback for the specified window.

- GLFWAPI void glfwPollEvents (void)

  Processes all pending events.

- GLFWAPI void glfwWaitEvents (void)

  Waits until events are queued and processes them.

- GLFWAPI void glfwWaitEventsTimeout (double timeout)

Waits with timeout until events are queued and processes them.

- GLFWAPI void glfwPostEmptyEvent (void)

  Posts an empty event to the event queue.

- GLFWAPI void glfwSwapBuffers (GLFWwindow *window)

  Swaps the front and back buffers of the specified window.

25.6.1 Detailed Description

Functions and types related to windows.

This is the reference documentation for window related functions and types, including creation, deletion and event polling. For more task-oriented information, see the Window guide.

25.6.2 Macro Definition Documentation

25.6.2.1 GLFW_ACCUM_ALPHA_BITS

```
#define GLFW_ACCUM_ALPHA_BITS 0x0002100A
```

Framebuffer bit depth hint.

Framebuffer bit depth hint.

25.6.2.2 GLFW_ACCUM_BLUE_BITS

```
#define GLFW_ACCUM_BLUE_BITS 0x00021009
```

Framebuffer bit depth hint.

Framebuffer bit depth hint.

25.6.2.3 GLFW_ACCUM_GREEN_BITS

```
#define GLFW_ACCUM_GREEN_BITS 0x00021008
```

Framebuffer bit depth hint.

Framebuffer bit depth hint.

25.6.2.4 GLFW_ACCUM_RED_BITS

```
#define GLFW_ACCUM_RED_BITS 0x00021007
```

Framebuffer bit depth hint.

Framebuffer bit depth hint.

25.6.2.5 GLFW_ALPHA_BITS

```
#define GLFW_ALPHA_BITS 0x00021004
```

Framebuffer bit depth hint.

Framebuffer bit depth hint.

25.6.2.6 GLFW_AUTO_ICONIFY

```
#define GLFW_AUTO_ICONIFY 0x00020006
```

Window auto-iconification window hint and attribute.

Window auto-iconification window hint and window attribute.

25.6.2.7 GLFW_AUX_BUFFERS

```
#define GLFW_AUX_BUFFERS 0x0002100B
```

Framebuffer auxiliary buffer hint.

Framebuffer auxiliary buffer hint.

25.6.2.8 GLFW_BLUE_BITS

```
#define GLFW_BLUE_BITS 0x00021003
```

Framebuffer bit depth hint.

Framebuffer bit depth hint.

25.6.2.9 GLFW_CENTER_CURSOR

```
#define GLFW_CENTER_CURSOR 0x00020009
```

Cursor centering window hint.

Cursor centering window hint.

25.6.2.10 GLFW_CLIENT_API

```
#define GLFW_CLIENT_API 0x00022001
```

Context client API hint and attribute.

Context client API hint and attribute.

25.6.2.11 GLFW_CONTEXT_CREATION_API

```
#define GLFW_CONTEXT_CREATION_API 0x0002200B
```

Context creation API hint and attribute.

Context creation API hint and attribute.

25.6.2.12 GLFW_CONTEXT_DEBUG

```
#define GLFW_CONTEXT_DEBUG 0x00022007
```

Debug mode context hint and attribute.

Debug mode context hint and attribute.

25.6.2.13 GLFW_CONTEXT_NO_ERROR

```
#define GLFW_CONTEXT_NO_ERROR 0x0002200A
```

Context error suppression hint and attribute.

Context error suppression hint and attribute.

25.6.2.14 GLFW_CONTEXT_RELEASE_BEHAVIOR

```
#define GLFW_CONTEXT_RELEASE_BEHAVIOR 0x00022009
```

Context flush-on-release hint and attribute.

Context flush-on-release hint and attribute.

25.6.2.15 GLFW_CONTEXT_REVISION

```
#define GLFW_CONTEXT_REVISION 0x00022004
```

Context client API revision number attribute.

Context client API revision number attribute.

25.6.2.16 GLFW_CONTEXT_ROBUSTNESS

```
#define GLFW_CONTEXT_ROBUSTNESS 0x00022005
```

Context robustness hint and attribute.

Context client API revision number hint and attribute.

25.6.2.17 GLFW_CONTEXT_VERSION_MAJOR

```
#define GLFW_CONTEXT_VERSION_MAJOR 0x00022002
```

Context client API major version hint and attribute.

Context client API major version hint and attribute.

25.6.2.18 GLFW_CONTEXT_VERSION_MINOR

```
#define GLFW_CONTEXT_VERSION_MINOR 0x00022003
```

Context client API minor version hint and attribute.

Context client API minor version hint and attribute.

25.6.2.19 GLFW_DECORATED

```
#define GLFW_DECORATED 0x00020005
```

Window decoration window hint and attribute.

Window decoration window hint and window attribute.

25.6.2.20 GLFW_DEPTH_BITS

```
#define GLFW_DEPTH_BITS 0x00021005
```

Framebuffer bit depth hint.

Framebuffer bit depth hint.

25.6.2.21 GLFW_DOUBLEBUFFER

```
#define GLFW_DOUBLEBUFFER 0x00021010
```

Framebuffer double buffering hint and attribute.

Framebuffer double buffering hint and attribute.

25.6.2.22 GLFW_FLOATING

```
#define GLFW_FLOATING 0x00020007
```

Window decoration window hint and attribute.

Window decoration window hint and window attribute.

25.6.2.23 GLFW_FOCUS_ON_SHOW

```
#define GLFW_FOCUS_ON_SHOW 0x0002000C
```

Input focus on calling show window hint and attribute.

Input focus window hint or window attribute.

25.6.2.24 GLFW_FOCUSED

```
#define GLFW_FOCUSED 0x00020001
```

Input focus window hint and attribute.

Input focus window hint or window attribute.

25.6.2.25 GLFW_GREEN_BITS

```
#define GLFW_GREEN_BITS 0x00021002
```

Framebuffer bit depth hint.

Framebuffer bit depth hint.

25.6.2.26 GLFW_HOVERED

```
#define GLFW_HOVERED 0x0002000B
```

Mouse cursor hover window attribute.

Mouse cursor hover window attribute.

25.6.2.27 GLFW_ICONIFIED

```
#define GLFW_ICONIFIED 0x00020002
```

Window iconification window attribute.

Window iconification window attribute.

25.6.2.28 GLFW_MAXIMIZED

```
#define GLFW_MAXIMIZED 0x00020008
```

Window maximization window hint and attribute.

Window maximization window hint and window attribute.

25.6.2.29 GLFW_MOUSE_PASSTHROUGH

```
#define GLFW_MOUSE_PASSTHROUGH 0x0002000D
```

Mouse input transparency window hint and attribute.

Mouse input transparency window hint or window attribute.

25.6.2.30 GLFW_OPENGL_DEBUG_CONTEXT

```
#define GLFW_OPENGL_DEBUG_CONTEXT GLFW_CONTEXT_DEBUG
```

Legacy name for compatibility.

This is an alias for compatibility with earlier versions.

25.6.2.31 GLFW_OPENGL_FORWARD_COMPAT

```
#define GLFW_OPENGL_FORWARD_COMPAT 0x00022006
```

OpenGL forward-compatibility hint and attribute.

OpenGL forward-compatibility hint and attribute.

25.6.2.32 GLFW_OPENGL_PROFILE

```
#define GLFW_OPENGL_PROFILE 0x00022008
```

OpenGL profile hint and attribute.

OpenGL profile hint and attribute.

25.6.2.33 GLFW_RED_BITS

```
#define GLFW_RED_BITS 0x00021001
```

Framebuffer bit depth hint.

Framebuffer bit depth hint.

25.6.2.34 GLFW_REFRESH_RATE

```
#define GLFW_REFRESH_RATE 0x0002100F
```

Monitor refresh rate hint.

Monitor refresh rate hint.

25.6.2.35 GLFW_RESIZABLE

```
#define GLFW_RESIZABLE 0x00020003
```

Window resize-ability window hint and attribute.

Window resize-ability window hint and window attribute.

25.6.2.36 GLFW_SAMPLES

```
#define GLFW_SAMPLES 0x0002100D
```

Framebuffer MSAA samples hint.

Framebuffer MSAA samples hint.

25.6.2.37 GLFW_SRGB_CAPABLE

```
#define GLFW_SRGB_CAPABLE 0x0002100E
```

Framebuffer sRGB hint.

Framebuffer sRGB hint.

25.6.2.38 GLFW_STENCIL_BITS

```
#define GLFW_STENCIL_BITS 0x00021006
```

Framebuffer bit depth hint.

Framebuffer bit depth hint.

25.6.2.39 GLFW_STEREO

```
#define GLFW_STEREO 0x0002100C
```

OpenGL stereoscopic rendering hint.

OpenGL stereoscopic rendering hint.

25.6.2.40 GLFW_TRANSPARENT_FRAMEBUFFER

```
#define GLFW_TRANSPARENT_FRAMEBUFFER 0x0002000A
```

Window framebuffer transparency hint and attribute.

Window framebuffer transparency window hint and window attribute.

25.6.2.41 GLFW_VISIBLE

```
#define GLFW_VISIBLE 0x00020004
```

Window visibility window hint and attribute.

Window visibility window hint and window attribute.

25.6.3 Typedef Documentation

25.6.3.1 GLFWframebuffersizefun

```
typedef void(* GLFWframebuffersizefun) (GLFWwindow *window, int width, int height)
```

The function pointer type for framebuffer size callbacks.

This is the function pointer type for framebuffer size callbacks. A framebuffer size callback function has the following signature:

```
void function_name(GLFWwindow* window, int width, int height)
```

Parameters

in	*window*	The window whose framebuffer was resized.
in	*width*	The new width, in pixels, of the framebuffer.
in	*height*	The new height, in pixels, of the framebuffer.

See also

Framebuffer size

glfwSetFramebufferSizeCallback

Since

Added in version 3.0.

25.6.3.2 GLFWimage

```
typedef struct GLFWimage GLFWimage
```

Image data.

This describes a single 2D image. See the documentation for each related function what the expected pixel format is.

See also

Custom cursor creation

Window icon

Since

Added in version 2.1. @glfw3 Removed format and bytes-per-pixel members.

25.6.3.3 GLFWwindow

```
typedef struct GLFWwindow GLFWwindow
```

Opaque window object.

Opaque window object.

See also

Window objects

Since

Added in version 3.0.

25.6.3.4 GLFWwindowclosefun

```
typedef void(* GLFWwindowclosefun) (GLFWwindow *window)
```

The function pointer type for window close callbacks.

This is the function pointer type for window close callbacks. A window close callback function has the following signature:

```
void function_name(GLFWwindow* window)
```

Parameters

in	*window*	The window that the user attempted to close.

See also

Window closing and close flag

glfwSetWindowCloseCallback

Since

Added in version 2.5. @glfw3 Added window handle parameter.

25.6.3.5 GLFWwindowcontentscalefun

```
typedef void(* GLFWwindowcontentscalefun) (GLFWwindow *window, float xscale, float yscale)
```

The function pointer type for window content scale callbacks.

This is the function pointer type for window content scale callbacks. A window content scale callback function has the following signature:

```
void function_name(GLFWwindow* window, float xscale, float yscale)
```

Parameters

in	*window*	The window whose content scale changed.
in	*xscale*	The new x-axis content scale of the window.
in	*yscale*	The new y-axis content scale of the window.

See also

Window content scale

glfwSetWindowContentScaleCallback

Since

Added in version 3.3.

25.6.3.6 GLFWwindowfocusfun

```
typedef void(* GLFWwindowfocusfun) (GLFWwindow *window, int focused)
```

The function pointer type for window focus callbacks.

This is the function pointer type for window focus callbacks. A window focus callback function has the following signature:

```
void function_name(GLFWwindow* window, int focused)
```

Parameters

in	*window*	The window that gained or lost input focus.
in	*focused*	`GLFW_TRUE` if the window was given input focus, or `GLFW_FALSE` if it lost it.

See also

Window input focus

glfwSetWindowFocusCallback

Since

Added in version 3.0.

25.6.3.7 GLFWwindowiconifyfun

```
typedef void(* GLFWwindowiconifyfun) (GLFWwindow *window, int iconified)
```

The function pointer type for window iconify callbacks.

This is the function pointer type for window iconify callbacks. A window iconify callback function has the following signature:

```
void function_name(GLFWwindow* window, int iconified)
```

Parameters

in	*window*	The window that was iconified or restored.
in	*iconified*	GLFW_TRUE if the window was iconified, or GLFW_FALSE if it was restored.

See also

Window iconification

glfwSetWindowIconifyCallback

Since

Added in version 3.0.

25.6.3.8 GLFWwindowmaximizefun

```
typedef void(* GLFWwindowmaximizefun) (GLFWwindow *window, int maximized)
```

The function pointer type for window maximize callbacks.

This is the function pointer type for window maximize callbacks. A window maximize callback function has the following signature:

```
void function_name(GLFWwindow* window, int maximized)
```

Parameters

in	*window*	The window that was maximized or restored.
in	*maximized*	GLFW_TRUE if the window was maximized, or GLFW_FALSE if it was restored.

See also

Window maximization

glfwSetWindowMaximizeCallback

Since

Added in version 3.3.

25.6.3.9 GLFWwindowposfun

```
typedef void(* GLFWwindowposfun) (GLFWwindow *window, int xpos, int ypos)
```

The function pointer type for window position callbacks.

This is the function pointer type for window position callbacks. A window position callback function has the following signature:

```
void callback_name(GLFWwindow* window, int xpos, int ypos)
```

Parameters

in	*window*	The window that was moved.
in	*xpos*	The new x-coordinate, in screen coordinates, of the upper-left corner of the content area of the window.
in	*ypos*	The new y-coordinate, in screen coordinates, of the upper-left corner of the content area of the window.

See also

Window position

glfwSetWindowPosCallback

Since

Added in version 3.0.

25.6.3.10 GLFWwindowrefreshfun

```
typedef void(* GLFWwindowrefreshfun) (GLFWwindow *window)
```

The function pointer type for window content refresh callbacks.

This is the function pointer type for window content refresh callbacks. A window content refresh callback function has the following signature:

```
void function_name(GLFWwindow* window);
```

Parameters

in	*window*	The window whose content needs to be refreshed.

See also

Window damage and refresh

glfwSetWindowRefreshCallback

Since

Added in version 2.5. @glfw3 Added window handle parameter.

25.6.3.11 GLFWwindowsizefun

```
typedef void(* GLFWwindowsizefun) (GLFWwindow *window, int width, int height)
```

The function pointer type for window size callbacks.

This is the function pointer type for window size callbacks. A window size callback function has the following signature:

```
void callback_name(GLFWwindow* window, int width, int height)
```

Parameters

in	*window*	The window that was resized.
in	*width*	The new width, in screen coordinates, of the window.
in	*height*	The new height, in screen coordinates, of the window.

See also

Window size

glfwSetWindowSizeCallback

Since

Added in version 1.0. @glfw3 Added window handle parameter.

25.6.4 Function Documentation

25.6.4.1 glfwCreateWindow()

```
GLFWAPI GLFWwindow * glfwCreateWindow (
            int width,
            int height,
            const char * title,
            GLFWmonitor * monitor,
            GLFWwindow * share )
```

Creates a window and its associated context.

This function creates a window and its associated OpenGL or OpenGL ES context. Most of the options controlling how the window and its context should be created are specified with window hints.

Successful creation does not change which context is current. Before you can use the newly created context, you need to make it current. For information about the `share` parameter, see Context object sharing.

The created window, framebuffer and context may differ from what you requested, as not all parameters and hints are hard constraints. This includes the size of the window, especially for full screen windows. To query the actual attributes of the created window, framebuffer and context, see glfwGetWindowAttrib, glfwGetWindowSize and glfwGetFramebufferSize.

To create a full screen window, you need to specify the monitor the window will cover. If no monitor is specified, the window will be windowed mode. Unless you have a way for the user to choose a specific monitor, it is recommended that you pick the primary monitor. For more information on how to query connected monitors, see Retrieving monitors.

For full screen windows, the specified size becomes the resolution of the window's *desired video mode*. As long as a full screen window is not iconified, the supported video mode most closely matching the desired video mode is set for the specified monitor. For more information about full screen windows, including the creation of so called *windowed full screen* or *borderless full screen* windows, see "Windowed full screen" windows.

Once you have created the window, you can switch it between windowed and full screen mode with glfwSetWindowMonitor. This will not affect its OpenGL or OpenGL ES context.

By default, newly created windows use the placement recommended by the window system. To create the window at a specific position, make it initially invisible using the GLFW_VISIBLE window hint, set its position and then show it.

As long as at least one full screen window is not iconified, the screensaver is prohibited from starting.

Window systems put limits on window sizes. Very large or very small window dimensions may be overridden by the window system on creation. Check the actual size after creation.

The swap interval is not set during window creation and the initial value may vary depending on driver settings and defaults.

Parameters

in	*width*	The desired width, in screen coordinates, of the window. This must be greater than zero.
in	*height*	The desired height, in screen coordinates, of the window. This must be greater than zero.
in	*title*	The initial, UTF-8 encoded window title.
in	*monitor*	The monitor to use for full screen mode, or `NULL` for windowed mode.
in	*share*	The window whose context to share resources with, or `NULL` to not share resources.

Returns

The handle of the created window, or NULL if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_INVALID_ENUM, GLFW_INVALID_VALUE, GLFW_API_UNAVAILABLE, GLFW_VERSION_UNAVAILABLE, GLFW_FORMAT_UNAVAILABLE and GLFW_PLATFORM_ERROR.

Remarks

@win32 Window creation will fail if the Microsoft GDI software OpenGL implementation is the only one available.

@win32 If the executable has an icon resource named GLFW_ICON, it will be set as the initial icon for the window. If no such icon is present, the IDI_APPLICATION icon will be used instead. To set a different icon, see glfwSetWindowIcon.

@win32 The context to share resources with must not be current on any other thread.

@macos The OS only supports core profile contexts for OpenGL versions 3.2 and later. Before creating an OpenGL context of version 3.2 or later you must set the GLFW_OPENGL_PROFILE hint accordingly. OpenGL 3.0 and 3.1 contexts are not supported at all on macOS.

@macos The GLFW window has no icon, as it is not a document window, but the dock icon will be the same as the application bundle's icon. For more information on bundles, see the Bundle Programming Guide in the Mac Developer Library.

@macos On OS X 10.10 and later the window frame will not be rendered at full resolution on Retina displays unless the GLFW_COCOA_RETINA_FRAMEBUFFER hint is GLFW_TRUE and the NSHigh↩ ResolutionCapable key is enabled in the application bundle's Info.plist. For more information, see High Resolution Guidelines for OS X in the Mac Developer Library. The GLFW test and example programs use a custom Info.plist template for this, which can be found as CMake/Info.↩ plist.in in the source tree.

@macos When activating frame autosaving with GLFW_COCOA_FRAME_NAME, the specified window size and position may be overridden by previously saved values.

@x11 Some window managers will not respect the placement of initially hidden windows.

@x11 Due to the asynchronous nature of X11, it may take a moment for a window to reach its requested state. This means you may not be able to query the final size, position or other attributes directly after window creation.

@x11 The class part of the WM_CLASS window property will by default be set to the window title passed to this function. The instance part will use the contents of the RESOURCE_NAME environment variable, if present and not empty, or fall back to the window title. Set the GLFW_X11_CLASS_NAME and GLFW_X11_INSTANCE_NAME window hints to override this.

@wayland Compositors should implement the xdg-decoration protocol for GLFW to decorate the window properly. If this protocol isn't supported, or if the compositor prefers client-side decorations, a very simple fallback frame will be drawn using the wp_viewporter protocol. A compositor can still emit close, maximize or fullscreen events, using for instance a keybind mechanism. If neither of these protocols is supported, the window won't be decorated.

@wayland A full screen window will not attempt to change the mode, no matter what the requested size or refresh rate.

@wayland Screensaver inhibition requires the idle-inhibit protocol to be implemented in the user's compositor.

@thread_safety This function must only be called from the main thread.

See also

Window creation

glfwDestroyWindow

Since

Added in version 3.0. Replaces glfwOpenWindow.

25.6.4.2 glfwDefaultWindowHints()

```
GLFWAPI void glfwDefaultWindowHints (
            void  )
```

Resets all window hints to their default values.

This function resets all window hints to their default values.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Window creation hints
glfwWindowHint
glfwWindowHintString

Since

Added in version 3.0.

25.6.4.3 glfwDestroyWindow()

```
GLFWAPI void glfwDestroyWindow (
            GLFWwindow * window )
```

Destroys the specified window and its context.

This function destroys the specified window and its context. On calling this function, no further callbacks will be called for that window.

If the context of the specified window is current on the main thread, it is detached before being destroyed.

Parameters

in	*window*	The window to destroy.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

Note

The context of the specified window must not be current on any other thread when this function is called.

@reentrancy This function must not be called from a callback.

@thread_safety This function must only be called from the main thread.

See also

Window creation

glfwCreateWindow

Since

Added in version 3.0. Replaces `glfwCloseWindow`.

25.6.4.4 glfwFocusWindow()

```
GLFWAPI void glfwFocusWindow (
            GLFWwindow * window )
```

Brings the specified window to front and sets input focus.

This function brings the specified window to front and sets input focus. The window should already be visible and not iconified.

By default, both windowed and full screen mode windows are focused when initially created. Set the GLFW_FOCUSED to disable this behavior.

Also by default, windowed mode windows are focused when shown with glfwShowWindow. Set the GLFW_FOCUS_ON_SHOW to disable this behavior.

Do not use this function to steal focus from other applications unless you are certain that is what the user wants. Focus stealing can be extremely disruptive.

For a less disruptive way of getting the user's attention, see attention requests.

Parameters

in	*window*	The window to give input focus.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_PLATFORM_ERROR and GLFW_FEATURE_UNAVAILABLE (see remarks).

Remarks

@wayland It is not possible for an application to set the input focus. This function will emit GLFW_FEATURE_UNAVAILABLE.

@thread_safety This function must only be called from the main thread.

See also

Window input focus

Window attention request

Since

Added in version 3.2.

25.6.4.5 glfwGetFramebufferSize()

```
GLFWAPI void glfwGetFramebufferSize (
            GLFWwindow * window,
            int * width,
            int * height )
```

Retrieves the size of the framebuffer of the specified window.

This function retrieves the size, in pixels, of the framebuffer of the specified window. If you wish to retrieve the size of the window in screen coordinates, see glfwGetWindowSize.

Any or all of the size arguments may be NULL. If an error occurs, all non-NULL size arguments will be set to zero.

Parameters

in	*window*	The window whose framebuffer to query.
out	*width*	Where to store the width, in pixels, of the framebuffer, or NULL.
out	*height*	Where to store the height, in pixels, of the framebuffer, or NULL.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@thread_safety This function must only be called from the main thread.

See also

Framebuffer size

glfwSetFramebufferSizeCallback

Since

Added in version 3.0.

25.6.4.6 glfwGetWindowAttrib()

```
GLFWAPI int glfwGetWindowAttrib (
            GLFWwindow * window,
            int attrib )
```

Returns an attribute of the specified window.

This function returns the value of an attribute of the specified window or its OpenGL or OpenGL ES context.

Parameters

in	*window*	The window to query.
in	*attrib*	The window attribute whose value to return.

Returns

The value of the attribute, or zero if an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_INVALID_ENUM and GLFW_PLATFORM_ERROR.

Remarks

Framebuffer related hints are not window attributes. See Framebuffer related attributes for more information.

Zero is a valid value for many window and context related attributes so you cannot use a return value of zero as an indication of errors. However, this function should not fail as long as it is passed valid arguments and the library has been initialized.

@thread_safety This function must only be called from the main thread.

See also

Window attributes

glfwSetWindowAttrib

Since

Added in version 3.0. Replaces `glfwGetWindowParam` and `glfwGetGLVersion`.

25.6.4.7 glfwGetWindowContentScale()

```
GLFWAPI void glfwGetWindowContentScale (
            GLFWwindow * window,
            float * xscale,
            float * yscale )
```

Retrieves the content scale for the specified window.

This function retrieves the content scale for the specified window. The content scale is the ratio between the current DPI and the platform's default DPI. This is especially important for text and any UI elements. If the pixel dimensions of your UI scaled by this look appropriate on your machine then it should appear at a reasonable size on other machines regardless of their DPI and scaling settings. This relies on the system DPI and scaling settings being somewhat correct.

On platforms where each monitors can have its own content scale, the window content scale will depend on which monitor the system considers the window to be on.

Parameters

in	*window*	The window to query.
out	*xscale*	Where to store the x-axis content scale, or `NULL`.
out	*yscale*	Where to store the y-axis content scale, or `NULL`.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@thread_safety This function must only be called from the main thread.

See also

Window content scale

glfwSetWindowContentScaleCallback

glfwGetMonitorContentScale

Since

Added in version 3.3.

25.6.4.8 glfwGetWindowFrameSize()

```
GLFWAPI void glfwGetWindowFrameSize (
            GLFWwindow * window,
            int * left,
            int * top,
            int * right,
            int * bottom )
```

Retrieves the size of the frame of the window.

This function retrieves the size, in screen coordinates, of each edge of the frame of the specified window. This size includes the title bar, if the window has one. The size of the frame may vary depending on the window-related hints used to create it.

Because this function retrieves the size of each window frame edge and not the offset along a particular coordinate axis, the retrieved values will always be zero or positive.

Any or all of the size arguments may be NULL. If an error occurs, all non-NULL size arguments will be set to zero.

Parameters

in	*window*	The window whose frame size to query.
out	*left*	Where to store the size, in screen coordinates, of the left edge of the window frame, or NULL.
out	*top*	Where to store the size, in screen coordinates, of the top edge of the window frame, or NULL.
out	*right*	Where to store the size, in screen coordinates, of the right edge of the window frame, or NULL.
out	*bottom*	Where to store the size, in screen coordinates, of the bottom edge of the window frame, or NULL.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@thread_safety This function must only be called from the main thread.

See also

Window size

Since

Added in version 3.1.

25.6.4.9 glfwGetWindowMonitor()

```
GLFWAPI GLFWmonitor * glfwGetWindowMonitor (
            GLFWwindow * window )
```

Returns the monitor that the window uses for full screen mode.

This function returns the handle of the monitor that the specified window is in full screen on.

Parameters

in	*window*	The window to query.

Returns

The monitor, or `NULL` if the window is in windowed mode or an error occurred.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Window monitor

glfwSetWindowMonitor

Since

Added in version 3.0.

25.6.4.10 glfwGetWindowOpacity()

```
GLFWAPI float glfwGetWindowOpacity (
            GLFWwindow * window )
```

Returns the opacity of the whole window.

This function returns the opacity of the window, including any decorations.

The opacity (or alpha) value is a positive finite number between zero and one, where zero is fully transparent and one is fully opaque. If the system does not support whole window transparency, this function always returns one.

The initial opacity value for newly created windows is one.

Parameters

in	*window*	The window to query.

Returns

The opacity value of the specified window.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@thread_safety This function must only be called from the main thread.

See also

Window transparency

glfwSetWindowOpacity

Since

Added in version 3.3.

25.6.4.11 glfwGetWindowPos()

```
GLFWAPI void glfwGetWindowPos (
            GLFWwindow * window,
            int * xpos,
            int * ypos )
```

Retrieves the position of the content area of the specified window.

This function retrieves the position, in screen coordinates, of the upper-left corner of the content area of the specified window.

Any or all of the position arguments may be `NULL`. If an error occurs, all non-`NULL` position arguments will be set to zero.

Parameters

in	*window*	The window to query.
out	*xpos*	Where to store the x-coordinate of the upper-left corner of the content area, or `NULL`.
out	*ypos*	Where to store the y-coordinate of the upper-left corner of the content area, or `NULL`.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_PLATFORM_ERROR and GLFW_FEATURE_UNAVAILABLE (see remarks).

Remarks

@wayland There is no way for an application to retrieve the global position of its windows. This function will emit GLFW_FEATURE_UNAVAILABLE.

@thread_safety This function must only be called from the main thread.

See also

Window position

glfwSetWindowPos

Since

Added in version 3.0.

25.6.4.12 glfwGetWindowSize()

```
GLFWAPI void glfwGetWindowSize (
            GLFWwindow * window,
            int * width,
            int * height )
```

Retrieves the size of the content area of the specified window.

This function retrieves the size, in screen coordinates, of the content area of the specified window. If you wish to retrieve the size of the framebuffer of the window in pixels, see glfwGetFramebufferSize.

Any or all of the size arguments may be `NULL`. If an error occurs, all non-`NULL` size arguments will be set to zero.

Parameters

in	*window*	The window whose size to retrieve.
out	*width*	Where to store the width, in screen coordinates, of the content area, or `NULL`.
out	*height*	Where to store the height, in screen coordinates, of the content area, or `NULL`.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@thread_safety This function must only be called from the main thread.

See also

Window size

glfwSetWindowSize

Since

Added in version 1.0. @glfw3 Added window handle parameter.

25.6.4.13 glfwGetWindowUserPointer()

```
GLFWAPI void * glfwGetWindowUserPointer (
            GLFWwindow * window )
```

Returns the user pointer of the specified window.

This function returns the current value of the user-defined pointer of the specified window. The initial value is `NULL`.

Parameters

in	*window*	The window whose pointer to return.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function may be called from any thread. Access is not synchronized.

See also

User pointer

glfwSetWindowUserPointer

Since

Added in version 3.0.

25.6.4.14 glfwHideWindow()

```
GLFWAPI void glfwHideWindow (
            GLFWwindow * window )
```

Hides the specified window.

This function hides the specified window if it was previously visible. If the window is already hidden or is in full screen mode, this function does nothing.

Parameters

in	*window*	The window to hide.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@thread_safety This function must only be called from the main thread.

See also

Window visibility

glfwShowWindow

Since

Added in version 3.0.

25.6.4.15 glfwIconifyWindow()

```
GLFWAPI void glfwIconifyWindow (
            GLFWwindow * window )
```

Iconifies the specified window.

This function iconifies (minimizes) the specified window if it was previously restored. If the window is already iconified, this function does nothing.

If the specified window is a full screen window, the original monitor resolution is restored until the window is restored.

Parameters

in	*window*	The window to iconify.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

Remarks

@wayland Once a window is iconified, glfwRestoreWindow won't be able to restore it. This is a design decision of the xdg-shell protocol.

@thread_safety This function must only be called from the main thread.

See also

Window iconification

glfwRestoreWindow

glfwMaximizeWindow

Since

Added in version 2.1. @glfw3 Added window handle parameter.

25.6.4.16 glfwMaximizeWindow()

```
GLFWAPI void glfwMaximizeWindow (
            GLFWwindow * window )
```

Maximizes the specified window.

This function maximizes the specified window if it was previously not maximized. If the window is already maximized, this function does nothing.

If the specified window is a full screen window, this function does nothing.

Parameters

in	*window*	The window to maximize.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

Thread Safety

This function may only be called from the main thread.

See also

Window iconification

glfwIconifyWindow

glfwRestoreWindow

Since

Added in GLFW 3.2.

25.6.4.17 glfwPollEvents()

```
GLFWAPI void glfwPollEvents (
            void  )
```

Processes all pending events.

This function processes only those events that are already in the event queue and then returns immediately. Processing events will cause the window and input callbacks associated with those events to be called.

On some platforms, a window move, resize or menu operation will cause event processing to block. This is due to how event processing is designed on those platforms. You can use the window refresh callback to redraw the contents of your window when necessary during such operations.

Do not assume that callbacks you set will *only* be called in response to event processing functions like this one. While it is necessary to poll for events, window systems that require GLFW to register callbacks of its own can pass events to GLFW in response to many window system function calls. GLFW will pass those events on to the application callbacks before returning.

Event processing is not required for joystick input to work.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@reentrancy This function must not be called from a callback.

@thread_safety This function must only be called from the main thread.

See also

Event processing

glfwWaitEvents

glfwWaitEventsTimeout

Since

Added in version 1.0.

25.6.4.18 glfwPostEmptyEvent()

```
GLFWAPI void glfwPostEmptyEvent (
            void  )
```

Posts an empty event to the event queue.

This function posts an empty event from the current thread to the event queue, causing glfwWaitEvents or glfwWaitEventsTimeout to return.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@thread_safety This function may be called from any thread.

See also

Event processing

glfwWaitEvents

glfwWaitEventsTimeout

Since

Added in version 3.1.

25.6.4.19 glfwRequestWindowAttention()

```
GLFWAPI void glfwRequestWindowAttention (
            GLFWwindow * window )
```

Requests user attention to the specified window.

This function requests user attention to the specified window. On platforms where this is not supported, attention is requested to the application as a whole.

Once the user has given attention, usually by focusing the window or application, the system will end the request automatically.

Parameters

in	*window*	The window to request attention to.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

Remarks

@macos Attention is requested to the application as a whole, not the specific window.

@thread_safety This function must only be called from the main thread.

See also

Window attention request

Since

Added in version 3.3.

25.6.4.20 glfwRestoreWindow()

```
GLFWAPI void glfwRestoreWindow (
            GLFWwindow * window )
```

Restores the specified window.

This function restores the specified window if it was previously iconified (minimized) or maximized. If the window is already restored, this function does nothing.

If the specified window is a full screen window, the resolution chosen for the window is restored on the selected monitor.

Parameters

in	*window*	The window to restore.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@thread_safety This function must only be called from the main thread.

See also

Window iconification

glfwIconifyWindow

glfwMaximizeWindow

Since

Added in version 2.1. @glfw3 Added window handle parameter.

25.6.4.21 glfwSetFramebufferSizeCallback()

```
GLFWAPI GLFWframebuffersizefun glfwSetFramebufferSizeCallback (
            GLFWwindow * window,
            GLFWframebuffersizefun callback )
```

Sets the framebuffer resize callback for the specified window.

This function sets the framebuffer resize callback of the specified window, which is called when the framebuffer of the specified window is resized.

Parameters

in	*window*	The window whose callback to set.
in	*callback*	The new callback, or NULL to remove the currently set callback.

Returns

The previously set callback, or NULL if no callback was set or the library had not been initialized.

@callback_signature
```
void function_name(GLFWwindow* window, int width, int height)
```

For more information about the callback parameters, see the function pointer type.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Framebuffer size

Since

Added in version 3.0.

25.6.4.22 glfwSetWindowAspectRatio()

```
GLFWAPI void glfwSetWindowAspectRatio (
            GLFWwindow * window,
            int numer,
            int denom )
```

Sets the aspect ratio of the specified window.

This function sets the required aspect ratio of the content area of the specified window. If the window is full screen, the aspect ratio only takes effect once it is made windowed. If the window is not resizable, this function does nothing.

The aspect ratio is specified as a numerator and a denominator and both values must be greater than zero. For example, the common 16:9 aspect ratio is specified as 16 and 9, respectively.

If the numerator and denominator is set to GLFW_DONT_CARE then the aspect ratio limit is disabled.

The aspect ratio is applied immediately to a windowed mode window and may cause it to be resized.

Parameters

in	*window*	The window to set limits for.
in	*numer*	The numerator of the desired aspect ratio, or GLFW_DONT_CARE.
in	*denom*	The denominator of the desired aspect ratio, or GLFW_DONT_CARE.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_INVALID_VALUE and GLFW_PLATFORM_ERROR.

Remarks

If you set size limits and an aspect ratio that conflict, the results are undefined.

@wayland The aspect ratio will not be applied until the window is actually resized, either by the user or by the compositor.

@thread_safety This function must only be called from the main thread.

See also

Window size limits

glfwSetWindowSizeLimits

Since

Added in version 3.2.

25.6.4.23 glfwSetWindowAttrib()

```
GLFWAPI void glfwSetWindowAttrib (
            GLFWwindow * window,
            int attrib,
            int value )
```

Sets an attribute of the specified window.

This function sets the value of an attribute of the specified window.

The supported attributes are GLFW_DECORATED, GLFW_RESIZABLE, GLFW_FLOATING, GLFW_AUTO_ICONIFY and GLFW_FOCUS_ON_SHOW. GLFW_MOUSE_PASSTHROUGH

Some of these attributes are ignored for full screen windows. The new value will take effect if the window is later made windowed.

Some of these attributes are ignored for windowed mode windows. The new value will take effect if the window is later made full screen.

Parameters

in	*window*	The window to set the attribute for.
in	*attrib*	A supported window attribute.
in	*value*	GLFW_TRUE or GLFW_FALSE.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_INVALID_ENUM, GLFW_INVALID_VALUE and GLFW_PLATFORM_ERROR.

Remarks

Calling glfwGetWindowAttrib will always return the latest value, even if that value is ignored by the current mode of the window.

@thread_safety This function must only be called from the main thread.

See also

Window attributes

glfwGetWindowAttrib

Since

Added in version 3.3.

25.6.4.24 glfwSetWindowCloseCallback()

```
GLFWAPI GLFWwindowclosefun glfwSetWindowCloseCallback (
            GLFWwindow * window,
            GLFWwindowclosefun callback )
```

Sets the close callback for the specified window.

This function sets the close callback of the specified window, which is called when the user attempts to close the window, for example by clicking the close widget in the title bar.

The close flag is set before this callback is called, but you can modify it at any time with glfwSetWindowShouldClose.

The close callback is not triggered by glfwDestroyWindow.

Parameters

in	*window*	The window whose callback to set.
in	*callback*	The new callback, or NULL to remove the currently set callback.

Returns

The previously set callback, or NULL if no callback was set or the library had not been initialized.

@callback_signature
```
void function_name(GLFWwindow* window)
```

For more information about the callback parameters, see the function pointer type.

@errors Possible errors include GLFW_NOT_INITIALIZED.

Remarks

@macos Selecting Quit from the application menu will trigger the close callback for all windows.

@thread_safety This function must only be called from the main thread.

See also

Window closing and close flag

Since

Added in version 2.5. @glfw3 Added window handle parameter and return value.

25.6.4.25 glfwSetWindowContentScaleCallback()

```
GLFWAPI GLFWwindowcontentscalefun glfwSetWindowContentScaleCallback (
            GLFWwindow * window,
            GLFWwindowcontentscalefun callback )
```

Sets the window content scale callback for the specified window.

This function sets the window content scale callback of the specified window, which is called when the content scale of the specified window changes.

Parameters

in	*window*	The window whose callback to set.
in	*callback*	The new callback, or NULL to remove the currently set callback.

Returns

The previously set callback, or NULL if no callback was set or the library had not been initialized.

@callback_signature
```
void function_name(GLFWwindow* window, float xscale, float yscale)
```

For more information about the callback parameters, see the function pointer type.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Window content scale

glfwGetWindowContentScale

Since

Added in version 3.3.

25.6.4.26 glfwSetWindowFocusCallback()

```
GLFWAPI GLFWwindowfocusfun glfwSetWindowFocusCallback (
            GLFWwindow * window,
            GLFWwindowfocusfun callback )
```

Sets the focus callback for the specified window.

This function sets the focus callback of the specified window, which is called when the window gains or loses input focus.

After the focus callback is called for a window that lost input focus, synthetic key and mouse button release events will be generated for all such that had been pressed. For more information, see glfwSetKeyCallback and glfwSetMouseButtonCallback.

Parameters

in	*window*	The window whose callback to set.
in	*callback*	The new callback, or NULL to remove the currently set callback.

Returns

The previously set callback, or NULL if no callback was set or the library had not been initialized.

@callback_signature
```
void function_name(GLFWwindow* window, int focused)
```

For more information about the callback parameters, see the function pointer type.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Window input focus

Since

Added in version 3.0.

25.6.4.27 glfwSetWindowIcon()

```
GLFWAPI void glfwSetWindowIcon (
            GLFWwindow * window,
            int count,
            const GLFWimage * images )
```

Sets the icon for the specified window.

This function sets the icon of the specified window. If passed an array of candidate images, those of or closest to the sizes desired by the system are selected. If no images are specified, the window reverts to its default icon.

The pixels are 32-bit, little-endian, non-premultiplied RGBA, i.e. eight bits per channel with the red channel first. They are arranged canonically as packed sequential rows, starting from the top-left corner.

The desired image sizes varies depending on platform and system settings. The selected images will be rescaled as needed. Good sizes include 16x16, 32x32 and 48x48.

Parameters

in	*window*	The window whose icon to set.
in	*count*	The number of images in the specified array, or zero to revert to the default window icon.
in	*images*	The images to create the icon from. This is ignored if count is zero.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_INVALID_VALUE, GLFW_PLATFORM_ERROR and GLFW_FEATURE_UNAVAILABLE (see remarks).

@pointer_lifetime The specified image data is copied before this function returns.

Remarks

@macos Regular windows do not have icons on macOS. This function will emit GLFW_FEATURE_UNAVAILABLE. The dock icon will be the same as the application bundle's icon. For more information on bundles, see the Bundle Programming Guide in the Mac Developer Library.

@wayland There is no existing protocol to change an icon, the window will thus inherit the one defined in the application's desktop file. This function will emit GLFW_FEATURE_UNAVAILABLE.

@thread_safety This function must only be called from the main thread.

See also

Window icon

Since

Added in version 3.2.

25.6.4.28 glfwSetWindowIconifyCallback()

```
GLFWAPI GLFWwindowiconifyfun glfwSetWindowIconifyCallback (
            GLFWwindow * window,
            GLFWwindowiconifyfun callback )
```

Sets the iconify callback for the specified window.

This function sets the iconification callback of the specified window, which is called when the window is iconified or restored.

Parameters

in	*window*	The window whose callback to set.
in	*callback*	The new callback, or NULL to remove the currently set callback.

Returns

The previously set callback, or NULL if no callback was set or the library had not been initialized.

@callback_signature
```
void function_name(GLFWwindow* window, int iconified)
```

For more information about the callback parameters, see the function pointer type.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Window iconification

Since

Added in version 3.0.

25.6.4.29 glfwSetWindowMaximizeCallback()

```
GLFWAPI GLFWwindowmaximizefun glfwSetWindowMaximizeCallback (
            GLFWwindow * window,
            GLFWwindowmaximizefun callback )
```

Sets the maximize callback for the specified window.

This function sets the maximization callback of the specified window, which is called when the window is maximized or restored.

Parameters

in	*window*	The window whose callback to set.
in	*callback*	The new callback, or NULL to remove the currently set callback.

Returns

The previously set callback, or NULL if no callback was set or the library had not been initialized.

@callback_signature
```
void function_name(GLFWwindow* window, int maximized)
```

For more information about the callback parameters, see the function pointer type.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Window maximization

Since

Added in version 3.3.

25.6.4.30 glfwSetWindowMonitor()

```
GLFWAPI void glfwSetWindowMonitor (
            GLFWwindow * window,
            GLFWmonitor * monitor,
            int xpos,
            int ypos,
            int width,
            int height,
            int refreshRate )
```

Sets the mode, monitor, video mode and placement of a window.

This function sets the monitor that the window uses for full screen mode or, if the monitor is NULL, makes it windowed mode.

When setting a monitor, this function updates the width, height and refresh rate of the desired video mode and switches to the video mode closest to it. The window position is ignored when setting a monitor.

When the monitor is NULL, the position, width and height are used to place the window content area. The refresh rate is ignored when no monitor is specified.

If you only wish to update the resolution of a full screen window or the size of a windowed mode window, see glfwSetWindowSize.

When a window transitions from full screen to windowed mode, this function restores any previous window settings such as whether it is decorated, floating, resizable, has size or aspect ratio limits, etc.

Parameters

in	*window*	The window whose monitor, size or video mode to set.
in	*monitor*	The desired monitor, or NULL to set windowed mode.
in	*xpos*	The desired x-coordinate of the upper-left corner of the content area.
in	*ypos*	The desired y-coordinate of the upper-left corner of the content area.
in	*width*	The desired with, in screen coordinates, of the content area or video mode.
in	*height*	The desired height, in screen coordinates, of the content area or video mode.
in	*refreshRate*	The desired refresh rate, in Hz, of the video mode, or GLFW_DONT_CARE.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

Remarks

The OpenGL or OpenGL ES context will not be destroyed or otherwise affected by any resizing or mode switching, although you may need to update your viewport if the framebuffer size has changed.

@wayland The desired window position is ignored, as there is no way for an application to set this property.

@wayland Setting the window to full screen will not attempt to change the mode, no matter what the requested size or refresh rate.

@thread_safety This function must only be called from the main thread.

See also

Window monitor

Full screen windows

glfwGetWindowMonitor

glfwSetWindowSize

Since

Added in version 3.2.

25.6.4.31 glfwSetWindowOpacity()

```
GLFWAPI void glfwSetWindowOpacity (
            GLFWwindow * window,
            float opacity )
```

Sets the opacity of the whole window.

This function sets the opacity of the window, including any decorations.

The opacity (or alpha) value is a positive finite number between zero and one, where zero is fully transparent and one is fully opaque.

The initial opacity value for newly created windows is one.

A window created with framebuffer transparency may not use whole window transparency. The results of doing this are undefined.

Parameters

in	*window*	The window to set the opacity for.
in	*opacity*	The desired opacity of the specified window.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_PLATFORM_ERROR and GLFW_FEATURE_UNAVAILABLE (see remarks).

Remarks

@wayland There is no way to set an opacity factor for a window. This function will emit GLFW_FEATURE_UNAVAILABLE.

@thread_safety This function must only be called from the main thread.

See also

Window transparency

glfwGetWindowOpacity

Since

Added in version 3.3.

25.6.4.32 glfwSetWindowPos()

```
GLFWAPI void glfwSetWindowPos (
            GLFWwindow * window,
            int xpos,
            int ypos )
```

Sets the position of the content area of the specified window.

This function sets the position, in screen coordinates, of the upper-left corner of the content area of the specified windowed mode window. If the window is a full screen window, this function does nothing.

Do not use this function to move an already visible window unless you have very good reasons for doing so, as it will confuse and annoy the user.

The window manager may put limits on what positions are allowed. GLFW cannot and should not override these limits.

Parameters

in	*window*	The window to query.
in	*xpos*	The x-coordinate of the upper-left corner of the content area.
in	*ypos*	The y-coordinate of the upper-left corner of the content area.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_PLATFORM_ERROR and GLFW_FEATURE_UNAVAILABLE (see remarks).

Remarks

@wayland There is no way for an application to set the global position of its windows. This function will emit GLFW_FEATURE_UNAVAILABLE.

@thread_safety This function must only be called from the main thread.

See also

Window position

glfwGetWindowPos

Since

Added in version 1.0. @glfw3 Added window handle parameter.

25.6.4.33 glfwSetWindowPosCallback()

```
GLFWAPI GLFWwindowposfun glfwSetWindowPosCallback (
            GLFWwindow * window,
            GLFWwindowposfun callback )
```

Sets the position callback for the specified window.

This function sets the position callback of the specified window, which is called when the window is moved. The callback is provided with the position, in screen coordinates, of the upper-left corner of the content area of the window.

Parameters

in	*window*	The window whose callback to set.
in	*callback*	The new callback, or NULL to remove the currently set callback.

Returns

The previously set callback, or NULL if no callback was set or the library had not been initialized.

@callback_signature
```
void function_name(GLFWwindow* window, int xpos, int ypos)
```

For more information about the callback parameters, see the function pointer type.

@errors Possible errors include GLFW_NOT_INITIALIZED.

Remarks

@wayland This callback will never be called, as there is no way for an application to know its global position.

@thread_safety This function must only be called from the main thread.

See also

Window position

Since

Added in version 3.0.

25.6.4.34 glfwSetWindowRefreshCallback()

```
GLFWAPI GLFWwindowrefreshfun glfwSetWindowRefreshCallback (
             GLFWwindow * window,
             GLFWwindowrefreshfun callback )
```

Sets the refresh callback for the specified window.

This function sets the refresh callback of the specified window, which is called when the content area of the window needs to be redrawn, for example if the window has been exposed after having been covered by another window.

On compositing window systems such as Aero, Compiz, Aqua or Wayland, where the window contents are saved off-screen, this callback may be called only very infrequently or never at all.

Parameters

in	*window*	The window whose callback to set.
in	*callback*	The new callback, or NULL to remove the currently set callback.

Returns

The previously set callback, or `NULL` if no callback was set or the library had not been initialized.

@callback_signature
```
void function_name(GLFWwindow* window);
```

For more information about the callback parameters, see the function pointer type.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Window damage and refresh

Since

Added in version 2.5. @glfw3 Added window handle parameter and return value.

25.6.4.35 glfwSetWindowShouldClose()

```
GLFWAPI void glfwSetWindowShouldClose (
            GLFWwindow * window,
            int value )
```

Sets the close flag of the specified window.

This function sets the value of the close flag of the specified window. This can be used to override the user's attempt to close the window, or to signal that it should be closed.

Parameters

in	*window*	The window whose flag to change.
in	*value*	The new value.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function may be called from any thread. Access is not synchronized.

See also

Window closing and close flag

Since

Added in version 3.0.

25.6.4.36 glfwSetWindowSize()

```
GLFWAPI void glfwSetWindowSize (
            GLFWwindow * window,
            int width,
            int height )
```

Sets the size of the content area of the specified window.

This function sets the size, in screen coordinates, of the content area of the specified window.

For full screen windows, this function updates the resolution of its desired video mode and switches to the video mode closest to it, without affecting the window's context. As the context is unaffected, the bit depths of the framebuffer remain unchanged.

If you wish to update the refresh rate of the desired video mode in addition to its resolution, see glfwSetWindowMonitor.

The window manager may put limits on what sizes are allowed. GLFW cannot and should not override these limits.

Parameters

in	*window*	The window to resize.
in	*width*	The desired width, in screen coordinates, of the window content area.
in	*height*	The desired height, in screen coordinates, of the window content area.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

Remarks

@wayland A full screen window will not attempt to change the mode, no matter what the requested size.

@thread_safety This function must only be called from the main thread.

See also

Window size
glfwGetWindowSize
glfwSetWindowMonitor

Since

Added in version 1.0. @glfw3 Added window handle parameter.

25.6.4.37 glfwSetWindowSizeCallback()

```
GLFWAPI GLFWwindowsizefun glfwSetWindowSizeCallback (
            GLFWwindow * window,
            GLFWwindowsizefun callback )
```

Sets the size callback for the specified window.

This function sets the size callback of the specified window, which is called when the window is resized. The callback is provided with the size, in screen coordinates, of the content area of the window.

Parameters

in	*window*	The window whose callback to set.
in	*callback*	The new callback, or NULL to remove the currently set callback.

Returns

The previously set callback, or NULL if no callback was set or the library had not been initialized.

@callback_signature
```
void function_name(GLFWwindow* window, int width, int height)
```

For more information about the callback parameters, see the function pointer type.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function must only be called from the main thread.

See also

Window size

Since

Added in version 1.0. @glfw3 Added window handle parameter and return value.

25.6.4.38 glfwSetWindowSizeLimits()

```
GLFWAPI void glfwSetWindowSizeLimits (
            GLFWwindow * window,
            int minwidth,
            int minheight,
            int maxwidth,
            int maxheight )
```

Sets the size limits of the specified window.

This function sets the size limits of the content area of the specified window. If the window is full screen, the size limits only take effect once it is made windowed. If the window is not resizable, this function does nothing.

The size limits are applied immediately to a windowed mode window and may cause it to be resized.

The maximum dimensions must be greater than or equal to the minimum dimensions and all must be greater than or equal to zero.

Parameters

in	*window*	The window to set limits for.
in	*minwidth*	The minimum width, in screen coordinates, of the content area, or GLFW_DONT_CARE.
in	*minheight*	The minimum height, in screen coordinates, of the content area, or GLFW_DONT_CARE.
in	*maxwidth*	The maximum width, in screen coordinates, of the content area, or GLFW_DONT_CARE.
in	*maxheight*	The maximum height, in screen coordinates, of the content area, or GLFW_DONT_CARE.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_INVALID_VALUE and GLFW_PLATFORM_ERROR.

Remarks

If you set size limits and an aspect ratio that conflict, the results are undefined.

@wayland The size limits will not be applied until the window is actually resized, either by the user or by the compositor.

@thread_safety This function must only be called from the main thread.

See also

Window size limits

glfwSetWindowAspectRatio

Since

Added in version 3.2.

25.6.4.39 glfwSetWindowTitle()

```
GLFWAPI void glfwSetWindowTitle (
            GLFWwindow * window,
            const char * title )
```

Sets the title of the specified window.

This function sets the window title, encoded as UTF-8, of the specified window.

Parameters

in	*window*	The window whose title to change.
in	*title*	The UTF-8 encoded window title.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

Remarks

@macos The window title will not be updated until the next time you process events.

@thread_safety This function must only be called from the main thread.

See also

Window title

Since

Added in version 1.0. @glfw3 Added window handle parameter.

25.6.4.40 glfwSetWindowUserPointer()

```
GLFWAPI void glfwSetWindowUserPointer (
            GLFWwindow * window,
            void * pointer )
```

Sets the user pointer of the specified window.

This function sets the user-defined pointer of the specified window. The current value is retained until the window is destroyed. The initial value is NULL.

Parameters

in	*window*	The window whose pointer to set.
in	*pointer*	The new value.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function may be called from any thread. Access is not synchronized.

See also

User pointer

glfwGetWindowUserPointer

Since

Added in version 3.0.

25.6.4.41 glfwShowWindow()

```
GLFWAPI void glfwShowWindow (
            GLFWwindow * window )
```

Makes the specified window visible.

This function makes the specified window visible if it was previously hidden. If the window is already visible or is in full screen mode, this function does nothing.

By default, windowed mode windows are focused when shown Set the GLFW_FOCUS_ON_SHOW window hint to change this behavior for all newly created windows, or change the behavior for an existing window with glfwSetWindowAttrib.

Parameters

in	*window*	The window to make visible.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

Remarks

@wayland Because Wayland wants every frame of the desktop to be complete, this function does not immediately make the window visible. Instead it will become visible the next time the window framebuffer is updated after this call.

@thread_safety This function must only be called from the main thread.

See also

Window visibility

glfwHideWindow

Since

Added in version 3.0.

25.6.4.42 glfwSwapBuffers()

```
GLFWAPI void glfwSwapBuffers (
            GLFWwindow * window )
```

Swaps the front and back buffers of the specified window.

This function swaps the front and back buffers of the specified window when rendering with OpenGL or OpenGL ES. If the swap interval is greater than zero, the GPU driver waits the specified number of screen updates before swapping the buffers.

The specified window must have an OpenGL or OpenGL ES context. Specifying a window without a context will generate a GLFW_NO_WINDOW_CONTEXT error.

This function does not apply to Vulkan. If you are rendering with Vulkan, see `vkQueuePresentKHR` instead.

Parameters

in	*window*	The window whose buffers to swap.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_NO_WINDOW_CONTEXT and GLFW_PLATFORM_ERROR.

Remarks

EGL: The context of the specified window must be current on the calling thread.

@thread_safety This function may be called from any thread.

See also

Buffer swapping

glfwSwapInterval

Since

Added in version 1.0. @glfw3 Added window handle parameter.

25.6.4.43 glfwWaitEvents()

```
GLFWAPI void glfwWaitEvents (
            void  )
```

Waits until events are queued and processes them.

This function puts the calling thread to sleep until at least one event is available in the event queue. Once one or more events are available, it behaves exactly like glfwPollEvents, i.e. the events in the queue are processed and the function then returns immediately. Processing events will cause the window and input callbacks associated with those events to be called.

Since not all events are associated with callbacks, this function may return without a callback having been called even if you are monitoring all callbacks.

On some platforms, a window move, resize or menu operation will cause event processing to block. This is due to how event processing is designed on those platforms. You can use the window refresh callback to redraw the contents of your window when necessary during such operations.

Do not assume that callbacks you set will *only* be called in response to event processing functions like this one. While it is necessary to poll for events, window systems that require GLFW to register callbacks of its own can pass events to GLFW in response to many window system function calls. GLFW will pass those events on to the application callbacks before returning.

Event processing is not required for joystick input to work.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_PLATFORM_ERROR.

@reentrancy This function must not be called from a callback.

@thread_safety This function must only be called from the main thread.

See also

Event processing

glfwPollEvents

glfwWaitEventsTimeout

Since

Added in version 2.5.

25.6.4.44 glfwWaitEventsTimeout()

```
GLFWAPI void glfwWaitEventsTimeout (
            double timeout )
```

Waits with timeout until events are queued and processes them.

This function puts the calling thread to sleep until at least one event is available in the event queue, or until the specified timeout is reached. If one or more events are available, it behaves exactly like glfwPollEvents, i.e. the events in the queue are processed and the function then returns immediately. Processing events will cause the window and input callbacks associated with those events to be called.

The timeout value must be a positive finite number.

Since not all events are associated with callbacks, this function may return without a callback having been called even if you are monitoring all callbacks.

On some platforms, a window move, resize or menu operation will cause event processing to block. This is due to how event processing is designed on those platforms. You can use the window refresh callback to redraw the contents of your window when necessary during such operations.

Do not assume that callbacks you set will *only* be called in response to event processing functions like this one. While it is necessary to poll for events, window systems that require GLFW to register callbacks of its own can pass events to GLFW in response to many window system function calls. GLFW will pass those events on to the application callbacks before returning.

Event processing is not required for joystick input to work.

Parameters

in	*timeout*	The maximum amount of time, in seconds, to wait.

@errors Possible errors include GLFW_NOT_INITIALIZED, GLFW_INVALID_VALUE and GLFW_PLATFORM_ERROR.

@reentrancy This function must not be called from a callback.

@thread_safety This function must only be called from the main thread.

See also

Event processing

glfwPollEvents

glfwWaitEvents

Since

Added in version 3.2.

25.6.4.45 glfwWindowHint()

```
GLFWAPI void glfwWindowHint (
            int hint,
            int value )
```

Sets the specified window hint to the desired value.

This function sets hints for the next call to glfwCreateWindow. The hints, once set, retain their values until changed by a call to this function or glfwDefaultWindowHints, or until the library is terminated.

Only integer value hints can be set with this function. String value hints are set with glfwWindowHintString.

This function does not check whether the specified hint values are valid. If you set hints to invalid values this will instead be reported by the next call to glfwCreateWindow.

Some hints are platform specific. These may be set on any platform but they will only affect their specific platform. Other platforms will ignore them. Setting these hints requires no platform specific headers or functions.

Parameters

in	*hint*	The window hint to set.
in	*value*	The new value of the window hint.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_INVALID_ENUM.

@thread_safety This function must only be called from the main thread.

See also

Window creation hints

glfwWindowHintString

glfwDefaultWindowHints

Since

Added in version 3.0. Replaces `glfwOpenWindowHint`.

25.6.4.46 glfwWindowHintString()

```
GLFWAPI void glfwWindowHintString (
            int hint,
            const char * value )
```

Sets the specified window hint to the desired value.

This function sets hints for the next call to glfwCreateWindow. The hints, once set, retain their values until changed by a call to this function or glfwDefaultWindowHints, or until the library is terminated.

Only string type hints can be set with this function. Integer value hints are set with glfwWindowHint.

This function does not check whether the specified hint values are valid. If you set hints to invalid values this will instead be reported by the next call to glfwCreateWindow.

Some hints are platform specific. These may be set on any platform but they will only affect their specific platform. Other platforms will ignore them. Setting these hints requires no platform specific headers or functions.

Parameters

in	*hint*	The window hint to set.
in	*value*	The new value of the window hint.

@errors Possible errors include GLFW_NOT_INITIALIZED and GLFW_INVALID_ENUM.

@pointer_lifetime The specified string is copied before this function returns.

@thread_safety This function must only be called from the main thread.

See also

Window creation hints

glfwWindowHint

glfwDefaultWindowHints

Since

Added in version 3.3.

25.6.4.47 glfwWindowShouldClose()

```
GLFWAPI int glfwWindowShouldClose (
            GLFWwindow * window )
```

Checks the close flag of the specified window.

This function returns the value of the close flag of the specified window.

Parameters

in	*window*	The window to query.

Returns

The value of the close flag.

@errors Possible errors include GLFW_NOT_INITIALIZED.

@thread_safety This function may be called from any thread. Access is not synchronized.

See also

Window closing and close flag

Since

Added in version 3.0.

25.7 Joystick hat states

Joystick hat states.

Macros

- #define **GLFW_HAT_CENTERED** 0
- #define **GLFW_HAT_UP** 1
- #define **GLFW_HAT_RIGHT** 2
- #define **GLFW_HAT_DOWN** 4
- #define **GLFW_HAT_LEFT** 8
- #define **GLFW_HAT_RIGHT_UP** (GLFW_HAT_RIGHT | GLFW_HAT_UP)
- #define **GLFW_HAT_RIGHT_DOWN** (GLFW_HAT_RIGHT | GLFW_HAT_DOWN)
- #define **GLFW_HAT_LEFT_UP** (GLFW_HAT_LEFT | GLFW_HAT_UP)
- #define **GLFW_HAT_LEFT_DOWN** (GLFW_HAT_LEFT | GLFW_HAT_DOWN)

25.7.1 Detailed Description

Joystick hat states.

See joystick hat input for how these are used.

25.8 Keyboard keys

Keyboard key IDs.

Macros

- #define **GLFW_KEY_UNKNOWN** -1
- #define **GLFW_KEY_SPACE** 32
- #define **GLFW_KEY_APOSTROPHE** 39 /* ' */
- #define **GLFW_KEY_COMMA** 44 /* , */
- #define **GLFW_KEY_MINUS** 45 /* - */
- #define **GLFW_KEY_PERIOD** 46 /* . */
- #define **GLFW_KEY_SLASH** 47 /* / */
- #define **GLFW_KEY_0** 48
- #define **GLFW_KEY_1** 49
- #define **GLFW_KEY_2** 50
- #define **GLFW_KEY_3** 51
- #define **GLFW_KEY_4** 52
- #define **GLFW_KEY_5** 53
- #define **GLFW_KEY_6** 54
- #define **GLFW_KEY_7** 55
- #define **GLFW_KEY_8** 56
- #define **GLFW_KEY_9** 57
- #define **GLFW_KEY_SEMICOLON** 59 /* ; */
- #define **GLFW_KEY_EQUAL** 61 /* = */
- #define **GLFW_KEY_A** 65

- #define **GLFW_KEY_B** 66
- #define **GLFW_KEY_C** 67
- #define **GLFW_KEY_D** 68
- #define **GLFW_KEY_E** 69
- #define **GLFW_KEY_F** 70
- #define **GLFW_KEY_G** 71
- #define **GLFW_KEY_H** 72
- #define **GLFW_KEY_I** 73
- #define **GLFW_KEY_J** 74
- #define **GLFW_KEY_K** 75
- #define **GLFW_KEY_L** 76
- #define **GLFW_KEY_M** 77
- #define **GLFW_KEY_N** 78
- #define **GLFW_KEY_O** 79
- #define **GLFW_KEY_P** 80
- #define **GLFW_KEY_Q** 81
- #define **GLFW_KEY_R** 82
- #define **GLFW_KEY_S** 83
- #define **GLFW_KEY_T** 84
- #define **GLFW_KEY_U** 85
- #define **GLFW_KEY_V** 86
- #define **GLFW_KEY_W** 87
- #define **GLFW_KEY_X** 88
- #define **GLFW_KEY_Y** 89
- #define **GLFW_KEY_Z** 90
- #define **GLFW_KEY_LEFT_BRACKET** 91 /* [*/
- #define **GLFW_KEY_BACKSLASH** 92 /* \ */
- #define **GLFW_KEY_RIGHT_BRACKET** 93 /*] */
- #define **GLFW_KEY_GRAVE_ACCENT** 96 /* ` */
- #define **GLFW_KEY_WORLD_1** 161 /* non-US #1 */
- #define **GLFW_KEY_WORLD_2** 162 /* non-US #2 */
- #define **GLFW_KEY_ESCAPE** 256
- #define **GLFW_KEY_ENTER** 257
- #define **GLFW_KEY_TAB** 258
- #define **GLFW_KEY_BACKSPACE** 259
- #define **GLFW_KEY_INSERT** 260
- #define **GLFW_KEY_DELETE** 261
- #define **GLFW_KEY_RIGHT** 262
- #define **GLFW_KEY_LEFT** 263
- #define **GLFW_KEY_DOWN** 264
- #define **GLFW_KEY_UP** 265
- #define **GLFW_KEY_PAGE_UP** 266
- #define **GLFW_KEY_PAGE_DOWN** 267
- #define **GLFW_KEY_HOME** 268
- #define **GLFW_KEY_END** 269
- #define **GLFW_KEY_CAPS_LOCK** 280
- #define **GLFW_KEY_SCROLL_LOCK** 281
- #define **GLFW_KEY_NUM_LOCK** 282
- #define **GLFW_KEY_PRINT_SCREEN** 283
- #define **GLFW_KEY_PAUSE** 284
- #define **GLFW_KEY_F1** 290
- #define **GLFW_KEY_F2** 291
- #define **GLFW_KEY_F3** 292
- #define **GLFW_KEY_F4** 293
- #define **GLFW_KEY_F5** 294

- #define **GLFW_KEY_F6** 295
- #define **GLFW_KEY_F7** 296
- #define **GLFW_KEY_F8** 297
- #define **GLFW_KEY_F9** 298
- #define **GLFW_KEY_F10** 299
- #define **GLFW_KEY_F11** 300
- #define **GLFW_KEY_F12** 301
- #define **GLFW_KEY_F13** 302
- #define **GLFW_KEY_F14** 303
- #define **GLFW_KEY_F15** 304
- #define **GLFW_KEY_F16** 305
- #define **GLFW_KEY_F17** 306
- #define **GLFW_KEY_F18** 307
- #define **GLFW_KEY_F19** 308
- #define **GLFW_KEY_F20** 309
- #define **GLFW_KEY_F21** 310
- #define **GLFW_KEY_F22** 311
- #define **GLFW_KEY_F23** 312
- #define **GLFW_KEY_F24** 313
- #define **GLFW_KEY_F25** 314
- #define **GLFW_KEY_KP_0** 320
- #define **GLFW_KEY_KP_1** 321
- #define **GLFW_KEY_KP_2** 322
- #define **GLFW_KEY_KP_3** 323
- #define **GLFW_KEY_KP_4** 324
- #define **GLFW_KEY_KP_5** 325
- #define **GLFW_KEY_KP_6** 326
- #define **GLFW_KEY_KP_7** 327
- #define **GLFW_KEY_KP_8** 328
- #define **GLFW_KEY_KP_9** 329
- #define **GLFW_KEY_KP_DECIMAL** 330
- #define **GLFW_KEY_KP_DIVIDE** 331
- #define **GLFW_KEY_KP_MULTIPLY** 332
- #define **GLFW_KEY_KP_SUBTRACT** 333
- #define **GLFW_KEY_KP_ADD** 334
- #define **GLFW_KEY_KP_ENTER** 335
- #define **GLFW_KEY_KP_EQUAL** 336
- #define **GLFW_KEY_LEFT_SHIFT** 340
- #define **GLFW_KEY_LEFT_CONTROL** 341
- #define **GLFW_KEY_LEFT_ALT** 342
- #define **GLFW_KEY_LEFT_SUPER** 343
- #define **GLFW_KEY_RIGHT_SHIFT** 344
- #define **GLFW_KEY_RIGHT_CONTROL** 345
- #define **GLFW_KEY_RIGHT_ALT** 346
- #define **GLFW_KEY_RIGHT_SUPER** 347
- #define **GLFW_KEY_MENU** 348
- #define **GLFW_KEY_LAST** GLFW_KEY_MENU

25.8.1 Detailed Description

Keyboard key IDs.

See key input for how these are used.

These key codes are inspired by the *USB HID Usage Tables v1.12* (p. 53-60), but re-arranged to map to 7-bit ASCII for printable keys (function keys are put in the 256+ range).

The naming of the key codes follow these rules:

- The US keyboard layout is used
- Names of printable alphanumeric characters are used (e.g. "A", "R", "3", etc.)
- For non-alphanumeric characters, Unicode:ish names are used (e.g. "COMMA", "LEFT_SQUARE_↩ BRACKET", etc.). Note that some names do not correspond to the Unicode standard (usually for brevity)
- Keys that lack a clear US mapping are named "WORLD_x"
- For non-printable keys, custom names are used (e.g. "F4", "BACKSPACE", etc.)

25.9 Modifier key flags

Modifier key flags.

Macros

- #define GLFW_MOD_SHIFT 0x0001

  If this bit is set one or more Shift keys were held down.
- #define GLFW_MOD_CONTROL 0x0002

  If this bit is set one or more Control keys were held down.
- #define GLFW_MOD_ALT 0x0004

  If this bit is set one or more Alt keys were held down.
- #define GLFW_MOD_SUPER 0x0008

  If this bit is set one or more Super keys were held down.
- #define GLFW_MOD_CAPS_LOCK 0x0010

  If this bit is set the Caps Lock key is enabled.
- #define GLFW_MOD_NUM_LOCK 0x0020

  If this bit is set the Num Lock key is enabled.

25.9.1 Detailed Description

Modifier key flags.

See key input for how these are used.

25.9.2 Macro Definition Documentation

25.9.2.1 GLFW_MOD_ALT

```
#define GLFW_MOD_ALT 0x0004
```

If this bit is set one or more Alt keys were held down.

If this bit is set one or more Alt keys were held down.

25.9.2.2 GLFW_MOD_CAPS_LOCK

```
#define GLFW_MOD_CAPS_LOCK 0x0010
```

If this bit is set the Caps Lock key is enabled.

If this bit is set the Caps Lock key is enabled and the GLFW_LOCK_KEY_MODS input mode is set.

25.9.2.3 GLFW_MOD_CONTROL

```
#define GLFW_MOD_CONTROL 0x0002
```

If this bit is set one or more Control keys were held down.

If this bit is set one or more Control keys were held down.

25.9.2.4 GLFW_MOD_NUM_LOCK

```
#define GLFW_MOD_NUM_LOCK 0x0020
```

If this bit is set the Num Lock key is enabled.

If this bit is set the Num Lock key is enabled and the GLFW_LOCK_KEY_MODS input mode is set.

25.9.2.5 GLFW_MOD_SHIFT

```
#define GLFW_MOD_SHIFT 0x0001
```

If this bit is set one or more Shift keys were held down.

If this bit is set one or more Shift keys were held down.

25.9.2.6 GLFW_MOD_SUPER

```
#define GLFW_MOD_SUPER 0x0008
```

If this bit is set one or more Super keys were held down.

If this bit is set one or more Super keys were held down.

25.10 Mouse buttons

Mouse button IDs.

Macros

- #define **GLFW_MOUSE_BUTTON_1** 0
- #define **GLFW_MOUSE_BUTTON_2** 1
- #define **GLFW_MOUSE_BUTTON_3** 2
- #define **GLFW_MOUSE_BUTTON_4** 3
- #define **GLFW_MOUSE_BUTTON_5** 4
- #define **GLFW_MOUSE_BUTTON_6** 5
- #define **GLFW_MOUSE_BUTTON_7** 6
- #define **GLFW_MOUSE_BUTTON_8** 7
- #define **GLFW_MOUSE_BUTTON_LAST** GLFW_MOUSE_BUTTON_8
- #define **GLFW_MOUSE_BUTTON_LEFT** GLFW_MOUSE_BUTTON_1
- #define **GLFW_MOUSE_BUTTON_RIGHT** GLFW_MOUSE_BUTTON_2
- #define **GLFW_MOUSE_BUTTON_MIDDLE** GLFW_MOUSE_BUTTON_3

25.10.1 Detailed Description

Mouse button IDs.

See mouse button input for how these are used.

25.11 Joysticks

Joystick IDs.

Macros

- #define **GLFW_JOYSTICK_1** 0
- #define **GLFW_JOYSTICK_2** 1
- #define **GLFW_JOYSTICK_3** 2
- #define **GLFW_JOYSTICK_4** 3
- #define **GLFW_JOYSTICK_5** 4
- #define **GLFW_JOYSTICK_6** 5
- #define **GLFW_JOYSTICK_7** 6
- #define **GLFW_JOYSTICK_8** 7
- #define **GLFW_JOYSTICK_9** 8
- #define **GLFW_JOYSTICK_10** 9
- #define **GLFW_JOYSTICK_11** 10
- #define **GLFW_JOYSTICK_12** 11
- #define **GLFW_JOYSTICK_13** 12
- #define **GLFW_JOYSTICK_14** 13
- #define **GLFW_JOYSTICK_15** 14
- #define **GLFW_JOYSTICK_16** 15
- #define **GLFW_JOYSTICK_LAST** GLFW_JOYSTICK_16

25.11.1 Detailed Description

Joystick IDs.

See joystick input for how these are used.

25.12 Gamepad buttons

Gamepad buttons.

Macros

- #define **GLFW_GAMEPAD_BUTTON_A** 0
- #define **GLFW_GAMEPAD_BUTTON_B** 1
- #define **GLFW_GAMEPAD_BUTTON_X** 2
- #define **GLFW_GAMEPAD_BUTTON_Y** 3
- #define **GLFW_GAMEPAD_BUTTON_LEFT_BUMPER** 4
- #define **GLFW_GAMEPAD_BUTTON_RIGHT_BUMPER** 5
- #define **GLFW_GAMEPAD_BUTTON_BACK** 6
- #define **GLFW_GAMEPAD_BUTTON_START** 7
- #define **GLFW_GAMEPAD_BUTTON_GUIDE** 8
- #define **GLFW_GAMEPAD_BUTTON_LEFT_THUMB** 9
- #define **GLFW_GAMEPAD_BUTTON_RIGHT_THUMB** 10
- #define **GLFW_GAMEPAD_BUTTON_DPAD_UP** 11
- #define **GLFW_GAMEPAD_BUTTON_DPAD_RIGHT** 12
- #define **GLFW_GAMEPAD_BUTTON_DPAD_DOWN** 13
- #define **GLFW_GAMEPAD_BUTTON_DPAD_LEFT** 14
- #define **GLFW_GAMEPAD_BUTTON_LAST** GLFW_GAMEPAD_BUTTON_DPAD_LEFT
- #define **GLFW_GAMEPAD_BUTTON_CROSS** GLFW_GAMEPAD_BUTTON_A
- #define **GLFW_GAMEPAD_BUTTON_CIRCLE** GLFW_GAMEPAD_BUTTON_B
- #define **GLFW_GAMEPAD_BUTTON_SQUARE** GLFW_GAMEPAD_BUTTON_X
- #define **GLFW_GAMEPAD_BUTTON_TRIANGLE** GLFW_GAMEPAD_BUTTON_Y

25.12.1 Detailed Description

Gamepad buttons.

See Gamepad input for how these are used.

25.13 Gamepad axes

Gamepad axes.

Macros

- #define **GLFW_GAMEPAD_AXIS_LEFT_X** 0
- #define **GLFW_GAMEPAD_AXIS_LEFT_Y** 1
- #define **GLFW_GAMEPAD_AXIS_RIGHT_X** 2
- #define **GLFW_GAMEPAD_AXIS_RIGHT_Y** 3
- #define **GLFW_GAMEPAD_AXIS_LEFT_TRIGGER** 4
- #define **GLFW_GAMEPAD_AXIS_RIGHT_TRIGGER** 5
- #define **GLFW_GAMEPAD_AXIS_LAST** GLFW_GAMEPAD_AXIS_RIGHT_TRIGGER

25.13.1 Detailed Description

Gamepad axes.

See Gamepad input for how these are used.

25.14 Error codes

Error codes.

Macros

- #define GLFW_NO_ERROR 0

  No error has occurred.
- #define GLFW_NOT_INITIALIZED 0x00010001

  GLFW has not been initialized.
- #define GLFW_NO_CURRENT_CONTEXT 0x00010002

  No context is current for this thread.
- #define GLFW_INVALID_ENUM 0x00010003

  One of the arguments to the function was an invalid enum value.
- #define GLFW_INVALID_VALUE 0x00010004

  One of the arguments to the function was an invalid value.
- #define GLFW_OUT_OF_MEMORY 0x00010005

  A memory allocation failed.
- #define GLFW_API_UNAVAILABLE 0x00010006

  GLFW could not find support for the requested API on the system.
- #define GLFW_VERSION_UNAVAILABLE 0x00010007

  The requested OpenGL or OpenGL ES version is not available.
- #define GLFW_PLATFORM_ERROR 0x00010008

  A platform-specific error occurred that does not match any of the more specific categories.
- #define GLFW_FORMAT_UNAVAILABLE 0x00010009

  The requested format is not supported or available.
- #define GLFW_NO_WINDOW_CONTEXT 0x0001000A

  The specified window does not have an OpenGL or OpenGL ES context.
- #define GLFW_CURSOR_UNAVAILABLE 0x0001000B

  The specified cursor shape is not available.
- #define GLFW_FEATURE_UNAVAILABLE 0x0001000C

  The requested feature is not provided by the platform.
- #define GLFW_FEATURE_UNIMPLEMENTED 0x0001000D

  The requested feature is not implemented for the platform.
- #define GLFW_PLATFORM_UNAVAILABLE 0x0001000E

  Platform unavailable or no matching platform was found.

25.14.1 Detailed Description

Error codes.

See error handling for how these are used.

25.14.2 Macro Definition Documentation

25.14.2.1 GLFW_API_UNAVAILABLE

```
#define GLFW_API_UNAVAILABLE 0x00010006
```

GLFW could not find support for the requested API on the system.

GLFW could not find support for the requested API on the system.

@analysis The installed graphics driver does not support the requested API, or does not support it via the chosen context creation API. Below are a few examples.

> Some pre-installed Windows graphics drivers do not support OpenGL. AMD only supports OpenGL ES via EGL, while Nvidia and Intel only support it via a WGL or GLX extension. macOS does not provide OpenGL ES at all. The Mesa EGL, OpenGL and OpenGL ES libraries do not interface with the Nvidia binary driver. Older graphics drivers do not support Vulkan.

25.14.2.2 GLFW_CURSOR_UNAVAILABLE

```
#define GLFW_CURSOR_UNAVAILABLE 0x0001000B
```

The specified cursor shape is not available.

The specified standard cursor shape is not available, either because the current platform cursor theme does not provide it or because it is not available on the platform.

@analysis Platform or system settings limitation. Pick another standard cursor shape or create a custom cursor.

25.14.2.3 GLFW_FEATURE_UNAVAILABLE

```
#define GLFW_FEATURE_UNAVAILABLE 0x0001000C
```

The requested feature is not provided by the platform.

The requested feature is not provided by the platform, so GLFW is unable to implement it. The documentation for each function notes if it could emit this error.

@analysis Platform or platform version limitation. The error can be ignored unless the feature is critical to the application.

> A function call that emits this error has no effect other than the error and updating any existing out parameters.

25.14.2.4 GLFW_FEATURE_UNIMPLEMENTED

```
#define GLFW_FEATURE_UNIMPLEMENTED 0x0001000D
```

The requested feature is not implemented for the platform.

The requested feature has not yet been implemented in GLFW for this platform.

@analysis An incomplete implementation of GLFW for this platform, hopefully fixed in a future release. The error can be ignored unless the feature is critical to the application.

A function call that emits this error has no effect other than the error and updating any existing out parameters.

25.14.2.5 GLFW_FORMAT_UNAVAILABLE

```
#define GLFW_FORMAT_UNAVAILABLE 0x00010009
```

The requested format is not supported or available.

If emitted during window creation, the requested pixel format is not supported.

If emitted when querying the clipboard, the contents of the clipboard could not be converted to the requested format.

@analysis If emitted during window creation, one or more hard constraints did not match any of the available pixel formats. If your application is sufficiently flexible, downgrade your requirements and try again. Otherwise, inform the user that their machine does not match your requirements.

If emitted when querying the clipboard, ignore the error or report it to the user, as appropriate.

25.14.2.6 GLFW_INVALID_ENUM

```
#define GLFW_INVALID_ENUM 0x00010003
```

One of the arguments to the function was an invalid enum value.

One of the arguments to the function was an invalid enum value, for example requesting GLFW_RED_BITS with glfwGetWindowAttrib.

@analysis Application programmer error. Fix the offending call.

25.14.2.7 GLFW_INVALID_VALUE

```
#define GLFW_INVALID_VALUE 0x00010004
```

One of the arguments to the function was an invalid value.

One of the arguments to the function was an invalid value, for example requesting a non-existent OpenGL or OpenGL ES version like 2.7.

Requesting a valid but unavailable OpenGL or OpenGL ES version will instead result in a GLFW_VERSION_UNAVAILABLE error.

@analysis Application programmer error. Fix the offending call.

25.14.2.8 GLFW_NO_CURRENT_CONTEXT

```
#define GLFW_NO_CURRENT_CONTEXT 0x00010002
```

No context is current for this thread.

This occurs if a GLFW function was called that needs and operates on the current OpenGL or OpenGL ES context but no context is current on the calling thread. One such function is glfwSwapInterval.

@analysis Application programmer error. Ensure a context is current before calling functions that require a current context.

25.14.2.9 GLFW_NO_ERROR

```
#define GLFW_NO_ERROR 0
```

No error has occurred.

No error has occurred.

@analysis Yay.

25.14.2.10 GLFW_NO_WINDOW_CONTEXT

```
#define GLFW_NO_WINDOW_CONTEXT 0x0001000A
```

The specified window does not have an OpenGL or OpenGL ES context.

A window that does not have an OpenGL or OpenGL ES context was passed to a function that requires it to have one.

@analysis Application programmer error. Fix the offending call.

25.14.2.11 GLFW_NOT_INITIALIZED

```
#define GLFW_NOT_INITIALIZED 0x00010001
```

GLFW has not been initialized.

This occurs if a GLFW function was called that must not be called unless the library is initialized.

@analysis Application programmer error. Initialize GLFW before calling any function that requires initialization.

25.14.2.12 GLFW_OUT_OF_MEMORY

```
#define GLFW_OUT_OF_MEMORY 0x00010005
```

A memory allocation failed.

A memory allocation failed.

@analysis A bug in GLFW or the underlying operating system. Report the bug to our issue tracker.

25.14.2.13 GLFW_PLATFORM_ERROR

```
#define GLFW_PLATFORM_ERROR 0x00010008
```

A platform-specific error occurred that does not match any of the more specific categories.

A platform-specific error occurred that does not match any of the more specific categories.

@analysis A bug or configuration error in GLFW, the underlying operating system or its drivers, or a lack of required resources. Report the issue to our issue tracker.

25.14.2.14 GLFW_PLATFORM_UNAVAILABLE

```
#define GLFW_PLATFORM_UNAVAILABLE 0x0001000E
```

Platform unavailable or no matching platform was found.

If emitted during initialization, no matching platform was found. If GLFW_PLATFORM is set to GLFW_ANY_↩PLATFORM, GLFW could not detect any of the platforms supported by this library binary, except for the Null platform. If set to a specific platform, it is either not supported by this library binary or GLFW was not able to detect it.

If emitted by a native access function, GLFW was initialized for a different platform than the function is for.

@analysis Failure to detect any platform usually only happens on non-macOS Unix systems, either when no window system is running or the program was run from a terminal that does not have the necessary environment variables. Fall back to a different platform if possible or notify the user that no usable platform was detected.

Failure to detect a specific platform may have the same cause as above or be because support for that platform was not compiled in. Call glfwPlatformSupported to check whether a specific platform is supported by a library binary.

25.14.2.15 GLFW_VERSION_UNAVAILABLE

```
#define GLFW_VERSION_UNAVAILABLE 0x00010007
```

The requested OpenGL or OpenGL ES version is not available.

The requested OpenGL or OpenGL ES version (including any requested context or framebuffer hints) is not available on this machine.

@analysis The machine does not support your requirements. If your application is sufficiently flexible, downgrade your requirements and try again. Otherwise, inform the user that their machine does not match your requirements.

> Future invalid OpenGL and OpenGL ES versions, for example OpenGL 4.8 if 5.0 comes out before the 4.x series gets that far, also fail with this error and not GLFW_INVALID_VALUE, because GLFW cannot know what future versions will exist.

25.15 Standard cursor shapes

Standard system cursor shapes.

Macros

- #define GLFW_ARROW_CURSOR 0x00036001

  The regular arrow cursor shape.
- #define GLFW_IBEAM_CURSOR 0x00036002

  The text input I-beam cursor shape.
- #define GLFW_CROSSHAIR_CURSOR 0x00036003

  The crosshair cursor shape.
- #define GLFW_POINTING_HAND_CURSOR 0x00036004

  The pointing hand cursor shape.
- #define GLFW_RESIZE_EW_CURSOR 0x00036005

  The horizontal resize/move arrow shape.
- #define GLFW_RESIZE_NS_CURSOR 0x00036006

  The vertical resize/move arrow shape.
- #define GLFW_RESIZE_NWSE_CURSOR 0x00036007

  The top-left to bottom-right diagonal resize/move arrow shape.
- #define GLFW_RESIZE_NESW_CURSOR 0x00036008

  The top-right to bottom-left diagonal resize/move arrow shape.
- #define GLFW_RESIZE_ALL_CURSOR 0x00036009

  The omni-directional resize/move cursor shape.
- #define GLFW_NOT_ALLOWED_CURSOR 0x0003600A

  The operation-not-allowed shape.
- #define GLFW_HRESIZE_CURSOR GLFW_RESIZE_EW_CURSOR

  Legacy name for compatibility.
- #define GLFW_VRESIZE_CURSOR GLFW_RESIZE_NS_CURSOR

  Legacy name for compatibility.
- #define GLFW_HAND_CURSOR GLFW_POINTING_HAND_CURSOR

  Legacy name for compatibility.

25.15.1 Detailed Description

Standard system cursor shapes.

These are the standard cursor shapes that can be requested from the platform (window system).

25.15.2 Macro Definition Documentation

25.15.2.1 GLFW_ARROW_CURSOR

```
#define GLFW_ARROW_CURSOR 0x00036001
```

The regular arrow cursor shape.

The regular arrow cursor shape.

25.15.2.2 GLFW_CROSSHAIR_CURSOR

```
#define GLFW_CROSSHAIR_CURSOR 0x00036003
```

The crosshair cursor shape.

The crosshair cursor shape.

25.15.2.3 GLFW_HAND_CURSOR

```
#define GLFW_HAND_CURSOR GLFW_POINTING_HAND_CURSOR
```

Legacy name for compatibility.

This is an alias for compatibility with earlier versions.

25.15.2.4 GLFW_HRESIZE_CURSOR

```
#define GLFW_HRESIZE_CURSOR GLFW_RESIZE_EW_CURSOR
```

Legacy name for compatibility.

This is an alias for compatibility with earlier versions.

25.15.2.5 GLFW_IBEAM_CURSOR

```
#define GLFW_IBEAM_CURSOR 0x00036002
```

The text input I-beam cursor shape.

The text input I-beam cursor shape.

25.15.2.6 GLFW_NOT_ALLOWED_CURSOR

```
#define GLFW_NOT_ALLOWED_CURSOR 0x0003600A
```

The operation-not-allowed shape.

The operation-not-allowed shape. This is usually a circle with a diagonal line through it.

Note

@x11 This shape is provided by a newer standard not supported by all cursor themes.

@wayland This shape is provided by a newer standard not supported by all cursor themes.

25.15.2.7 GLFW_POINTING_HAND_CURSOR

```
#define GLFW_POINTING_HAND_CURSOR 0x00036004
```

The pointing hand cursor shape.

The pointing hand cursor shape.

25.15.2.8 GLFW_RESIZE_ALL_CURSOR

```
#define GLFW_RESIZE_ALL_CURSOR 0x00036009
```

The omni-directional resize/move cursor shape.

The omni-directional resize cursor/move shape. This is usually either a combined horizontal and vertical double-headed arrow or a grabbing hand.

25.15.2.9 GLFW_RESIZE_EW_CURSOR

```
#define GLFW_RESIZE_EW_CURSOR 0x00036005
```

The horizontal resize/move arrow shape.

The horizontal resize/move arrow shape. This is usually a horizontal double-headed arrow.

25.15.2.10 GLFW_RESIZE_NESW_CURSOR

```
#define GLFW_RESIZE_NESW_CURSOR 0x00036008
```

The top-right to bottom-left diagonal resize/move arrow shape.

The top-right to bottom-left diagonal resize/move shape. This is usually a diagonal double-headed arrow.

Note

@macos This shape is provided by a private system API and may fail with GLFW_CURSOR_UNAVAILABLE in the future.

@x11 This shape is provided by a newer standard not supported by all cursor themes.

@wayland This shape is provided by a newer standard not supported by all cursor themes.

25.15.2.11 GLFW_RESIZE_NS_CURSOR

```
#define GLFW_RESIZE_NS_CURSOR 0x00036006
```

The vertical resize/move arrow shape.

The vertical resize/move shape. This is usually a vertical double-headed arrow.

25.15.2.12 GLFW_RESIZE_NWSE_CURSOR

```
#define GLFW_RESIZE_NWSE_CURSOR 0x00036007
```

The top-left to bottom-right diagonal resize/move arrow shape.

The top-left to bottom-right diagonal resize/move shape. This is usually a diagonal double-headed arrow.

Note

@macos This shape is provided by a private system API and may fail with GLFW_CURSOR_UNAVAILABLE in the future.

@x11 This shape is provided by a newer standard not supported by all cursor themes.

@wayland This shape is provided by a newer standard not supported by all cursor themes.

25.15.2.13 GLFW_VRESIZE_CURSOR

```
#define GLFW_VRESIZE_CURSOR GLFW_RESIZE_NS_CURSOR
```

Legacy name for compatibility.

This is an alias for compatibility with earlier versions.

25.16 Native access

Functions related to accessing native handles.

Functions related to accessing native handles.

By using the native access functions you assert that you know what you're doing and how to fix problems caused by using them. If you don't, you shouldn't be using them.

Before the inclusion of glfw3native.h, you may define zero or more window system API macro and zero or more context creation API macros.

The chosen backends must match those the library was compiled for. Failure to do this will cause a link-time error.

The available window API macros are:

- GLFW_EXPOSE_NATIVE_WIN32

- GLFW_EXPOSE_NATIVE_COCOA
- GLFW_EXPOSE_NATIVE_X11
- GLFW_EXPOSE_NATIVE_WAYLAND

The available context API macros are:

- GLFW_EXPOSE_NATIVE_WGL
- GLFW_EXPOSE_NATIVE_NSGL
- GLFW_EXPOSE_NATIVE_GLX
- GLFW_EXPOSE_NATIVE_EGL
- GLFW_EXPOSE_NATIVE_OSMESA

These macros select which of the native access functions that are declared and which platform-specific headers to include. It is then up your (by definition platform-specific) code to handle which of these should be defined.

Chapter 26

Class Documentation

26.1 _CPOINT Struct Reference

Public Attributes

- LONG **IP**
- DWORD **dwLog**

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

- lib/glfw/deps/mingw/dinput.h

26.2 _DIACTIONA Struct Reference

Public Attributes

- UINT_PTR **uAppData**
- DWORD **dwSemantic**
- DWORD **dwFlags**
-

  union {
    LPCSTR **lptszActionName**
    UINT **uResIdString**
  } **DUMMYUNIONNAME**

- GUID **guidInstance**
- DWORD **dwObjID**
- DWORD **dwHow**

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

- lib/glfw/deps/mingw/dinput.h

26.3 _DIACTIONFORMATA Struct Reference

Public Attributes

- DWORD **dwSize**
- DWORD **dwActionSize**
- DWORD **dwDataSize**
- DWORD **dwNumActions**
- LPDIACTIONA **rgoAction**
- GUID **guidActionMap**
- DWORD **dwGenre**
- DWORD **dwBufferSize**
- LONG **lAxisMin**
- LONG **lAxisMax**
- HINSTANCE **hInstString**
- FILETIME **ftTimeStamp**
- DWORD **dwCRC**
- CHAR **tszActionMap** [MAX_PATH]

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

- lib/glfw/deps/mingw/dinput.h

26.4 _DIACTIONFORMATW Struct Reference

Public Attributes

- DWORD **dwSize**
- DWORD **dwActionSize**
- DWORD **dwDataSize**
- DWORD **dwNumActions**
- LPDIACTIONW **rgoAction**
- GUID **guidActionMap**
- DWORD **dwGenre**
- DWORD **dwBufferSize**
- LONG **lAxisMin**
- LONG **lAxisMax**
- HINSTANCE **hInstString**
- FILETIME **ftTimeStamp**
- DWORD **dwCRC**
- WCHAR **tszActionMap** [MAX_PATH]

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

- lib/glfw/deps/mingw/dinput.h

26.5 _DIACTIONW Struct Reference

Public Attributes

- UINT_PTR **uAppData**
- DWORD **dwSemantic**
- DWORD **dwFlags**
- union {
  LPCWSTR **lptszActionName**
  UINT **uResIdString**
  } **DUMMYUNIONNAME**
- GUID **guidInstance**
- DWORD **dwObjID**
- DWORD **dwHow**

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

- lib/glfw/deps/mingw/dinput.h

26.6 _DICOLORSET Struct Reference

Public Attributes

- DWORD **dwSize**
- D3DCOLOR **cTextFore**
- D3DCOLOR **cTextHighlight**
- D3DCOLOR **cCalloutLine**
- D3DCOLOR **cCalloutHighlight**
- D3DCOLOR **cBorder**
- D3DCOLOR **cControlFill**
- D3DCOLOR **cHighlightFill**
- D3DCOLOR **cAreaFill**

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

- lib/glfw/deps/mingw/dinput.h

26.7 _DICONFIGUREDEVICESPARAMSA Struct Reference

Public Attributes

- DWORD **dwSize**
- DWORD **dwcUsers**
- LPSTR **lptszUserNames**
- DWORD **dwcFormats**
- LPDIACTIONFORMATA **lprgFormats**
- HWND **hwnd**
- DICOLORSET **dics**
- LPUNKNOWN **lpUnkDDSTarget**

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

- lib/glfw/deps/mingw/dinput.h

26.8 _DICONFIGUREDEVICESPARAMSW Struct Reference

Public Attributes

- DWORD **dwSize**
- DWORD **dwcUsers**
- LPWSTR **lptszUserNames**
- DWORD **dwcFormats**
- LPDIACTIONFORMATW **lprgFormats**
- HWND **hwnd**
- DICOLORSET **dics**
- LPUNKNOWN **lpUnkDDSTarget**

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

- lib/glfw/deps/mingw/dinput.h

26.9 _DIDATAFORMAT Struct Reference

Public Attributes

- DWORD **dwSize**
- DWORD **dwObjSize**
- DWORD **dwFlags**
- DWORD **dwDataSize**
- DWORD **dwNumObjs**
- LPDIOBJECTDATAFORMAT **rgodf**

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

- lib/glfw/deps/mingw/dinput.h

26.10 _DIDEVICEIMAGEINFOA Struct Reference

Public Attributes

- CHAR **tszImagePath** [MAX_PATH]
- DWORD **dwFlags**
- DWORD **dwViewID**
- RECT **rcOverlay**
- DWORD **dwObjID**
- DWORD **dwcValidPts**
- POINT **rgptCalloutLine** [5]
- RECT **rcCalloutRect**
- DWORD **dwTextAlign**

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

- lib/glfw/deps/mingw/dinput.h

26.11 _DIDEVICEIMAGEINFOHEADERA Struct Reference

Public Attributes

- DWORD **dwSize**
- DWORD **dwSizeImageInfo**
- DWORD **dwcViews**
- DWORD **dwcButtons**
- DWORD **dwcAxes**
- DWORD **dwcPOVs**
- DWORD **dwBufferSize**
- DWORD **dwBufferUsed**
- LPDIDEVICEIMAGEINFOA **lprgImageInfoArray**

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

- lib/glfw/deps/mingw/dinput.h

26.12 _DIDEVICEIMAGEINFOHEADERW Struct Reference

Public Attributes

- DWORD **dwSize**
- DWORD **dwSizeImageInfo**
- DWORD **dwcViews**
- DWORD **dwcButtons**
- DWORD **dwcAxes**
- DWORD **dwcPOVs**
- DWORD **dwBufferSize**
- DWORD **dwBufferUsed**
- LPDIDEVICEIMAGEINFOW **lprgImageInfoArray**

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

- lib/glfw/deps/mingw/dinput.h

26.13 _DIDEVICEIMAGEINFOW Struct Reference

Public Attributes

- WCHAR **tszImagePath** [MAX_PATH]
- DWORD **dwFlags**
- DWORD **dwViewID**
- RECT **rcOverlay**
- DWORD **dwObjID**
- DWORD **dwcValidPts**
- POINT **rgptCalloutLine** [5]
- RECT **rcCalloutRect**
- DWORD **dwTextAlign**

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

- lib/glfw/deps/mingw/dinput.h

26.14 _DIOBJECTDATAFORMAT Struct Reference

Public Attributes

- const GUID * **pguid**
- DWORD **dwOfs**
- DWORD **dwType**
- DWORD **dwFlags**

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

- lib/glfw/deps/mingw/dinput.h

26.15 _GLFWcontext Struct Reference

Public Attributes

- int **client**
- int **source**
- int **major**
- int **minor**
- int **revision**
- GLFWbool **forward**
- GLFWbool **debug**
- GLFWbool **noerror**
- int **profile**
- int **robustness**
- int **release**
- PFNGLGETSTRINGIPROC **GetStringi**
- PFNGLGETINTEGERVPROC **GetIntegerv**
- PFNGLGETSTRINGPROC **GetString**
- void(* **makeCurrent**)(_GLFWwindow *)
- void(* **swapBuffers**)(_GLFWwindow *)
- void(* **swapInterval**)(int)
- int(* **extensionSupported**)(const char *)
- GLFWglproc(* **getProcAddress**)(const char *)
- void(* **destroy**)(_GLFWwindow *)
- struct {
  EGLConfig **config**
  EGLContext **handle**
  EGLSurface **surface**
  void * **client**
  } **egl**
- struct {
  OSMesaContext **handle**
  int **width**
  int **height**
  void * **buffer**
  } **osmesa**

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

- lib/glfw/src/internal.h

26.16 _GLFWcontextGLX Struct Reference

Public Attributes

- GLXContext **handle**
- GLXWindow **window**

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

- lib/glfw/src/x11_platform.h

26.17 _GLFWcontextNSGL Struct Reference

Public Attributes

- id **pixelFormat**
- id **object**

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

- lib/glfw/src/cocoa_platform.h

26.18 _GLFWcontextWGL Struct Reference

Public Attributes

- HDC **dc**
- HGLRC **handle**
- int **interval**

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

- lib/glfw/src/win32_platform.h

26.19 _GLFWctxconfig Struct Reference

Public Attributes

- int **client**
- int **source**
- int **major**
- int **minor**
- GLFWbool **forward**
- GLFWbool **debug**
- GLFWbool **noerror**
- int **profile**
- int **robustness**
- int **release**
- _GLFWwindow * **share**
-

  struct {
    GLFWbool **offline**
  } **nsgl**

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

- lib/glfw/src/internal.h

26.20 _GLFWcursor Struct Reference

Public Attributes

- _GLFWcursor * **next**

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

- lib/glfw/src/internal.h

26.21 _GLFWcursorNS Struct Reference

Public Attributes

- id **object**

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

- lib/glfw/src/cocoa_platform.h

26.22 _GLFWcursorWayland Struct Reference

Public Attributes

- struct wl_cursor * **cursor**
- struct wl_cursor * **cursorHiDPI**
- struct wl_buffer * **buffer**
- int **width**
- int **height**
- int **xhot**
- int **yhot**
- int **currentImage**

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

- lib/glfw/src/wl_platform.h

26.23 _GLFWcursorWin32 Struct Reference

Public Attributes

- HCURSOR **handle**

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

- lib/glfw/src/win32_platform.h

26.24 _GLFWcursorX11 Struct Reference

Public Attributes

- Cursor **handle**

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

- lib/glfw/src/x11_platform.h

26.25 _GLFWdecorationWayland Struct Reference

Public Attributes

- struct wl_surface * **surface**
- struct wl_subsurface * **subsurface**
- struct wp_viewport * **viewport**

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

- lib/glfw/src/wl_platform.h

26.26 _GLFWerror Struct Reference

Public Attributes

- _GLFWerror * **next**
- int **code**
- char **description** [_GLFW_MESSAGE_SIZE]

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

- lib/glfw/src/internal.h

26.27 _GLFWfbconfig Struct Reference

Public Attributes

- int **redBits**
- int **greenBits**
- int **blueBits**
- int **alphaBits**
- int **depthBits**
- int **stencilBits**
- int **accumRedBits**
- int **accumGreenBits**
- int **accumBlueBits**
- int **accumAlphaBits**
- int **auxBuffers**
- GLFWbool **stereo**
- int **samples**
- GLFWbool **sRGB**
- GLFWbool **doublebuffer**
- GLFWbool **transparent**
- uintptr_t **handle**

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

- lib/glfw/src/internal.h

26.28 _GLFWinitconfig Struct Reference

Public Attributes

- GLFWbool **hatButtons**
- int **angleType**
- int **platformID**
- PFN_vkGetInstanceProcAddr **vulkanLoader**
- struct {
  GLFWbool **menubar**
  GLFWbool **chdir**
  } **ns**
- struct {
  GLFWbool **xcbVulkanSurface**
  } **x11**

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

- lib/glfw/src/internal.h

26.29 _GLFWjoyelementNS Struct Reference

Protected Attributes

- IOHIDElementRef **native**
- uint32_t **usage**
- int **index**
- long **minimum**
- long **maximum**

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

- lib/glfw/src/cocoa_joystick.m

26.30 _GLFWjoyobjectWin32 Struct Reference

Public Attributes

- int **offset**
- int **type**

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

- lib/glfw/src/win32_joystick.h

26.31 _GLFWjoystick Struct Reference

Public Attributes

- GLFWbool **present**
- float * **axes**
- int **axisCount**
- unsigned char * **buttons**
- int **buttonCount**
- unsigned char * **hats**
- int **hatCount**
- char **name** [128]
- void * **userPointer**
- char **guid** [33]
- _GLFWmapping * **mapping**

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

- lib/glfw/src/internal.h

26.32 _GLFWjoystickLinux Struct Reference

Public Attributes

- int **fd**
- char **path** [PATH_MAX]
- int **keyMap** [KEY_CNT - BTN_MISC]
- int **absMap** [ABS_CNT]
- struct input_absinfo **absInfo** [ABS_CNT]
- int **hats** [4][2]

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

- lib/glfw/src/linux_joystick.h

26.33 _GLFWjoystickNS Struct Reference

Public Attributes

- IOHIDDeviceRef **device**
- CFMutableArrayRef **axes**
- CFMutableArrayRef **buttons**
- CFMutableArrayRef **hats**

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

- lib/glfw/src/cocoa_joystick.h

26.34 _GLFWjoystickWin32 Struct Reference

Public Attributes

- _GLFWjoyobjectWin32 * **objects**
- int **objectCount**
- IDirectInputDevice8W * **device**
- DWORD **index**
- GUID **guid**

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

- lib/glfw/src/win32_joystick.h

26.35 _GLFWlibrary Struct Reference

Public Attributes

- GLFWbool **initialized**
- GLFWallocator **allocator**
- _GLFWplatform **platform**
- struct {
  _GLFWinitconfig **init**
  _GLFWfbconfig **framebuffer**
  _GLFWwndconfig **window**
  _GLFWctxconfig **context**
  int **refreshRate**
  } **hints**
- _GLFWerror * **errorListHead**
- _GLFWcursor * **cursorListHead**
- _GLFWwindow * **windowListHead**
- _GLFWmonitor ** **monitors**
- int **monitorCount**
- GLFWbool **joysticksInitialized**
- _GLFWjoystick **joysticks** [GLFW_JOYSTICK_LAST+1]
- _GLFWmapping * **mappings**
- int **mappingCount**
- _GLFWtls **errorSlot**
- _GLFWtls **contextSlot**
- _GLFWmutex **errorLock**
- struct {
  uint64_t **offset**
  } **timer**
-

struct {
EGLenum **platform**
EGLDisplay **display**
EGLint **major**
EGLint **minor**
GLFWbool **prefix**
GLFWbool **KHR_create_context**
GLFWbool **KHR_create_context_no_error**
GLFWbool **KHR_gl_colorspace**
GLFWbool **KHR_get_all_proc_addresses**
GLFWbool **KHR_context_flush_control**
GLFWbool **EXT_client_extensions**
GLFWbool **EXT_platform_base**
GLFWbool **EXT_platform_x11**
GLFWbool **EXT_platform_wayland**
GLFWbool **EXT_present_opaque**
GLFWbool **ANGLE_platform_angle**
GLFWbool **ANGLE_platform_angle_opengl**
GLFWbool **ANGLE_platform_angle_d3d**
GLFWbool **ANGLE_platform_angle_vulkan**
GLFWbool **ANGLE_platform_angle_metal**
void * **handle**
PFN_eglGetConfigAttrib **GetConfigAttrib**
PFN_eglGetConfigs **GetConfigs**
PFN_eglGetDisplay **GetDisplay**
PFN_eglGetError **GetError**
PFN_eglInitialize **Initialize**
PFN_eglTerminate **Terminate**
PFN_eglBindAPI **BindAPI**
PFN_eglCreateContext **CreateContext**
PFN_eglDestroySurface **DestroySurface**
PFN_eglDestroyContext **DestroyContext**
PFN_eglCreateWindowSurface **CreateWindowSurface**
PFN_eglMakeCurrent **MakeCurrent**
PFN_eglSwapBuffers **SwapBuffers**
PFN_eglSwapInterval **SwapInterval**
PFN_eglQueryString **QueryString**
PFN_eglGetProcAddress **GetProcAddress**
PFNEGLGETPLATFORMDISPLAYEXTPROC **GetPlatformDisplayEXT**
PFNEGLCREATEPLATFORMWINDOWSURFACEEXTPROC **CreatePlatformWindowSurfaceEXT**
} **egl**

-

struct {
void * **handle**
PFN_OSMesaCreateContextExt **CreateContextExt**
PFN_OSMesaCreateContextAttribs **CreateContextAttribs**
PFN_OSMesaDestroyContext **DestroyContext**
PFN_OSMesaMakeCurrent **MakeCurrent**
PFN_OSMesaGetColorBuffer **GetColorBuffer**
PFN_OSMesaGetDepthBuffer **GetDepthBuffer**
PFN_OSMesaGetProcAddress **GetProcAddress**
} **osmesa**

-

struct {

GLFWbool **available**
void * **handle**
char * **extensions** [2]
PFN_vkGetInstanceProcAddr **GetInstanceProcAddr**
GLFWbool **KHR_surface**
GLFWbool **KHR_win32_surface**
GLFWbool **MVK_macos_surface**
GLFWbool **EXT_metal_surface**
GLFWbool **KHR_xlib_surface**
GLFWbool **KHR_xcb_surface**
GLFWbool **KHR_wayland_surface**
} **vk**

- struct {
GLFWmonitorfun **monitor**
GLFWjoystickfun **joystick**
} **callbacks**

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

- lib/glfw/src/internal.h

26.36 _GLFWlibraryGLX Struct Reference

Public Attributes

- int **major**
- int **minor**
- int **eventBase**
- int **errorBase**
- void * **handle**
- PFNGLXGETFBCONFIGSPROC **GetFBConfigs**
- PFNGLXGETFBCONFIGATTRIBPROC **GetFBConfigAttrib**
- PFNGLXGETCLIENTSTRINGPROC **GetClientString**
- PFNGLXQUERYEXTENSIONPROC **QueryExtension**
- PFNGLXQUERYVERSIONPROC **QueryVersion**
- PFNGLXDESTROYCONTEXTPROC **DestroyContext**
- PFNGLXMAKECURRENTPROC **MakeCurrent**
- PFNGLXSWAPBUFFERSPROC **SwapBuffers**
- PFNGLXQUERYEXTENSIONSSTRINGPROC **QueryExtensionsString**
- PFNGLXCREATENEWCONTEXTPROC **CreateNewContext**
- PFNGLXGETVISUALFROMFBCONFIGPROC **GetVisualFromFBConfig**
- PFNGLXCREATEWINDOWPROC **CreateWindow**
- PFNGLXDESTROYWINDOWPROC **DestroyWindow**
- PFNGLXGETPROCADDRESSPROC **GetProcAddress**
- PFNGLXGETPROCADDRESSPROC **GetProcAddressARB**
- PFNGLXSWAPINTERVALSGIPROC **SwapIntervalSGI**
- PFNGLXSWAPINTERVALEXTPROC **SwapIntervalEXT**
- PFNGLXSWAPINTERVALMESAPROC **SwapIntervalMESA**

- PFNGLXCREATECONTEXTATTRIBSARBPROC **CreateContextAttribsARB**
- GLFWbool **SGI_swap_control**
- GLFWbool **EXT_swap_control**
- GLFWbool **MESA_swap_control**
- GLFWbool **ARB_multisample**
- GLFWbool **ARB_framebuffer_sRGB**
- GLFWbool **EXT_framebuffer_sRGB**
- GLFWbool **ARB_create_context**
- GLFWbool **ARB_create_context_profile**
- GLFWbool **ARB_create_context_robustness**
- GLFWbool **EXT_create_context_es2_profile**
- GLFWbool **ARB_create_context_no_error**
- GLFWbool **ARB_context_flush_control**

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

- lib/glfw/src/x11_platform.h

26.37 _GLFWlibraryLinux Struct Reference

Public Attributes

- int **inotify**
- int **watch**
- regex_t **regex**
- GLFWbool **dropped**

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

- lib/glfw/src/linux_joystick.h

26.38 _GLFWlibraryNS Struct Reference

Public Attributes

- CGEventSourceRef **eventSource**
- id **delegate**
- GLFWbool **cursorHidden**
- TISInputSourceRef **inputSource**
- IOHIDManagerRef **hidManager**
- id **unicodeData**
- id **helper**
- id **keyUpMonitor**
- id **nibObjects**
- char **keynames** [GLFW_KEY_LAST+1][17]
- short int **keycodes** [256]
- short int **scancodes** [GLFW_KEY_LAST+1]
- char * **clipboardString**

- CGPoint **cascadePoint**
- double **restoreCursorPosX**
- double **restoreCursorPosY**
- _GLFWwindow * **disabledCursorWindow**
-

  struct {
  CFBundleRef **bundle**
  PFN_TISCopyCurrentKeyboardLayoutInputSource **CopyCurrentKeyboardLayoutInputSource**
  PFN_TISGetInputSourceProperty **GetInputSourceProperty**
  PFN_LMGetKbdType **GetKbdType**
  CFStringRef **kPropertyUnicodeKeyLayoutData**
  } **tis**

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

- lib/glfw/src/cocoa_platform.h

26.39 _GLFWlibraryNSGL Struct Reference

Public Attributes

- CFBundleRef **framework**

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

- lib/glfw/src/cocoa_platform.h

26.40 _GLFWlibraryNull Struct Reference

Public Attributes

- int **xcursor**
- int **ycursor**
- char * **clipboardString**
- _GLFWwindow * **focusedWindow**

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

- lib/glfw/src/null_platform.h

26.41 _GLFWlibraryWayland Struct Reference

Public Attributes

- struct wl_display * **display**
- struct wl_registry * **registry**
- struct wl_compositor * **compositor**
- struct wl_subcompositor * **subcompositor**
- struct wl_shm * **shm**
- struct wl_seat * **seat**
- struct wl_pointer * **pointer**
- struct wl_keyboard * **keyboard**
- struct wl_data_device_manager * **dataDeviceManager**
- struct wl_data_device * **dataDevice**
- struct wl_data_offer * **dataOffer**
- struct wl_data_source * **dataSource**
- struct xdg_wm_base * **wmBase**
- struct zxdg_decoration_manager_v1 * **decorationManager**
- struct wp_viewporter * **viewporter**
- struct zwp_relative_pointer_manager_v1 * **relativePointerManager**
- struct zwp_pointer_constraints_v1 * **pointerConstraints**
- struct zwp_idle_inhibit_manager_v1 * **idleInhibitManager**
- int **compositorVersion**
- int **seatVersion**
- struct wl_cursor_theme * **cursorTheme**
- struct wl_cursor_theme * **cursorThemeHiDPI**
- struct wl_surface * **cursorSurface**
- const char * **cursorPreviousName**
- int **cursorTimerfd**
- uint32_t **serial**
- uint32_t **pointerEnterSerial**
- int32_t **keyboardRepeatRate**
- int32_t **keyboardRepeatDelay**
- int **keyboardLastKey**
- int **keyboardLastScancode**
- char * **clipboardString**
- size_t **clipboardSize**
- char * **clipboardSendString**
- size_t **clipboardSendSize**
- int **timerfd**
- short int **keycodes** [256]
- short int **scancodes** [GLFW_KEY_LAST+1]
- char **keynames** [GLFW_KEY_LAST+1][5]
-

  struct {
    void * **handle**
    struct xkb_context * **context**
    struct xkb_keymap * **keymap**
    struct xkb_state * **state**
    struct xkb_compose_state * **composeState**
    xkb_mod_mask_t **controlMask**
    xkb_mod_mask_t **altMask**
    xkb_mod_mask_t **shiftMask**
    xkb_mod_mask_t **superMask**

xkb_mod_mask_t **capsLockMask**
xkb_mod_mask_t **numLockMask**
unsigned int **modifiers**
PFN_xkb_context_new **context_new**
PFN_xkb_context_unref **context_unref**
PFN_xkb_keymap_new_from_string **keymap_new_from_string**
PFN_xkb_keymap_unref **keymap_unref**
PFN_xkb_keymap_mod_get_index **keymap_mod_get_index**
PFN_xkb_keymap_key_repeats **keymap_key_repeats**
PFN_xkb_keymap_key_get_syms_by_level **keymap_key_get_syms_by_level**
PFN_xkb_state_new **state_new**
PFN_xkb_state_unref **state_unref**
PFN_xkb_state_key_get_syms **state_key_get_syms**
PFN_xkb_state_update_mask **state_update_mask**
PFN_xkb_state_serialize_mods **state_serialize_mods**
PFN_xkb_state_key_get_layout **state_key_get_layout**
PFN_xkb_compose_table_new_from_locale **compose_table_new_from_locale**
PFN_xkb_compose_table_unref **compose_table_unref**
PFN_xkb_compose_state_new **compose_state_new**
PFN_xkb_compose_state_unref **compose_state_unref**
PFN_xkb_compose_state_feed **compose_state_feed**
PFN_xkb_compose_state_get_status **compose_state_get_status**
PFN_xkb_compose_state_get_one_sym **compose_state_get_one_sym**
} **xkb**

- _GLFWwindow * **pointerFocus**
- _GLFWwindow * **keyboardFocus**
-

struct {
void * **handle**
PFN_wl_display_flush **display_flush**
PFN_wl_display_cancel_read **display_cancel_read**
PFN_wl_display_dispatch_pending **display_dispatch_pending**
PFN_wl_display_read_events **display_read_events**
PFN_wl_display_disconnect **display_disconnect**
PFN_wl_display_roundtrip **display_roundtrip**
PFN_wl_display_get_fd **display_get_fd**
PFN_wl_display_prepare_read **display_prepare_read**
PFN_wl_proxy_marshal **proxy_marshal**
PFN_wl_proxy_add_listener **proxy_add_listener**
PFN_wl_proxy_destroy **proxy_destroy**
PFN_wl_proxy_marshal_constructor **proxy_marshal_constructor**
PFN_wl_proxy_marshal_constructor_versioned **proxy_marshal_constructor_versioned**
PFN_wl_proxy_get_user_data **proxy_get_user_data**
PFN_wl_proxy_set_user_data **proxy_set_user_data**
PFN_wl_proxy_get_version **proxy_get_version**
PFN_wl_proxy_marshal_flags **proxy_marshal_flags**
} **client**

-

struct {
void * **handle**
PFN_wl_cursor_theme_load **theme_load**
PFN_wl_cursor_theme_destroy **theme_destroy**
PFN_wl_cursor_theme_get_cursor **theme_get_cursor**

PFN_wl_cursor_image_get_buffer **image_get_buffer**
} **cursor**

•

struct {
void * **handle**
PFN_wl_egl_window_create **window_create**
PFN_wl_egl_window_destroy **window_destroy**
PFN_wl_egl_window_resize **window_resize**
} **egl**

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

- lib/glfw/src/wl_platform.h

26.42 _GLFWlibraryWGL Struct Reference

Public Attributes

- HINSTANCE **instance**
- PFN_wglCreateContext **CreateContext**
- PFN_wglDeleteContext **DeleteContext**
- PFN_wglGetProcAddress **GetProcAddress**
- PFN_wglGetCurrentDC **GetCurrentDC**
- PFN_wglGetCurrentContext **GetCurrentContext**
- PFN_wglMakeCurrent **MakeCurrent**
- PFN_wglShareLists **ShareLists**
- PFNWGLSWAPINTERVALEXTPROC **SwapIntervalEXT**
- PFNWGLGETPIXELFORMATATTRIBIVARBPROC **GetPixelFormatAttribivARB**
- PFNWGLGETEXTENSIONSSTRINGEXTPROC **GetExtensionsStringEXT**
- PFNWGLGETEXTENSIONSSTRINGARBPROC **GetExtensionsStringARB**
- PFNWGLCREATECONTEXTATTRIBSARBPROC **CreateContextAttribsARB**
- GLFWbool **EXT_swap_control**
- GLFWbool **EXT_colorspace**
- GLFWbool **ARB_multisample**
- GLFWbool **ARB_framebuffer_sRGB**
- GLFWbool **EXT_framebuffer_sRGB**
- GLFWbool **ARB_pixel_format**
- GLFWbool **ARB_create_context**
- GLFWbool **ARB_create_context_profile**
- GLFWbool **EXT_create_context_es2_profile**
- GLFWbool **ARB_create_context_robustness**
- GLFWbool **ARB_create_context_no_error**
- GLFWbool **ARB_context_flush_control**

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

- lib/glfw/src/win32_platform.h

26.43 _GLFWlibraryWin32 Struct Reference

Public Attributes

- HINSTANCE **instance**
- HWND **helperWindowHandle**
- HDEVNOTIFY **deviceNotificationHandle**
- int **acquiredMonitorCount**
- char * **clipboardString**
- short int **keycodes** [512]
- short int **scancodes** [GLFW_KEY_LAST+1]
- char **keynames** [GLFW_KEY_LAST+1][5]
- double **restoreCursorPosX**
- double **restoreCursorPosY**
- _GLFWwindow * **disabledCursorWindow**
- RAWINPUT * **rawInput**
- int **rawInputSize**
- UINT **mouseTrailSize**
- struct {
  HINSTANCE **instance**
  PFN_DirectInput8Create **Create**
  IDirectInput8W * **api**
  } **dinput8**
- struct {
  HINSTANCE **instance**
  PFN_XInputGetCapabilities **GetCapabilities**
  PFN_XInputGetState **GetState**
  } **xinput**
- struct {
  HINSTANCE **instance**
  PFN_SetProcessDPIAware **SetProcessDPIAware_**
  PFN_ChangeWindowMessageFilterEx **ChangeWindowMessageFilterEx_**
  PFN_EnableNonClientDpiScaling **EnableNonClientDpiScaling_**
  PFN_SetProcessDpiAwarenessContext **SetProcessDpiAwarenessContext_**
  PFN_GetDpiForWindow **GetDpiForWindow_**
  PFN_AdjustWindowRectExForDpi **AdjustWindowRectExForDpi_**
  PFN_GetSystemMetricsForDpi **GetSystemMetricsForDpi_**
  } **user32**
- struct {
  HINSTANCE **instance**
  PFN_DwmIsCompositionEnabled **IsCompositionEnabled**
  PFN_DwmFlush **Flush**
  PFN_DwmEnableBlurBehindWindow **EnableBlurBehindWindow**
  PFN_DwmGetColorizationColor **GetColorizationColor**
  } **dwmapi**

- struct {
  HINSTANCE **instance**
  PFN_SetProcessDpiAwareness **SetProcessDpiAwareness_**
  PFN_GetDpiForMonitor **GetDpiForMonitor_**
  } **shcore**

- struct {
  HINSTANCE **instance**
  PFN_RtlVerifyVersionInfo **RtlVerifyVersionInfo_**
  } **ntdll**

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

- lib/glfw/src/win32_platform.h

26.44 _GLFWlibraryX11 Struct Reference

Public Attributes

- Display ∗ **display**
- int **screen**
- Window **root**
- float **contentScaleX**
- float **contentScaleY**
- Window **helperWindowHandle**
- Cursor **hiddenCursorHandle**
- XContext **context**
- XIM **im**
- int **errorCode**
- char ∗ **primarySelectionString**
- char ∗ **clipboardString**
- char **keynames** [GLFW_KEY_LAST+1][5]
- short int **keycodes** [256]
- short int **scancodes** [GLFW_KEY_LAST+1]
- double **restoreCursorPosX**
- double **restoreCursorPosY**
- _GLFWwindow ∗ **disabledCursorWindow**
- int **emptyEventPipe** [2]
- Atom **NET_SUPPORTED**
- Atom **NET_SUPPORTING_WM_CHECK**
- Atom **WM_PROTOCOLS**
- Atom **WM_STATE**
- Atom **WM_DELETE_WINDOW**
- Atom **NET_WM_NAME**
- Atom **NET_WM_ICON_NAME**
- Atom **NET_WM_ICON**
- Atom **NET_WM_PID**

- Atom **NET_WM_PING**
- Atom **NET_WM_WINDOW_TYPE**
- Atom **NET_WM_WINDOW_TYPE_NORMAL**
- Atom **NET_WM_STATE**
- Atom **NET_WM_STATE_ABOVE**
- Atom **NET_WM_STATE_FULLSCREEN**
- Atom **NET_WM_STATE_MAXIMIZED_VERT**
- Atom **NET_WM_STATE_MAXIMIZED_HORZ**
- Atom **NET_WM_STATE_DEMANDS_ATTENTION**
- Atom **NET_WM_BYPASS_COMPOSITOR**
- Atom **NET_WM_FULLSCREEN_MONITORS**
- Atom **NET_WM_WINDOW_OPACITY**
- Atom **NET_WM_CM_Sx**
- Atom **NET_WORKAREA**
- Atom **NET_CURRENT_DESKTOP**
- Atom **NET_ACTIVE_WINDOW**
- Atom **NET_FRAME_EXTENTS**
- Atom **NET_REQUEST_FRAME_EXTENTS**
- Atom **MOTIF_WM_HINTS**
- Atom **XdndAware**
- Atom **XdndEnter**
- Atom **XdndPosition**
- Atom **XdndStatus**
- Atom **XdndActionCopy**
- Atom **XdndDrop**
- Atom **XdndFinished**
- Atom **XdndSelection**
- Atom **XdndTypeList**
- Atom **text_uri_list**
- Atom **TARGETS**
- Atom **MULTIPLE**
- Atom **INCR**
- Atom **CLIPBOARD**
- Atom **PRIMARY**
- Atom **CLIPBOARD_MANAGER**
- Atom **SAVE_TARGETS**
- Atom **NULL_**
- Atom **UTF8_STRING**
- Atom **COMPOUND_STRING**
- Atom **ATOM_PAIR**
- Atom **GLFW_SELECTION**
-

struct {
  void ∗ **handle**
  GLFWbool **utf8**
  PFN_XAllocClassHint **AllocClassHint**
  PFN_XAllocSizeHints **AllocSizeHints**
  PFN_XAllocWMHints **AllocWMHints**
  PFN_XChangeProperty **ChangeProperty**
  PFN_XChangeWindowAttributes **ChangeWindowAttributes**
  PFN_XCheckIfEvent **CheckIfEvent**
  PFN_XCheckTypedWindowEvent **CheckTypedWindowEvent**
  PFN_XCloseDisplay **CloseDisplay**
  PFN_XCloseIM **CloseIM**

PFN_XConvertSelection **ConvertSelection**
PFN_XCreateColormap **CreateColormap**
PFN_XCreateFontCursor **CreateFontCursor**
PFN_XCreateIC **CreateIC**
PFN_XCreateRegion **CreateRegion**
PFN_XCreateWindow **CreateWindow**
PFN_XDefineCursor **DefineCursor**
PFN_XDeleteContext **DeleteContext**
PFN_XDeleteProperty **DeleteProperty**
PFN_XDestroyIC **DestroyIC**
PFN_XDestroyRegion **DestroyRegion**
PFN_XDestroyWindow **DestroyWindow**
PFN_XDisplayKeycodes **DisplayKeycodes**
PFN_XEventsQueued **EventsQueued**
PFN_XFilterEvent **FilterEvent**
PFN_XFindContext **FindContext**
PFN_XFlush **Flush**
PFN_XFree **Free**
PFN_XFreeColormap **FreeColormap**
PFN_XFreeCursor **FreeCursor**
PFN_XFreeEventData **FreeEventData**
PFN_XGetErrorText **GetErrorText**
PFN_XGetEventData **GetEventData**
PFN_XGetICValues **GetICValues**
PFN_XGetIMValues **GetIMValues**
PFN_XGetInputFocus **GetInputFocus**
PFN_XGetKeyboardMapping **GetKeyboardMapping**
PFN_XGetScreenSaver **GetScreenSaver**
PFN_XGetSelectionOwner **GetSelectionOwner**
PFN_XGetVisualInfo **GetVisualInfo**
PFN_XGetWMNormalHints **GetWMNormalHints**
PFN_XGetWindowAttributes **GetWindowAttributes**
PFN_XGetWindowProperty **GetWindowProperty**
PFN_XGrabPointer **GrabPointer**
PFN_XIconifyWindow **IconifyWindow**
PFN_XInternAtom **InternAtom**
PFN_XLookupString **LookupString**
PFN_XMapRaised **MapRaised**
PFN_XMapWindow **MapWindow**
PFN_XMoveResizeWindow **MoveResizeWindow**
PFN_XMoveWindow **MoveWindow**
PFN_XNextEvent **NextEvent**
PFN_XOpenIM **OpenIM**
PFN_XPeekEvent **PeekEvent**
PFN_XPending **Pending**
PFN_XQueryExtension **QueryExtension**
PFN_XQueryPointer **QueryPointer**
PFN_XRaiseWindow **RaiseWindow**
PFN_XRegisterIMInstantiateCallback **RegisterIMInstantiateCallback**
PFN_XResizeWindow **ResizeWindow**
PFN_XResourceManagerString **ResourceManagerString**
PFN_XSaveContext **SaveContext**
PFN_XSelectInput **SelectInput**
PFN_XSendEvent **SendEvent**
PFN_XSetClassHint **SetClassHint**
PFN_XSetErrorHandler **SetErrorHandler**
PFN_XSetICFocus **SetICFocus**
PFN_XSetIMValues **SetIMValues**

PFN_XSetInputFocus **SetInputFocus**
PFN_XSetLocaleModifiers **SetLocaleModifiers**
PFN_XSetScreenSaver **SetScreenSaver**
PFN_XSetSelectionOwner **SetSelectionOwner**
PFN_XSetWMHints **SetWMHints**
PFN_XSetWMNormalHints **SetWMNormalHints**
PFN_XSetWMProtocols **SetWMProtocols**
PFN_XSupportsLocale **SupportsLocale**
PFN_XSync **Sync**
PFN_XTranslateCoordinates **TranslateCoordinates**
PFN_XUndefineCursor **UndefineCursor**
PFN_XUngrabPointer **UngrabPointer**
PFN_XUnmapWindow **UnmapWindow**
PFN_XUnsetICFocus **UnsetICFocus**
PFN_XVisualIDFromVisual **VisualIDFromVisual**
PFN_XWarpPointer **WarpPointer**
PFN_XUnregisterIMInstantiateCallback **UnregisterIMInstantiateCallback**
PFN_Xutf8LookupString **utf8LookupString**
PFN_Xutf8SetWMProperties **utf8SetWMProperties**
} **xlib**

-

struct {
PFN_XrmDestroyDatabase **DestroyDatabase**
PFN_XrmGetResource **GetResource**
PFN_XrmGetStringDatabase **GetStringDatabase**
PFN_XrmUniqueQuark **UniqueQuark**
} **xrm**

-

struct {
GLFWbool **available**
void * **handle**
int **eventBase**
int **errorBase**
int **major**
int **minor**
GLFWbool **gammaBroken**
GLFWbool **monitorBroken**
PFN_XRRAllocGamma **AllocGamma**
PFN_XRRFreeCrtcInfo **FreeCrtcInfo**
PFN_XRRFreeGamma **FreeGamma**
PFN_XRRFreeOutputInfo **FreeOutputInfo**
PFN_XRRFreeScreenResources **FreeScreenResources**
PFN_XRRGetCrtcGamma **GetCrtcGamma**
PFN_XRRGetCrtcGammaSize **GetCrtcGammaSize**
PFN_XRRGetCrtcInfo **GetCrtcInfo**
PFN_XRRGetOutputInfo **GetOutputInfo**
PFN_XRRGetOutputPrimary **GetOutputPrimary**
PFN_XRRGetScreenResourcesCurrent **GetScreenResourcesCurrent**
PFN_XRRQueryExtension **QueryExtension**
PFN_XRRQueryVersion **QueryVersion**
PFN_XRRSelectInput **SelectInput**
PFN_XRRSetCrtcConfig **SetCrtcConfig**
PFN_XRRSetCrtcGamma **SetCrtcGamma**

PFN_XRRUpdateConfiguration **UpdateConfiguration**
} **randr**

- struct {
  GLFWbool **available**
  GLFWbool **detectable**
  int **majorOpcode**
  int **eventBase**
  int **errorBase**
  int **major**
  int **minor**
  unsigned int **group**
  PFN_XkbFreeKeyboard **FreeKeyboard**
  PFN_XkbFreeNames **FreeNames**
  PFN_XkbGetMap **GetMap**
  PFN_XkbGetNames **GetNames**
  PFN_XkbGetState **GetState**
  PFN_XkbKeycodeToKeysym **KeycodeToKeysym**
  PFN_XkbQueryExtension **QueryExtension**
  PFN_XkbSelectEventDetails **SelectEventDetails**
  PFN_XkbSetDetectableAutoRepeat **SetDetectableAutoRepeat**
  } **xkb**

- struct {
  int **count**
  int **timeout**
  int **interval**
  int **blanking**
  int **exposure**
  } **saver**

- struct {
  int **version**
  Window **source**
  Atom **format**
  } **xdnd**

- struct {
  void * **handle**
  PFN_XcursorImageCreate **ImageCreate**
  PFN_XcursorImageDestroy **ImageDestroy**
  PFN_XcursorImageLoadCursor **ImageLoadCursor**
  PFN_XcursorGetTheme **GetTheme**
  PFN_XcursorGetDefaultSize **GetDefaultSize**
  PFN_XcursorLibraryLoadImage **LibraryLoadImage**
  } **xcursor**

-

struct {
GLFWbool **available**
void ∗ **handle**
int **major**
int **minor**
PFN_XineramaIsActive **IsActive**
PFN_XineramaQueryExtension **QueryExtension**
PFN_XineramaQueryScreens **QueryScreens**
} **xinerama**

- struct {
  GLFWbool **available**
  void ∗ **handle**
  PFN_XGetXCBConnection **GetXCBConnection**
  } **x11xcb**

- struct {
  GLFWbool **available**
  void ∗ **handle**
  int **eventBase**
  int **errorBase**
  PFN_XF86VidModeQueryExtension **QueryExtension**
  PFN_XF86VidModeGetGammaRamp **GetGammaRamp**
  PFN_XF86VidModeSetGammaRamp **SetGammaRamp**
  PFN_XF86VidModeGetGammaRampSize **GetGammaRampSize**
  } **vidmode**

- struct {
  GLFWbool **available**
  void ∗ **handle**
  int **majorOpcode**
  int **eventBase**
  int **errorBase**
  int **major**
  int **minor**
  PFN_XIQueryVersion **QueryVersion**
  PFN_XISelectEvents **SelectEvents**
  } **xi**

- struct {
  GLFWbool **available**
  void ∗ **handle**
  int **major**
  int **minor**
  int **eventBase**
  int **errorBase**
  PFN_XRenderQueryExtension **QueryExtension**
  PFN_XRenderQueryVersion **QueryVersion**
  PFN_XRenderFindVisualFormat **FindVisualFormat**
  } **xrender**

- struct {
  GLFWbool **available**
  void * **handle**
  int **major**
  int **minor**
  int **eventBase**
  int **errorBase**
  PFN_XShapeQueryExtension **QueryExtension**
  PFN_XShapeCombineRegion **ShapeCombineRegion**
  PFN_XShapeQueryVersion **QueryVersion**
  PFN_XShapeCombineMask **ShapeCombineMask**
  } **xshape**

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

- lib/glfw/src/x11_platform.h

26.45 _GLFWmapelement Struct Reference

Public Attributes

- uint8_t **type**
- uint8_t **index**
- int8_t **axisScale**
- int8_t **axisOffset**

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

- lib/glfw/src/internal.h

26.46 _GLFWmapping Struct Reference

Public Attributes

- char **name** [128]
- char **guid** [33]
- _GLFWmapelement **buttons** [15]
- _GLFWmapelement **axes** [6]

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

- lib/glfw/src/internal.h

26.47 _GLFWmonitor Struct Reference

Public Attributes

- char **name** [128]
- void * **userPointer**
- int **widthMM**
- int **heightMM**
- _GLFWwindow * **window**
- GLFWvidmode * **modes**
- int **modeCount**
- GLFWvidmode **currentMode**
- GLFWgammaramp **originalRamp**
- GLFWgammaramp **currentRamp**

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

- lib/glfw/src/internal.h

26.48 _GLFWmonitorNS Struct Reference

Public Attributes

- CGDirectDisplayID **displayID**
- CGDisplayModeRef **previousMode**
- uint32_t **unitNumber**
- id **screen**
- double **fallbackRefreshRate**

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

- lib/glfw/src/cocoa_platform.h

26.49 _GLFWmonitorNull Struct Reference

Public Attributes

- GLFWgammaramp **ramp**

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

- lib/glfw/src/null_platform.h

26.50 _GLFWmonitorWayland Struct Reference

Public Attributes

- struct wl_output ∗ **output**
- uint32_t **name**
- int **currentMode**
- int **x**
- int **y**
- int **scale**

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

- lib/glfw/src/wl_platform.h

26.51 _GLFWmonitorWin32 Struct Reference

Public Attributes

- HMONITOR **handle**
- WCHAR **adapterName** [32]
- WCHAR **displayName** [32]
- char **publicAdapterName** [32]
- char **publicDisplayName** [32]
- GLFWbool **modesPruned**
- GLFWbool **modeChanged**

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

- lib/glfw/src/win32_platform.h

26.52 _GLFWmonitorX11 Struct Reference

Public Attributes

- RROutput **output**
- RRCrtc **crtc**
- RRMode **oldMode**
- int **index**

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

- lib/glfw/src/x11_platform.h

26.53 _GLFWmutex Struct Reference

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

- lib/glfw/src/internal.h

26.54 _GLFWmutexPOSIX Struct Reference

Public Attributes

- GLFWbool **allocated**
- pthread_mutex_t **handle**

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

- lib/glfw/src/posix_thread.h

26.55 _GLFWmutexWin32 Struct Reference

Public Attributes

- GLFWbool **allocated**
- CRITICAL_SECTION **section**

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

- lib/glfw/src/win32_thread.h

26.56 _GLFWobjenumWin32 Struct Reference

Public Attributes

- IDirectInputDevice8W * **device**
- _GLFWjoyobjectWin32 * **objects**
- int **objectCount**
- int **axisCount**
- int **sliderCount**
- int **buttonCount**
- int **povCount**

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

- lib/glfw/src/win32_joystick.c

26.57 _GLFWplatform Struct Reference

Public Attributes

- int **platformID**
- GLFWbool(∗ **init**)(void)
- void(∗ **terminate**)(void)
- void(∗ **getCursorPos**)(_GLFWwindow ∗, double ∗, double ∗)
- void(∗ **setCursorPos**)(_GLFWwindow ∗, double, double)
- void(∗ **setCursorMode**)(_GLFWwindow ∗, int)
- void(∗ **setRawMouseMotion**)(_GLFWwindow ∗, GLFWbool)
- GLFWbool(∗ **rawMouseMotionSupported**)(void)
- int(∗ **createCursor**)(_GLFWcursor ∗, const GLFWimage ∗, int, int)
- int(∗ **createStandardCursor**)(_GLFWcursor ∗, int)
- void(∗ **destroyCursor**)(_GLFWcursor ∗)
- void(∗ **setCursor**)(_GLFWwindow ∗, _GLFWcursor ∗)
- const char ∗(∗ **getScancodeName**)(int)
- int(∗ **getKeyScancode**)(int)
- void(∗ **setClipboardString**)(const char ∗)
- const char ∗(∗ **getClipboardString**)(void)
- GLFWbool(∗ **initJoysticks**)(void)
- void(∗ **terminateJoysticks**)(void)
- int(∗ **pollJoystick**)(_GLFWjoystick ∗, int)
- const char ∗(∗ **getMappingName**)(void)
- void(∗ **updateGamepadGUID**)(char ∗)
- void(∗ **freeMonitor**)(_GLFWmonitor ∗)
- void(∗ **getMonitorPos**)(_GLFWmonitor ∗, int ∗, int ∗)
- void(∗ **getMonitorContentScale**)(_GLFWmonitor ∗, float ∗, float ∗)
- void(∗ **getMonitorWorkarea**)(_GLFWmonitor ∗, int ∗, int ∗, int ∗, int ∗)
- GLFWvidmode ∗(∗ **getVideoModes**)(_GLFWmonitor ∗, int ∗)
- void(∗ **getVideoMode**)(_GLFWmonitor ∗, GLFWvidmode ∗)
- GLFWbool(∗ **getGammaRamp**)(_GLFWmonitor ∗, GLFWgammaramp ∗)
- void(∗ **setGammaRamp**)(_GLFWmonitor ∗, const GLFWgammaramp ∗)
- int(∗ **createWindow**)(_GLFWwindow ∗, const _GLFWwndconfig ∗, const _GLFWctxconfig ∗, const _GLFWfbconfig ∗)
- void(∗ **destroyWindow**)(_GLFWwindow ∗)
- void(∗ **setWindowTitle**)(_GLFWwindow ∗, const char ∗)
- void(∗ **setWindowIcon**)(_GLFWwindow ∗, int, const GLFWimage ∗)
- void(∗ **getWindowPos**)(_GLFWwindow ∗, int ∗, int ∗)
- void(∗ **setWindowPos**)(_GLFWwindow ∗, int, int)
- void(∗ **getWindowSize**)(_GLFWwindow ∗, int ∗, int ∗)
- void(∗ **setWindowSize**)(_GLFWwindow ∗, int, int)
- void(∗ **setWindowSizeLimits**)(_GLFWwindow ∗, int, int, int, int)
- void(∗ **setWindowAspectRatio**)(_GLFWwindow ∗, int, int)
- void(∗ **getFramebufferSize**)(_GLFWwindow ∗, int ∗, int ∗)
- void(∗ **getWindowFrameSize**)(_GLFWwindow ∗, int ∗, int ∗, int ∗, int ∗)
- void(∗ **getWindowContentScale**)(_GLFWwindow ∗, float ∗, float ∗)
- void(∗ **iconifyWindow**)(_GLFWwindow ∗)
- void(∗ **restoreWindow**)(_GLFWwindow ∗)
- void(∗ **maximizeWindow**)(_GLFWwindow ∗)
- void(∗ **showWindow**)(_GLFWwindow ∗)
- void(∗ **hideWindow**)(_GLFWwindow ∗)
- void(∗ **requestWindowAttention**)(_GLFWwindow ∗)
- void(∗ **focusWindow**)(_GLFWwindow ∗)

- void(* **setWindowMonitor**)(_GLFWwindow *, _GLFWmonitor *, int, int, int, int, int)
- int(* **windowFocused**)(_GLFWwindow *)
- int(* **windowIconified**)(_GLFWwindow *)
- int(* **windowVisible**)(_GLFWwindow *)
- int(* **windowMaximized**)(_GLFWwindow *)
- int(* **windowHovered**)(_GLFWwindow *)
- int(* **framebufferTransparent**)(_GLFWwindow *)
- float(* **getWindowOpacity**)(_GLFWwindow *)
- void(* **setWindowResizable**)(_GLFWwindow *, GLFWbool)
- void(* **setWindowDecorated**)(_GLFWwindow *, GLFWbool)
- void(* **setWindowFloating**)(_GLFWwindow *, GLFWbool)
- void(* **setWindowOpacity**)(_GLFWwindow *, float)
- void(* **setWindowMousePassthrough**)(_GLFWwindow *, GLFWbool)
- void(* **pollEvents**)(void)
- void(* **waitEvents**)(void)
- void(* **waitEventsTimeout**)(double)
- void(* **postEmptyEvent**)(void)
- EGLenum(* **getEGLPlatform**)(EGLint **)
- EGLNativeDisplayType(* **getEGLNativeDisplay**)(void)
- EGLNativeWindowType(* **getEGLNativeWindow**)(_GLFWwindow *)
- void(* **getRequiredInstanceExtensions**)(char **)
- int(* **getPhysicalDevicePresentationSupport**)(VkInstance, VkPhysicalDevice, uint32_t)
- VkResult(* **createWindowSurface**)(VkInstance, _GLFWwindow *, const VkAllocationCallbacks *, VkSurfaceKHR *)

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

- lib/glfw/src/internal.h

26.58 _GLFWtimerNS Struct Reference

Public Attributes

- uint64_t **frequency**

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

- lib/glfw/src/cocoa_time.h

26.59 _GLFWtimerPOSIX Struct Reference

Public Attributes

- clockid_t **clock**
- uint64_t **frequency**

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

- lib/glfw/src/posix_time.h

26.60 _GLFWtimerWin32 Struct Reference

Public Attributes

- uint64_t **frequency**

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

- lib/glfw/src/win32_time.h

26.61 _GLFWtls Struct Reference

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

- lib/glfw/src/internal.h

26.62 _GLFWtlsPOSIX Struct Reference

Public Attributes

- GLFWbool **allocated**
- pthread_key_t **key**

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

- lib/glfw/src/posix_thread.h

26.63 _GLFWtlsWin32 Struct Reference

Public Attributes

- GLFWbool **allocated**
- DWORD **index**

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

- lib/glfw/src/win32_thread.h

26.64 _GLFWwindow Struct Reference

Public Attributes

- struct _GLFWwindow * **next**
- GLFWbool **resizable**
- GLFWbool **decorated**
- GLFWbool **autoIconify**
- GLFWbool **floating**
- GLFWbool **focusOnShow**
- GLFWbool **mousePassthrough**
- GLFWbool **shouldClose**
- void * **userPointer**
- GLFWbool **doublebuffer**
- GLFWvidmode **videoMode**
- _GLFWmonitor * **monitor**
- _GLFWcursor * **cursor**
- int **minwidth**
- int **minheight**
- int **maxwidth**
- int **maxheight**
- int **numer**
- int **denom**
- GLFWbool **stickyKeys**
- GLFWbool **stickyMouseButtons**
- GLFWbool **lockKeyMods**
- int **cursorMode**
- char **mouseButtons** [GLFW_MOUSE_BUTTON_LAST+1]
- char **keys** [GLFW_KEY_LAST+1]
- double **virtualCursorPosX**
- double **virtualCursorPosY**
- GLFWbool **rawMouseMotion**
- _GLFWcontext **context**
- struct {
  GLFWwindowposfun **pos**
  GLFWwindowsizefun **size**
  GLFWwindowclosefun **close**
  GLFWwindowrefreshfun **refresh**
  GLFWwindowfocusfun **focus**
  GLFWwindowiconifyfun **iconify**
  GLFWwindowmaximizefun **maximize**
  GLFWframebuffersizefun **fbsize**
  GLFWwindowcontentscalefun **scale**
  GLFWmousebuttonfun **mouseButton**
  GLFWcursorposfun **cursorPos**
  GLFWcursorenterfun **cursorEnter**
  GLFWscrollfun **scroll**
  GLFWkeyfun **key**
  GLFWcharfun **character**
  GLFWcharmodsfun **charmods**
  GLFWdropfun **drop**
  } **callbacks**

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

- lib/glfw/src/internal.h

26.65 _GLFWwindowNS Struct Reference

Public Attributes

- id **object**
- id **delegate**
- id **view**
- id **layer**
- GLFWbool **maximized**
- GLFWbool **occluded**
- GLFWbool **retina**
- int **width**
- int **height**
- int **fbWidth**
- int **fbHeight**
- float **xscale**
- float **yscale**
- double **cursorWarpDeltaX**
- double **cursorWarpDeltaY**

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

- lib/glfw/src/cocoa_platform.h

26.66 _GLFWwindowNull Struct Reference

Public Attributes

- int **xpos**
- int **ypos**
- int **width**
- int **height**
- char * **title**
- GLFWbool **visible**
- GLFWbool **iconified**
- GLFWbool **maximized**
- GLFWbool **resizable**
- GLFWbool **decorated**
- GLFWbool **floating**
- GLFWbool **transparent**
- float **opacity**

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

- lib/glfw/src/null_platform.h

26.67 _GLFWwindowWayland Struct Reference

Public Attributes

- int **width**
- int **height**
- GLFWbool **visible**
- GLFWbool **maximized**
- GLFWbool **hovered**
- GLFWbool **transparent**
- struct wl_surface * **surface**
- struct wl_egl_window * **native**
- struct wl_callback * **callback**
- struct {
  struct xdg_surface * **surface**
  struct xdg_toplevel * **toplevel**
  struct zxdg_toplevel_decoration_v1 * **decoration**
  } **xdg**
- _GLFWcursor * **currentCursor**
- double **cursorPosX**
- double **cursorPosY**
- char * **title**
- int **scale**
- _GLFWmonitor ** **monitors**
- int **monitorsCount**
- int **monitorsSize**
- struct {
  struct zwp_relative_pointer_v1 * **relativePointer**
  struct zwp_locked_pointer_v1 * **lockedPointer**
  } **pointerLock**
- struct zwp_idle_inhibitor_v1 * **idleInhibitor**
- GLFWbool **wasFullscreen**
- struct {
  GLFWbool **serverSide**
  struct wl_buffer * **buffer**
  _GLFWdecorationWayland **top**
  _GLFWdecorationWayland **left**
  _GLFWdecorationWayland **right**
  _GLFWdecorationWayland **bottom**
  int **focus**
  } **decorations**

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

- lib/glfw/src/wl_platform.h

26.68 _GLFWwindowWin32 Struct Reference

Public Attributes

- HWND **handle**
- HICON **bigIcon**
- HICON **smallIcon**
- GLFWbool **cursorTracked**
- GLFWbool **frameAction**
- GLFWbool **iconified**
- GLFWbool **maximized**
- GLFWbool **transparent**
- GLFWbool **scaleToMonitor**
- GLFWbool **keymenu**
- int **width**
- int **height**
- int **lastCursorPosX**
- int **lastCursorPosY**
- WCHAR **highSurrogate**

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

- lib/glfw/src/win32_platform.h

26.69 _GLFWwindowX11 Struct Reference

Public Attributes

- Colormap **colormap**
- Window **handle**
- Window **parent**
- XIC **ic**
- GLFWbool **overrideRedirect**
- GLFWbool **iconified**
- GLFWbool **maximized**
- GLFWbool **transparent**
- int **width**
- int **height**
- int **xpos**
- int **ypos**
- int **lastCursorPosX**
- int **lastCursorPosY**
- int **warpCursorPosX**
- int **warpCursorPosY**
- Time **keyPressTimes** [256]

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

- lib/glfw/src/x11_platform.h

26.70 _GLFWwndconfig Struct Reference

Public Attributes

- int **width**
- int **height**
- const char * **title**
- GLFWbool **resizable**
- GLFWbool **visible**
- GLFWbool **decorated**
- GLFWbool **focused**
- GLFWbool **autoIconify**
- GLFWbool **floating**
- GLFWbool **maximized**
- GLFWbool **centerCursor**
- GLFWbool **focusOnShow**
- GLFWbool **mousePassthrough**
- GLFWbool **scaleToMonitor**
- struct {
  GLFWbool **retina**
  char **frameName** [256]
  } **ns**
- struct {
  char **className** [256]
  char **instanceName** [256]
  } **x11**
- struct {
  GLFWbool **keymenu**
  } **win32**

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

- lib/glfw/src/internal.h

26.71 _thread_start_info Struct Reference

Public Attributes

- thrd_start_t mFunction
- void * mArg

26.71.1 Detailed Description

Information to pass to the new thread (what to run).

26.71.2 Member Data Documentation

26.71.2.1 mArg

```
void* _thread_start_info::mArg
```

Function argument for the thread function.

26.71.2.2 mFunction

```
thrd_start_t _thread_start_info::mFunction
```

Pointer to the function to be executed.

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

- lib/glfw/deps/tinycthread.c

26.72 _XINPUT_BATTERY_INFORMATION Struct Reference

Public Attributes

- BYTE **BatteryType**
- BYTE **BatteryLevel**

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

- lib/glfw/deps/mingw/xinput.h

26.73 _XINPUT_CAPABILITIES Struct Reference

Public Attributes

- BYTE **Type**
- BYTE **SubType**
- WORD **Flags**
- XINPUT_GAMEPAD **Gamepad**
- XINPUT_VIBRATION **Vibration**

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

- lib/glfw/deps/mingw/xinput.h

26.74 _XINPUT_GAMEPAD Struct Reference

Public Attributes

- WORD **wButtons**
- BYTE **bLeftTrigger**
- BYTE **bRightTrigger**
- SHORT **sThumbLX**
- SHORT **sThumbLY**
- SHORT **sThumbRX**
- SHORT **sThumbRY**

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

- lib/glfw/deps/mingw/xinput.h

26.75 _XINPUT_KEYSTROKE Struct Reference

Public Attributes

- WORD **VirtualKey**
- WCHAR **Unicode**
- WORD **Flags**
- BYTE **UserIndex**
- BYTE **HidCode**

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

- lib/glfw/deps/mingw/xinput.h

26.76 _XINPUT_STATE Struct Reference

Public Attributes

- DWORD **dwPacketNumber**
- XINPUT_GAMEPAD **Gamepad**

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

- lib/glfw/deps/mingw/xinput.h

26.77 _XINPUT_VIBRATION Struct Reference

Public Attributes

- WORD **wLeftMotorSpeed**
- WORD **wRightMotorSpeed**

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

- lib/glfw/deps/mingw/xinput.h

26.78 allocator_stats Struct Reference

Public Attributes

- size_t **total**
- size_t **current**
- size_t **maximum**

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

- lib/glfw/tests/allocator.c

26.79 CHANGEFILTERSTRUCT Struct Reference

Public Attributes

- DWORD **cbSize**
- DWORD **ExtStatus**

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

- lib/glfw/src/win32_platform.h

26.80 demo Struct Reference

Public Attributes

- GLFWwindow * **window**
- VkSurfaceKHR **surface**
- bool **use_staging_buffer**
- VkInstance **inst**
- VkPhysicalDevice **gpu**
- VkDevice **device**
- VkQueue **queue**
- VkPhysicalDeviceProperties **gpu_props**
- VkPhysicalDeviceFeatures **gpu_features**
- VkQueueFamilyProperties * **queue_props**
- uint32_t **graphics_queue_node_index**
- uint32_t **enabled_extension_count**
- uint32_t **enabled_layer_count**
- const char * **extension_names** [64]
- const char * **enabled_layers** [64]
- int **width**
- int **height**
- VkFormat **format**
- VkColorSpaceKHR **color_space**
- uint32_t **swapchainImageCount**
- VkSwapchainKHR **swapchain**
- SwapchainBuffers * **buffers**
- VkCommandPool **cmd_pool**
- struct {
  VkFormat **format**
  VkImage **image**
  VkDeviceMemory **mem**
  VkImageView **view**
  } **depth**
- struct texture_object **textures** [DEMO_TEXTURE_COUNT]
- struct {
  VkBuffer **buf**
  VkDeviceMemory **mem**
  VkPipelineVertexInputStateCreateInfo **vi**
  VkVertexInputBindingDescription **vi_bindings** [1]
  VkVertexInputAttributeDescription **vi_attrs** [2]
  } **vertices**
- VkCommandBuffer **setup_cmd**
- VkCommandBuffer **draw_cmd**
- VkPipelineLayout **pipeline_layout**
- VkDescriptorSetLayout **desc_layout**
- VkPipelineCache **pipelineCache**
- VkRenderPass **render_pass**
- VkPipeline **pipeline**

- VkShaderModule **vert_shader_module**
- VkShaderModule **frag_shader_module**
- VkDescriptorPool **desc_pool**
- VkDescriptorSet **desc_set**
- VkFramebuffer * **framebuffers**
- VkPhysicalDeviceMemoryProperties **memory_properties**
- int32_t **curFrame**
- int32_t **frameCount**
- bool **validate**
- bool **use_break**
- VkDebugReportCallbackEXT **msg_callback**
- float **depthStencil**
- float **depthIncrement**
- uint32_t **current_buffer**
- uint32_t **queue_count**

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

- lib/glfw/tests/triangle-vulkan.c

26.81 DICONDITION Struct Reference

Public Attributes

- LONG **lOffset**
- LONG **lPositiveCoefficient**
- LONG **lNegativeCoefficient**
- DWORD **dwPositiveSaturation**
- DWORD **dwNegativeSaturation**
- LONG **lDeadBand**

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

- lib/glfw/deps/mingw/dinput.h

26.82 DICONSTANTFORCE Struct Reference

Public Attributes

- LONG **lMagnitude**

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

- lib/glfw/deps/mingw/dinput.h

26.83 DICUSTOMFORCE Struct Reference

Public Attributes

- DWORD **cChannels**
- DWORD **dwSamplePeriod**
- DWORD **cSamples**
- LPLONG **rglForceData**

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

- lib/glfw/deps/mingw/dinput.h

26.84 DIDEVCAPS Struct Reference

Public Attributes

- DWORD **dwSize**
- DWORD **dwFlags**
- DWORD **dwDevType**
- DWORD **dwAxes**
- DWORD **dwButtons**
- DWORD **dwPOVs**
- DWORD **dwFFSamplePeriod**
- DWORD **dwFFMinTimeResolution**
- DWORD **dwFirmwareRevision**
- DWORD **dwHardwareRevision**
- DWORD **dwFFDriverVersion**

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

- lib/glfw/deps/mingw/dinput.h

26.85 DIDEVCAPS_DX3 Struct Reference

Public Attributes

- DWORD **dwSize**
- DWORD **dwFlags**
- DWORD **dwDevType**
- DWORD **dwAxes**
- DWORD **dwButtons**
- DWORD **dwPOVs**

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

- lib/glfw/deps/mingw/dinput.h

26.86 DIDEVICEINSTANCE_DX3A Struct Reference

Public Attributes

- DWORD **dwSize**
- GUID **guidInstance**
- GUID **guidProduct**
- DWORD **dwDevType**
- CHAR **tszInstanceName** [MAX_PATH]
- CHAR **tszProductName** [MAX_PATH]

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

- lib/glfw/deps/mingw/dinput.h

26.87 DIDEVICEINSTANCE_DX3W Struct Reference

Public Attributes

- DWORD **dwSize**
- GUID **guidInstance**
- GUID **guidProduct**
- DWORD **dwDevType**
- WCHAR **tszInstanceName** [MAX_PATH]
- WCHAR **tszProductName** [MAX_PATH]

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

- lib/glfw/deps/mingw/dinput.h

26.88 DIDEVICEINSTANCEA Struct Reference

Public Attributes

- DWORD **dwSize**
- GUID **guidInstance**
- GUID **guidProduct**
- DWORD **dwDevType**
- CHAR **tszInstanceName** [MAX_PATH]
- CHAR **tszProductName** [MAX_PATH]
- GUID **guidFFDriver**
- WORD **wUsagePage**
- WORD **wUsage**

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

- lib/glfw/deps/mingw/dinput.h

26.89 DIDEVICEINSTANCEW Struct Reference

Public Attributes

- DWORD **dwSize**
- GUID **guidInstance**
- GUID **guidProduct**
- DWORD **dwDevType**
- WCHAR **tszInstanceName** [MAX_PATH]
- WCHAR **tszProductName** [MAX_PATH]
- GUID **guidFFDriver**
- WORD **wUsagePage**
- WORD **wUsage**

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

- lib/glfw/deps/mingw/dinput.h

26.90 DIDEVICEOBJECTDATA Struct Reference

Public Attributes

- DWORD **dwOfs**
- DWORD **dwData**
- DWORD **dwTimeStamp**
- DWORD **dwSequence**
- UINT_PTR **uAppData**

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

- lib/glfw/deps/mingw/dinput.h

26.91 DIDEVICEOBJECTDATA_DX3 Struct Reference

Public Attributes

- DWORD **dwOfs**
- DWORD **dwData**
- DWORD **dwTimeStamp**
- DWORD **dwSequence**

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

- lib/glfw/deps/mingw/dinput.h

26.92 DIDEVICEOBJECTINSTANCE_DX3A Struct Reference

Public Attributes

- DWORD **dwSize**
- GUID **guidType**
- DWORD **dwOfs**
- DWORD **dwType**
- DWORD **dwFlags**
- CHAR **tszName** [MAX_PATH]

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

- lib/glfw/deps/mingw/dinput.h

26.93 DIDEVICEOBJECTINSTANCE_DX3W Struct Reference

Public Attributes

- DWORD **dwSize**
- GUID **guidType**
- DWORD **dwOfs**
- DWORD **dwType**
- DWORD **dwFlags**
- WCHAR **tszName** [MAX_PATH]

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

- lib/glfw/deps/mingw/dinput.h

26.94 DIDEVICEOBJECTINSTANCEA Struct Reference

Public Attributes

- DWORD **dwSize**
- GUID **guidType**
- DWORD **dwOfs**
- DWORD **dwType**
- DWORD **dwFlags**
- CHAR **tszName** [MAX_PATH]
- DWORD **dwFFMaxForce**
- DWORD **dwFFForceResolution**
- WORD **wCollectionNumber**
- WORD **wDesignatorIndex**
- WORD **wUsagePage**
- WORD **wUsage**
- DWORD **dwDimension**
- WORD **wExponent**
- WORD **wReserved**

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

- lib/glfw/deps/mingw/dinput.h

26.95 DIDEVICEOBJECTINSTANCEW Struct Reference

Public Attributes

- DWORD **dwSize**
- GUID **guidType**
- DWORD **dwOfs**
- DWORD **dwType**
- DWORD **dwFlags**
- WCHAR **tszName** [MAX_PATH]
- DWORD **dwFFMaxForce**
- DWORD **dwFFForceResolution**
- WORD **wCollectionNumber**
- WORD **wDesignatorIndex**
- WORD **wUsagePage**
- WORD **wUsage**
- DWORD **dwDimension**
- WORD **wExponent**
- WORD **wReserved**

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

- lib/glfw/deps/mingw/dinput.h

26.96 DIEFFECT Struct Reference

Public Attributes

- DWORD **dwSize**
- DWORD **dwFlags**
- DWORD **dwDuration**
- DWORD **dwSamplePeriod**
- DWORD **dwGain**
- DWORD **dwTriggerButton**
- DWORD **dwTriggerRepeatInterval**
- DWORD **cAxes**
- LPDWORD **rgdwAxes**
- LPLONG **rglDirection**
- LPDIENVELOPE **lpEnvelope**
- DWORD **cbTypeSpecificParams**
- LPVOID **lpvTypeSpecificParams**
- DWORD **dwStartDelay**

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

- lib/glfw/deps/mingw/dinput.h

26.97 DIEFFECT_DX5 Struct Reference

Public Attributes

- DWORD **dwSize**
- DWORD **dwFlags**
- DWORD **dwDuration**
- DWORD **dwSamplePeriod**
- DWORD **dwGain**
- DWORD **dwTriggerButton**
- DWORD **dwTriggerRepeatInterval**
- DWORD **cAxes**
- LPDWORD **rgdwAxes**
- LPLONG **rglDirection**
- LPDIENVELOPE **lpEnvelope**
- DWORD **cbTypeSpecificParams**
- LPVOID **lpvTypeSpecificParams**

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

- lib/glfw/deps/mingw/dinput.h

26.98 DIEFFECTINFOA Struct Reference

Public Attributes

- DWORD **dwSize**
- GUID **guid**
- DWORD **dwEffType**
- DWORD **dwStaticParams**
- DWORD **dwDynamicParams**
- CHAR **tszName** [MAX_PATH]

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

- lib/glfw/deps/mingw/dinput.h

26.99 DIEFFECTINFOW Struct Reference

Public Attributes

- DWORD **dwSize**
- GUID **guid**
- DWORD **dwEffType**
- DWORD **dwStaticParams**
- DWORD **dwDynamicParams**
- WCHAR **tszName** [MAX_PATH]

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

- lib/glfw/deps/mingw/dinput.h

26.100 DIEFFESCAPE Struct Reference

Public Attributes

- DWORD **dwSize**
- DWORD **dwCommand**
- LPVOID **lpvInBuffer**
- DWORD **cbInBuffer**
- LPVOID **lpvOutBuffer**
- DWORD **cbOutBuffer**

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

- lib/glfw/deps/mingw/dinput.h

26.101 DIENVELOPE Struct Reference

Public Attributes

- DWORD **dwSize**
- DWORD **dwAttackLevel**
- DWORD **dwAttackTime**
- DWORD **dwFadeLevel**
- DWORD **dwFadeTime**

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

- lib/glfw/deps/mingw/dinput.h

26.102 DIFILEEFFECT Struct Reference

Public Attributes

- DWORD **dwSize**
- GUID **GuidEffect**
- LPCDIEFFECT **lpDiEffect**
- CHAR **szFriendlyName** [MAX_PATH]

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

- lib/glfw/deps/mingw/dinput.h

26.103 DIJOYSTATE Struct Reference

Public Attributes

- LONG **lX**
- LONG **lY**
- LONG **lZ**
- LONG **lRx**
- LONG **lRy**
- LONG **lRz**
- LONG **rglSlider** [2]
- DWORD **rgdwPOV** [4]
- BYTE **rgbButtons** [32]

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

- lib/glfw/deps/mingw/dinput.h

26.104 DIJOYSTATE2 Struct Reference

Public Attributes

- LONG **lX**
- LONG **lY**
- LONG **lZ**
- LONG **lRx**
- LONG **lRy**
- LONG **lRz**
- LONG **rglSlider** [2]
- DWORD **rgdwPOV** [4]
- BYTE **rgbButtons** [128]
- LONG **lVX**
- LONG **lVY**
- LONG **lVZ**
- LONG **lVRx**
- LONG **lVRy**
- LONG **lVRz**
- LONG **rglVSlider** [2]
- LONG **lAX**
- LONG **lAY**
- LONG **lAZ**
- LONG **lARx**
- LONG **lARy**
- LONG **lARz**
- LONG **rglASlider** [2]
- LONG **lFX**
- LONG **lFY**
- LONG **lFZ**
- LONG **lFRx**
- LONG **lFRy**
- LONG **lFRz**
- LONG **rglFSlider** [2]

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

- lib/glfw/deps/mingw/dinput.h

26.105 DIMOUSESTATE Struct Reference

Public Attributes

- LONG **lX**
- LONG **lY**
- LONG **lZ**
- BYTE **rgbButtons** [4]

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

- lib/glfw/deps/mingw/dinput.h

26.106 DIMOUSESTATE2 Struct Reference

Public Attributes

- LONG **lX**
- LONG **lY**
- LONG **lZ**
- BYTE **rgbButtons** [8]

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

- lib/glfw/deps/mingw/dinput.h

26.107 DIPERIODIC Struct Reference

Public Attributes

- DWORD **dwMagnitude**
- LONG **lOffset**
- DWORD **dwPhase**
- DWORD **dwPeriod**

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

- lib/glfw/deps/mingw/dinput.h

26.108 DIPROPCAL Struct Reference

Public Attributes

- DIPROPHEADER **diph**
- LONG **lMin**
- LONG **lCenter**
- LONG **lMax**

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

- lib/glfw/deps/mingw/dinput.h

26.109 DIPROPCALPOV Struct Reference

Public Attributes

- DIPROPHEADER **diph**
- LONG **lMin** [5]
- LONG **lMax** [5]

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

- lib/glfw/deps/mingw/dinput.h

26.110 DIPROPCPOINTS Struct Reference

Public Attributes

- DIPROPHEADER **diph**
- DWORD **dwCPointsNum**
- CPOINT **cp** [MAXCPOINTSNUM]

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

- lib/glfw/deps/mingw/dinput.h

26.111 DIPROPDWORD Struct Reference

Public Attributes

- DIPROPHEADER **diph**
- DWORD **dwData**

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

- lib/glfw/deps/mingw/dinput.h

26.112 DIPROPGUIDANDPATH Struct Reference

Public Attributes

- DIPROPHEADER **diph**
- GUID **guidClass**
- WCHAR **wszPath** [MAX_PATH]

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

- lib/glfw/deps/mingw/dinput.h

26.113 DIPROPHEADER Struct Reference

Public Attributes

- DWORD **dwSize**
- DWORD **dwHeaderSize**
- DWORD **dwObj**
- DWORD **dwHow**

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

- lib/glfw/deps/mingw/dinput.h

26.114 DIPROPPOINTER Struct Reference

Public Attributes

- DIPROPHEADER **diph**
- UINT_PTR **uData**

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

- lib/glfw/deps/mingw/dinput.h

26.115 DIPROPRANGE Struct Reference

Public Attributes

- DIPROPHEADER **diph**
- LONG **lMin**
- LONG **lMax**

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

- lib/glfw/deps/mingw/dinput.h

26.116 DIPROPSTRING Struct Reference

Public Attributes

- DIPROPHEADER **diph**
- WCHAR **wsz** [MAX_PATH]

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

- lib/glfw/deps/mingw/dinput.h

26.117 DIRAMPFORCE Struct Reference

Public Attributes

- LONG **lStart**
- LONG **lEnd**

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

- lib/glfw/deps/mingw/dinput.h

26.118 DWM_BLURBEHIND Struct Reference

Public Attributes

- DWORD **dwFlags**
- BOOL **fEnable**
- HRGN **hRgnBlur**
- BOOL **fTransitionOnMaximized**

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

- lib/glfw/src/win32_platform.h

26.119 GLFWallocator Struct Reference

`#include <glfw3.h>`

Public Attributes

- GLFWallocatefun **allocate**
- GLFWreallocatefun **reallocate**
- GLFWdeallocatefun **deallocate**
- void * **user**

26.119.1 Detailed Description

See also

Custom heap memory allocator

glfwInitAllocator

Since

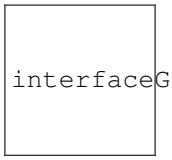
Added in version 3.4.

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

- lib/glfw/include/GLFW/glfw3.h

26.120 GLFWApplicationDelegate Class Reference

Inheritance diagram for GLFWApplicationDelegate:

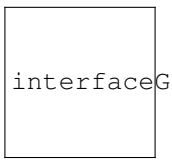


The documentation for this class was generated from the following file:

- lib/glfw/src/cocoa_init.m

26.121 GLFWContentView Class Reference

Inheritance diagram for GLFWContentView:



Instance Methods

- (instancetype) - **initWithGlfwWindow:**

Protected Attributes

- _GLFWwindow * **window**
- NSTrackingArea * **trackingArea**
- NSMutableAttributedString * **markedText**

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

- lib/glfw/src/cocoa_window.m

26.122 GLFWgamepadstate Struct Reference

Gamepad input state.

```
#include <glfw3.h>
```

Public Attributes

- unsigned char buttons [15]
- float axes [6]

26.122.1 Detailed Description

Gamepad input state.

This describes the input state of a gamepad.

See also

Gamepad input
glfwGetGamepadState

Since

Added in version 3.3.

26.122.2 Member Data Documentation

26.122.2.1 axes

```
float GLFWgamepadstate::axes[6]
```

The states of each gamepad axis, in the range -1.0 to 1.0 inclusive.

26.122.2.2 buttons

```
unsigned char GLFWgamepadstate::buttons[15]
```

The states of each gamepad button, `GLFW_PRESS` or `GLFW_RELEASE`.

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

- lib/glfw/include/GLFW/glfw3.h

26.123 GLFWgammaramp Struct Reference

Gamma ramp.

```
#include <glfw3.h>
```

Public Attributes

- unsigned short * red
- unsigned short * green
- unsigned short * blue
- unsigned int size

26.123.1 Detailed Description

Gamma ramp.

This describes the gamma ramp for a monitor.

See also

Gamma ramp
glfwGetGammaRamp
glfwSetGammaRamp

Since

Added in version 3.0.

26.123.2 Member Data Documentation

26.123.2.1 blue

```
unsigned short* GLFWgammaramp::blue
```

An array of value describing the response of the blue channel.

26.123.2.2 green

```
unsigned short* GLFWgammaramp::green
```

An array of value describing the response of the green channel.

26.123.2.3 red

```
unsigned short* GLFWgammaramp::red
```

An array of value describing the response of the red channel.

26.123.2.4 size

```
unsigned int GLFWgammaramp::size
```

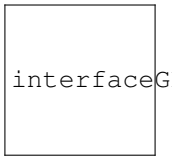
The number of elements in each array.

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

- lib/glfw/include/GLFW/glfw3.h

26.124 GLFWHelper Class Reference

Inheritance diagram for GLFWHelper:



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

- lib/glfw/src/cocoa_init.m

26.125 GLFWimage Struct Reference

Image data.

```
#include <glfw3.h>
```

Public Attributes

- int width
- int height
- unsigned char * pixels

26.125.1 Detailed Description

Image data.

This describes a single 2D image. See the documentation for each related function what the expected pixel format is.

See also

Custom cursor creation

Window icon

Since

Added in version 2.1. @glfw3 Removed format and bytes-per-pixel members.

26.125.2 Member Data Documentation

26.125.2.1 height

```
int GLFWimage::height
```

The height, in pixels, of this image.

26.125.2.2 pixels

```
unsigned char* GLFWimage::pixels
```

The pixel data of this image, arranged left-to-right, top-to-bottom.

26.125.2.3 width

```
int GLFWimage::width
```

The width, in pixels, of this image.

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

- lib/glfw/include/GLFW/glfw3.h

26.126 GLFWvidmode Struct Reference

Video mode type.

`#include <glfw3.h>`

Public Attributes

- int width
- int height
- int redBits
- int greenBits
- int blueBits
- int refreshRate

26.126.1 Detailed Description

Video mode type.

This describes a single video mode.

See also

Video modes
glfwGetVideoMode
glfwGetVideoModes

Since

Added in version 1.0. @glfw3 Added refresh rate member.

26.126.2 Member Data Documentation

26.126.2.1 blueBits

`int GLFWvidmode::blueBits`

The bit depth of the blue channel of the video mode.

26.126.2.2 greenBits

`int GLFWvidmode::greenBits`

The bit depth of the green channel of the video mode.

26.126.2.3 height

```
int GLFWvidmode::height
```

The height, in screen coordinates, of the video mode.

26.126.2.4 redBits

```
int GLFWvidmode::redBits
```

The bit depth of the red channel of the video mode.

26.126.2.5 refreshRate

```
int GLFWvidmode::refreshRate
```

The refresh rate, in Hz, of the video mode.

26.126.2.6 width

```
int GLFWvidmode::width
```

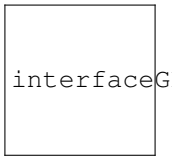
The width, in screen coordinates, of the video mode.

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

- lib/glfw/include/GLFW/glfw3.h

26.127 GLFWWindow Class Reference

Inheritance diagram for GLFWWindow:



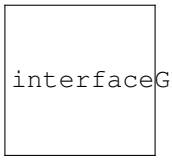
GLFWWindow-eps-converted-to.pdf

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

- lib/glfw/src/cocoa_window.m

26.128 GLFWWindowDelegate Class Reference

Inheritance diagram for GLFWWindowDelegate:



GLFWWindowDelegate-eps-converted-to.pdf

Instance Methods

- (instancetype) - **initWithGlfwWindow:**

Protected Attributes

- _GLFWwindow * **window**

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

- lib/glfw/src/cocoa_window.m

26.129 nk_allocator Struct Reference

Public Attributes

- nk_handle **userdata**
- nk_plugin_alloc **alloc**
- nk_plugin_free **free**

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

- lib/glfw/deps/nuklear.h

26.130 nk_buffer Struct Reference

Public Attributes

- struct nk_buffer_marker **marker** [NK_BUFFER_MAX]
- struct nk_allocator **pool**
- enum nk_allocation_type **type**
- struct nk_memory **memory**
- float **grow_factor**
- nk_size **allocated**
- nk_size **needed**
- nk_size **calls**
- nk_size **size**

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

- lib/glfw/deps/nuklear.h

26.131 nk_buffer_marker Struct Reference

Public Attributes

- int **active**
- nk_size **offset**

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

- lib/glfw/deps/nuklear.h

26.132 nk_chart Struct Reference

Public Attributes

- int **slot**
- float **x**
- float **y**
- float **w**
- float **h**
- struct nk_chart_slot **slots** [NK_CHART_MAX_SLOT]

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

- lib/glfw/deps/nuklear.h

26.133 nk_chart_slot Struct Reference

Public Attributes

- enum nk_chart_type **type**
- struct nk_color **color**
- struct nk_color **highlight**
- float **min**
- float **max**
- float **range**
- int **count**
- struct nk_vec2 **last**
- int **index**

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

- lib/glfw/deps/nuklear.h

26.134 nk_clipboard Struct Reference

Public Attributes

- nk_handle **userdata**
- nk_plugin_paste **paste**
- nk_plugin_copy **copy**

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

- lib/glfw/deps/nuklear.h

26.135 nk_color Struct Reference

Public Attributes

- nk_byte **r**
- nk_byte **g**
- nk_byte **b**
- nk_byte **a**

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

- lib/glfw/deps/nuklear.h

26.136 nk_colorf Struct Reference

Public Attributes

- float **r**
- float **g**
- float **b**
- float **a**

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

- lib/glfw/deps/nuklear.h

26.137 nk_command Struct Reference

Public Attributes

- enum nk_command_type **type**
- nk_size **next**

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

- lib/glfw/deps/nuklear.h

26.138 nk_command_arc Struct Reference

Public Attributes

- struct nk_command **header**
- short **cx**
- short **cy**
- unsigned short **r**
- unsigned short **line_thickness**
- float **a** [2]
- struct nk_color **color**

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

- lib/glfw/deps/nuklear.h

26.139 nk_command_arc_filled Struct Reference

Public Attributes

- struct nk_command **header**
- short **cx**
- short **cy**
- unsigned short **r**
- float **a** [2]
- struct nk_color **color**

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

- lib/glfw/deps/nuklear.h

26.140 nk_command_buffer Struct Reference

Public Attributes

- struct nk_buffer * **base**
- struct nk_rect **clip**
- int **use_clipping**
- nk_handle **userdata**
- nk_size **begin**
- nk_size **end**
- nk_size **last**

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

- lib/glfw/deps/nuklear.h

26.141 nk_command_circle Struct Reference

Public Attributes

- struct nk_command **header**
- short **x**
- short **y**
- unsigned short **line_thickness**
- unsigned short **w**
- unsigned short **h**
- struct nk_color **color**

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

- lib/glfw/deps/nuklear.h

26.142 nk_command_circle_filled Struct Reference

Public Attributes

- struct nk_command **header**
- short **x**
- short **y**
- unsigned short **w**
- unsigned short **h**
- struct nk_color **color**

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

- lib/glfw/deps/nuklear.h

26.143 nk_command_curve Struct Reference

Public Attributes

- struct nk_command **header**
- unsigned short **line_thickness**
- struct nk_vec2i **begin**
- struct nk_vec2i **end**
- struct nk_vec2i **ctrl** [2]
- struct nk_color **color**

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

- lib/glfw/deps/nuklear.h

26.144 nk_command_custom Struct Reference

Public Attributes

- struct nk_command **header**
- short **x**
- short **y**
- unsigned short **w**
- unsigned short **h**
- nk_handle **callback_data**
- nk_command_custom_callback **callback**

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

- lib/glfw/deps/nuklear.h

26.145 nk_command_image Struct Reference

Public Attributes

- struct nk_command **header**
- short **x**
- short **y**
- unsigned short **w**
- unsigned short **h**
- struct nk_image **img**
- struct nk_color **col**

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

- lib/glfw/deps/nuklear.h

26.146 nk_command_line Struct Reference

Public Attributes

- struct nk_command **header**
- unsigned short **line_thickness**
- struct nk_vec2i **begin**
- struct nk_vec2i **end**
- struct nk_color **color**

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

- lib/glfw/deps/nuklear.h

26.147 nk_command_polygon Struct Reference

Public Attributes

- struct nk_command **header**
- struct nk_color **color**
- unsigned short **line_thickness**
- unsigned short **point_count**
- struct nk_vec2i **points** [1]

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

- lib/glfw/deps/nuklear.h

26.148 nk_command_polygon_filled Struct Reference

Public Attributes

- struct nk_command **header**
- struct nk_color **color**
- unsigned short **point_count**
- struct nk_vec2i **points** [1]

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

- lib/glfw/deps/nuklear.h

26.149 nk_command_polyline Struct Reference

Public Attributes

- struct nk_command **header**
- struct nk_color **color**
- unsigned short **line_thickness**
- unsigned short **point_count**
- struct nk_vec2i **points** [1]

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

- lib/glfw/deps/nuklear.h

26.150 nk_command_rect Struct Reference

Public Attributes

- struct nk_command **header**
- unsigned short **rounding**
- unsigned short **line_thickness**
- short **x**
- short **y**
- unsigned short **w**
- unsigned short **h**
- struct nk_color **color**

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

- lib/glfw/deps/nuklear.h

26.151 nk_command_rect_filled Struct Reference

Public Attributes

- struct nk_command **header**
- unsigned short **rounding**
- short **x**
- short **y**
- unsigned short **w**
- unsigned short **h**
- struct nk_color **color**

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

- lib/glfw/deps/nuklear.h

26.152 nk_command_rect_multi_color Struct Reference

Public Attributes

- struct nk_command **header**
- short **x**
- short **y**
- unsigned short **w**
- unsigned short **h**
- struct nk_color **left**
- struct nk_color **top**
- struct nk_color **bottom**
- struct nk_color **right**

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

- lib/glfw/deps/nuklear.h

26.153 nk_command_scissor Struct Reference

Public Attributes

- struct nk_command **header**
- short **x**
- short **y**
- unsigned short **w**
- unsigned short **h**

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

- lib/glfw/deps/nuklear.h

26.154 nk_command_text Struct Reference

Public Attributes

- struct nk_command **header**
- const struct nk_user_font * **font**
- struct nk_color **background**
- struct nk_color **foreground**
- short **x**
- short **y**
- unsigned short **w**
- unsigned short **h**
- float **height**
- int **length**
- char **string** [1]

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

- lib/glfw/deps/nuklear.h

26.155 nk_command_triangle Struct Reference

Public Attributes

- struct nk_command **header**
- unsigned short **line_thickness**
- struct nk_vec2i **a**
- struct nk_vec2i **b**
- struct nk_vec2i **c**
- struct nk_color **color**

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

- lib/glfw/deps/nuklear.h

26.156 nk_command_triangle_filled Struct Reference

Public Attributes

- struct nk_command **header**
- struct nk_vec2i **a**
- struct nk_vec2i **b**
- struct nk_vec2i **c**
- struct nk_color **color**

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

- lib/glfw/deps/nuklear.h

26.157 nk_configuration_stacks Struct Reference

Public Attributes

- struct nk_config_stack_style_item **style_items**
- struct nk_config_stack_float **floats**
- struct nk_config_stack_vec2 **vectors**
- struct nk_config_stack_flags **flags**
- struct nk_config_stack_color **colors**
- struct nk_config_stack_user_font **fonts**
- struct nk_config_stack_button_behavior **button_behaviors**

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

- lib/glfw/deps/nuklear.h

26.158 nk_context Struct Reference

Public Attributes

- struct nk_input **input**
- struct nk_style **style**
- struct nk_buffer **memory**
- struct nk_clipboard **clip**
- nk_flags **last_widget_state**
- enum nk_button_behavior **button_behavior**
- struct nk_configuration_stacks **stacks**
- float **delta_time_seconds**
- struct nk_text_edit **text_edit**
- struct nk_command_buffer **overlay**
- int **build**
- int **use_pool**
- struct nk_pool **pool**
- struct nk_window * **begin**
- struct nk_window * **end**
- struct nk_window * **active**
- struct nk_window * **current**
- struct nk_page_element * **freelist**
- unsigned int **count**
- unsigned int **seq**

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

- lib/glfw/deps/nuklear.h

26.159 nk_convert_config Struct Reference

Public Attributes

- float **global_alpha**
- enum nk_anti_aliasing **line_AA**
- enum nk_anti_aliasing **shape_AA**
- unsigned **circle_segment_count**
- unsigned **arc_segment_count**
- unsigned **curve_segment_count**
- struct nk_draw_null_texture **null**
- const struct nk_draw_vertex_layout_element * **vertex_layout**
- nk_size **vertex_size**
- nk_size **vertex_alignment**

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

- lib/glfw/deps/nuklear.h

26.160 nk_cursor Struct Reference

Public Attributes

- struct nk_image **img**
- struct nk_vec2 size **offset**

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

- lib/glfw/deps/nuklear.h

26.161 nk_draw_null_texture Struct Reference

Public Attributes

- nk_handle **texture**
- struct nk_vec2 **uv**

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

- lib/glfw/deps/nuklear.h

26.162 nk_edit_state Struct Reference

Public Attributes

- nk_hash **name**
- unsigned int **seq**
- unsigned int **old**
- int **active**
- int **prev**
- int **cursor**
- int **sel_start**
- int **sel_end**
- struct nk_scroll **scrollbar**
- unsigned char **mode**
- unsigned char **single_line**

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

- lib/glfw/deps/nuklear.h

26.163 nk_handle Union Reference

Public Attributes

- void * **ptr**
- int **id**

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

- lib/glfw/deps/nuklear.h

26.164 nk_image Struct Reference

Public Attributes

- nk_handle **handle**
- unsigned short **w**
- unsigned short **h**
- unsigned short **region** [4]

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

- lib/glfw/deps/nuklear.h

26.165 nk_input Struct Reference

Public Attributes

- struct nk_keyboard **keyboard**
- struct nk_mouse **mouse**

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

- lib/glfw/deps/nuklear.h

26.166 nk_key Struct Reference

Public Attributes

- int **down**
- unsigned int **clicked**

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

- lib/glfw/deps/nuklear.h

26.167 nk_keyboard Struct Reference

Public Attributes

- struct nk_key **keys** [NK_KEY_MAX]
- char **text** [NK_INPUT_MAX]
- int **text_len**

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

- lib/glfw/deps/nuklear.h

26.168 nk_list_view Struct Reference

Public Attributes

- int **begin**
- int **end**
- int **count**
- int **total_height**
- struct nk_context * **ctx**
- nk_uint * **scroll_pointer**
- nk_uint **scroll_value**

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

- lib/glfw/deps/nuklear.h

26.169 nk_memory Struct Reference

Public Attributes

- void * **ptr**
- nk_size **size**

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

- lib/glfw/deps/nuklear.h

26.170 nk_memory_status Struct Reference

Public Attributes

- void * **memory**
- unsigned int **type**
- nk_size **size**
- nk_size **allocated**
- nk_size **needed**
- nk_size **calls**

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

- lib/glfw/deps/nuklear.h

26.171 nk_menu_state Struct Reference

Public Attributes

- float **x**
- float **y**
- float **w**
- float **h**
- struct nk_scroll **offset**

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

- lib/glfw/deps/nuklear.h

26.172 nk_mouse Struct Reference

Public Attributes

- struct nk_mouse_button **buttons** [NK_BUTTON_MAX]
- struct nk_vec2 **pos**
- struct nk_vec2 **prev**
- struct nk_vec2 **delta**
- struct nk_vec2 **scroll_delta**
- unsigned char **grab**
- unsigned char **grabbed**
- unsigned char **ungrab**

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

- lib/glfw/deps/nuklear.h

26.173 nk_mouse_button Struct Reference

Public Attributes

- int **down**
- unsigned int **clicked**
- struct nk_vec2 **clicked_pos**

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

- lib/glfw/deps/nuklear.h

26.174 nk_page Struct Reference

Public Attributes

- unsigned int **size**
- struct nk_page * **next**
- struct nk_page_element **win** [1]

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

- lib/glfw/deps/nuklear.h

26.175 nk_page_data Union Reference

Public Attributes

- struct nk_table **tbl**
- struct nk_panel **pan**
- struct nk_window **win**

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

- lib/glfw/deps/nuklear.h

26.176 nk_page_element Struct Reference

Public Attributes

- union nk_page_data **data**
- struct nk_page_element * **next**
- struct nk_page_element * **prev**

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

- lib/glfw/deps/nuklear.h

26.177 nk_panel Struct Reference

Public Attributes

- enum nk_panel_type **type**
- nk_flags **flags**
- struct nk_rect **bounds**
- nk_uint * **offset_x**
- nk_uint * **offset_y**
- float **at_x**
- float **at_y**
- float **max_x**
- float **footer_height**
- float **header_height**
- float **border**
- unsigned int **has_scrolling**
- struct nk_rect **clip**
- struct nk_menu_state **menu**
- struct nk_row_layout **row**
- struct nk_chart **chart**
- struct nk_command_buffer * **buffer**
- struct nk_panel * **parent**

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

- lib/glfw/deps/nuklear.h

26.178 nk_pool Struct Reference

Public Attributes

- struct nk_allocator **alloc**
- enum nk_allocation_type **type**
- unsigned int **page_count**
- struct nk_page * **pages**
- struct nk_page_element * **freelist**
- unsigned **capacity**
- nk_size **size**
- nk_size **cap**

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

- lib/glfw/deps/nuklear.h

26.179 nk_popup_buffer Struct Reference

Public Attributes

- nk_size **begin**
- nk_size **parent**
- nk_size **last**
- nk_size **end**
- int **active**

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

- lib/glfw/deps/nuklear.h

26.180 nk_popup_state Struct Reference

Public Attributes

- struct nk_window * **win**
- enum nk_panel_type **type**
- struct nk_popup_buffer **buf**
- nk_hash **name**
- int **active**
- unsigned **combo_count**
- unsigned **con_count**
- unsigned **con_old**
- unsigned **active_con**
- struct nk_rect **header**

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

- lib/glfw/deps/nuklear.h

26.181 nk_property_state Struct Reference

Public Attributes

- int **active**
- int **prev**
- char **buffer** [NK_MAX_NUMBER_BUFFER]
- int **length**
- int **cursor**
- int **select_start**
- int **select_end**
- nk_hash **name**
- unsigned int **seq**
- unsigned int **old**
- int **state**

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

- lib/glfw/deps/nuklear.h

26.182 nk_rect Struct Reference

Public Attributes

- float **x**
- float **y**
- float **w**
- float **h**

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

- lib/glfw/deps/nuklear.h

26.183 nk_recti Struct Reference

Public Attributes

- short **x**
- short **y**
- short **w**
- short **h**

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

- lib/glfw/deps/nuklear.h

26.184 nk_row_layout Struct Reference

Public Attributes

- enum nk_panel_row_layout_type **type**
- int **index**
- float **height**
- float **min_height**
- int **columns**
- const float * **ratio**
- float **item_width**
- float **item_height**
- float **item_offset**
- float **filled**
- struct nk_rect **item**
- int **tree_depth**
- float **templates** [NK_MAX_LAYOUT_ROW_TEMPLATE_COLUMNS]

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

- lib/glfw/deps/nuklear.h

26.185 nk_scroll Struct Reference

Public Attributes

- nk_uint **x**
- nk_uint **y**

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

- lib/glfw/deps/nuklear.h

26.186 nk_str Struct Reference

Public Attributes

- struct nk_buffer **buffer**
- int **len**

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

- lib/glfw/deps/nuklear.h

26.187 nk_style Struct Reference

Public Attributes

- const struct nk_user_font * **font**
- const struct nk_cursor * **cursors** [NK_CURSOR_COUNT]
- const struct nk_cursor * **cursor_active**
- struct nk_cursor * **cursor_last**
- int **cursor_visible**
- struct nk_style_text **text**
- struct nk_style_button **button**
- struct nk_style_button **contextual_button**
- struct nk_style_button **menu_button**
- struct nk_style_toggle **option**
- struct nk_style_toggle **checkbox**
- struct nk_style_selectable **selectable**
- struct nk_style_slider **slider**
- struct nk_style_progress **progress**
- struct nk_style_property **property**
- struct nk_style_edit **edit**
- struct nk_style_chart **chart**
- struct nk_style_scrollbar **scrollh**
- struct nk_style_scrollbar **scrollv**
- struct nk_style_tab **tab**
- struct nk_style_combo **combo**
- struct nk_style_window **window**

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

- lib/glfw/deps/nuklear.h

26.188 nk_style_button Struct Reference

Public Attributes

- struct nk_style_item **normal**
- struct nk_style_item **hover**
- struct nk_style_item **active**
- struct nk_color **border_color**
- struct nk_color **text_background**
- struct nk_color **text_normal**
- struct nk_color **text_hover**
- struct nk_color **text_active**
- nk_flags **text_alignment**
- float **border**
- float **rounding**
- struct nk_vec2 **padding**
- struct nk_vec2 **image_padding**
- struct nk_vec2 **touch_padding**
- nk_handle **userdata**
- void(* **draw_begin**)(struct nk_command_buffer *, nk_handle userdata)
- void(* **draw_end**)(struct nk_command_buffer *, nk_handle userdata)

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

- lib/glfw/deps/nuklear.h

26.189 nk_style_chart Struct Reference

Public Attributes

- struct nk_style_item **background**
- struct nk_color **border_color**
- struct nk_color **selected_color**
- struct nk_color **color**
- float **border**
- float **rounding**
- struct nk_vec2 **padding**

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

- lib/glfw/deps/nuklear.h

26.190 nk_style_combo Struct Reference

Public Attributes

- struct nk_style_item **normal**
- struct nk_style_item **hover**
- struct nk_style_item **active**
- struct nk_color **border_color**
- struct nk_color **label_normal**
- struct nk_color **label_hover**
- struct nk_color **label_active**
- struct nk_color **symbol_normal**
- struct nk_color **symbol_hover**
- struct nk_color **symbol_active**
- struct nk_style_button **button**
- enum nk_symbol_type **sym_normal**
- enum nk_symbol_type **sym_hover**
- enum nk_symbol_type **sym_active**
- float **border**
- float **rounding**
- struct nk_vec2 **content_padding**
- struct nk_vec2 **button_padding**
- struct nk_vec2 **spacing**

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

- lib/glfw/deps/nuklear.h

26.191 nk_style_edit Struct Reference

Public Attributes

- struct nk_style_item **normal**
- struct nk_style_item **hover**
- struct nk_style_item **active**
- struct nk_color **border_color**
- struct nk_style_scrollbar **scrollbar**
- struct nk_color **cursor_normal**
- struct nk_color **cursor_hover**
- struct nk_color **cursor_text_normal**
- struct nk_color **cursor_text_hover**
- struct nk_color **text_normal**
- struct nk_color **text_hover**
- struct nk_color **text_active**
- struct nk_color **selected_normal**
- struct nk_color **selected_hover**
- struct nk_color **selected_text_normal**
- struct nk_color **selected_text_hover**
- float **border**
- float **rounding**
- float **cursor_size**
- struct nk_vec2 **scrollbar_size**
- struct nk_vec2 **padding**
- float **row_padding**

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

- lib/glfw/deps/nuklear.h

26.192 nk_style_item Struct Reference

Public Attributes

- enum nk_style_item_type **type**
- union nk_style_item_data **data**

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

- lib/glfw/deps/nuklear.h

26.193 nk_style_item_data Union Reference

Public Attributes

- struct nk_image **image**
- struct nk_color **color**

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

- lib/glfw/deps/nuklear.h

26.194 nk_style_progress Struct Reference

Public Attributes

- struct nk_style_item **normal**
- struct nk_style_item **hover**
- struct nk_style_item **active**
- struct nk_color **border_color**
- struct nk_style_item **cursor_normal**
- struct nk_style_item **cursor_hover**
- struct nk_style_item **cursor_active**
- struct nk_color **cursor_border_color**
- float **rounding**
- float **border**
- float **cursor_border**
- float **cursor_rounding**
- struct nk_vec2 **padding**
- nk_handle **userdata**
- void(∗ **draw_begin**)(struct nk_command_buffer ∗, nk_handle)
- void(∗ **draw_end**)(struct nk_command_buffer ∗, nk_handle)

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

- lib/glfw/deps/nuklear.h

26.195 nk_style_property Struct Reference

Public Attributes

- struct nk_style_item **normal**
- struct nk_style_item **hover**
- struct nk_style_item **active**
- struct nk_color **border_color**
- struct nk_color **label_normal**
- struct nk_color **label_hover**
- struct nk_color **label_active**
- enum nk_symbol_type **sym_left**
- enum nk_symbol_type **sym_right**
- float **border**
- float **rounding**
- struct nk_vec2 **padding**
- struct nk_style_edit **edit**
- struct nk_style_button **inc_button**
- struct nk_style_button **dec_button**
- nk_handle **userdata**
- void(∗ **draw_begin**)(struct nk_command_buffer ∗, nk_handle)
- void(∗ **draw_end**)(struct nk_command_buffer ∗, nk_handle)

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

- lib/glfw/deps/nuklear.h

26.196 nk_style_scrollbar Struct Reference

Public Attributes

- struct nk_style_item **normal**
- struct nk_style_item **hover**
- struct nk_style_item **active**
- struct nk_color **border_color**
- struct nk_style_item **cursor_normal**
- struct nk_style_item **cursor_hover**
- struct nk_style_item **cursor_active**
- struct nk_color **cursor_border_color**
- float **border**
- float **rounding**
- float **border_cursor**
- float **rounding_cursor**
- struct nk_vec2 **padding**
- int **show_buttons**
- struct nk_style_button **inc_button**
- struct nk_style_button **dec_button**
- enum nk_symbol_type **inc_symbol**
- enum nk_symbol_type **dec_symbol**
- nk_handle **userdata**
- void(∗ **draw_begin**)(struct nk_command_buffer ∗, nk_handle)
- void(∗ **draw_end**)(struct nk_command_buffer ∗, nk_handle)

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

- lib/glfw/deps/nuklear.h

26.197 nk_style_selectable Struct Reference

Public Attributes

- struct nk_style_item **normal**
- struct nk_style_item **hover**
- struct nk_style_item **pressed**
- struct nk_style_item **normal_active**
- struct nk_style_item **hover_active**
- struct nk_style_item **pressed_active**
- struct nk_color **text_normal**
- struct nk_color **text_hover**
- struct nk_color **text_pressed**
- struct nk_color **text_normal_active**
- struct nk_color **text_hover_active**
- struct nk_color **text_pressed_active**
- struct nk_color **text_background**
- nk_flags **text_alignment**
- float **rounding**
- struct nk_vec2 **padding**
- struct nk_vec2 **touch_padding**
- struct nk_vec2 **image_padding**
- nk_handle **userdata**
- void(∗ **draw_begin**)(struct nk_command_buffer ∗, nk_handle)
- void(∗ **draw_end**)(struct nk_command_buffer ∗, nk_handle)

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

- lib/glfw/deps/nuklear.h

26.198 nk_style_slider Struct Reference

Public Attributes

- struct nk_style_item **normal**
- struct nk_style_item **hover**
- struct nk_style_item **active**
- struct nk_color **border_color**
- struct nk_color **bar_normal**
- struct nk_color **bar_hover**
- struct nk_color **bar_active**
- struct nk_color **bar_filled**
- struct nk_style_item **cursor_normal**
- struct nk_style_item **cursor_hover**
- struct nk_style_item **cursor_active**
- float **border**
- float **rounding**
- float **bar_height**
- struct nk_vec2 **padding**
- struct nk_vec2 **spacing**
- struct nk_vec2 **cursor_size**
- int **show_buttons**
- struct nk_style_button **inc_button**
- struct nk_style_button **dec_button**
- enum nk_symbol_type **inc_symbol**
- enum nk_symbol_type **dec_symbol**
- nk_handle **userdata**
- void(∗ **draw_begin**)(struct nk_command_buffer ∗, nk_handle)
- void(∗ **draw_end**)(struct nk_command_buffer ∗, nk_handle)

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

- lib/glfw/deps/nuklear.h

26.199 nk_style_tab Struct Reference

Public Attributes

- struct nk_style_item **background**
- struct nk_color **border_color**
- struct nk_color **text**
- struct nk_style_button **tab_maximize_button**
- struct nk_style_button **tab_minimize_button**
- struct nk_style_button **node_maximize_button**
- struct nk_style_button **node_minimize_button**
- enum nk_symbol_type **sym_minimize**
- enum nk_symbol_type **sym_maximize**
- float **border**
- float **rounding**
- float **indent**
- struct nk_vec2 **padding**
- struct nk_vec2 **spacing**

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

- lib/glfw/deps/nuklear.h

26.200 nk_style_text Struct Reference

Public Attributes

- struct nk_color **color**
- struct nk_vec2 **padding**

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

- lib/glfw/deps/nuklear.h

26.201 nk_style_toggle Struct Reference

Public Attributes

- struct nk_style_item **normal**
- struct nk_style_item **hover**
- struct nk_style_item **active**
- struct nk_color **border_color**
- struct nk_style_item **cursor_normal**
- struct nk_style_item **cursor_hover**
- struct nk_color **text_normal**
- struct nk_color **text_hover**
- struct nk_color **text_active**
- struct nk_color **text_background**
- nk_flags **text_alignment**
- struct nk_vec2 **padding**
- struct nk_vec2 **touch_padding**
- float **spacing**
- float **border**
- nk_handle **userdata**
- void(∗ **draw_begin**)(struct nk_command_buffer ∗, nk_handle)
- void(∗ **draw_end**)(struct nk_command_buffer ∗, nk_handle)

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

- lib/glfw/deps/nuklear.h

26.202 nk_style_window Struct Reference

Public Attributes

- struct nk_style_window_header **header**
- struct nk_style_item **fixed_background**
- struct nk_color **background**
- struct nk_color **border_color**
- struct nk_color **popup_border_color**
- struct nk_color **combo_border_color**
- struct nk_color **contextual_border_color**
- struct nk_color **menu_border_color**
- struct nk_color **group_border_color**
- struct nk_color **tooltip_border_color**
- struct nk_style_item **scaler**
- float **border**
- float **combo_border**
- float **contextual_border**
- float **menu_border**
- float **group_border**
- float **tooltip_border**
- float **popup_border**
- float **min_row_height_padding**
- float **rounding**
- struct nk_vec2 **spacing**
- struct nk_vec2 **scrollbar_size**
- struct nk_vec2 **min_size**
- struct nk_vec2 **padding**
- struct nk_vec2 **group_padding**
- struct nk_vec2 **popup_padding**
- struct nk_vec2 **combo_padding**
- struct nk_vec2 **contextual_padding**
- struct nk_vec2 **menu_padding**
- struct nk_vec2 **tooltip_padding**

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

- lib/glfw/deps/nuklear.h

26.203 nk_style_window_header Struct Reference

Public Attributes

- struct nk_style_item **normal**
- struct nk_style_item **hover**
- struct nk_style_item **active**
- struct nk_style_button **close_button**
- struct nk_style_button **minimize_button**
- enum nk_symbol_type **close_symbol**
- enum nk_symbol_type **minimize_symbol**
- enum nk_symbol_type **maximize_symbol**

- struct nk_color **label_normal**
- struct nk_color **label_hover**
- struct nk_color **label_active**
- enum nk_style_header_align **align**
- struct nk_vec2 **padding**
- struct nk_vec2 **label_padding**
- struct nk_vec2 **spacing**

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

- lib/glfw/deps/nuklear.h

26.204 nk_table Struct Reference

Public Attributes

- unsigned int **seq**
- unsigned int **size**
- nk_hash **keys** [NK_VALUE_PAGE_CAPACITY]
- nk_uint **values** [NK_VALUE_PAGE_CAPACITY]
- struct nk_table * **next**
- struct nk_table * **prev**

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

- lib/glfw/deps/nuklear.h

26.205 nk_text_edit Struct Reference

Public Attributes

- struct nk_clipboard **clip**
- struct nk_str **string**
- nk_plugin_filter **filter**
- struct nk_vec2 **scrollbar**
- int **cursor**
- int **select_start**
- int **select_end**
- unsigned char **mode**
- unsigned char **cursor_at_end_of_line**
- unsigned char **initialized**
- unsigned char **has_preferred_x**
- unsigned char **single_line**
- unsigned char **active**
- unsigned char **padding1**
- float **preferred_x**
- struct nk_text_undo_state **undo**

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

- lib/glfw/deps/nuklear.h

26.206 nk_text_undo_record Struct Reference

Public Attributes

- int **where**
- short **insert_length**
- short **delete_length**
- short **char_storage**

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

- lib/glfw/deps/nuklear.h

26.207 nk_text_undo_state Struct Reference

Public Attributes

- struct nk_text_undo_record **undo_rec** [NK_TEXTEDIT_UNDOSTATECOUNT]
- nk_rune **undo_char** [NK_TEXTEDIT_UNDOCHARCOUNT]
- short **undo_point**
- short **redo_point**
- short **undo_char_point**
- short **redo_char_point**

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

- lib/glfw/deps/nuklear.h

26.208 nk_user_font Struct Reference

Public Attributes

- nk_handle **userdata**
- float **height**
- nk_text_width_f **width**

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

- lib/glfw/deps/nuklear.h

26.209 nk_vec2 Struct Reference

Public Attributes

- float **x**
- float **y**

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

- lib/glfw/deps/nuklear.h

26.210 nk_vec2i Struct Reference

Public Attributes

- short **x**
- short **y**

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

- lib/glfw/deps/nuklear.h

26.211 nk_window Struct Reference

Public Attributes

- unsigned int **seq**
- nk_hash **name**
- char **name_string** [NK_WINDOW_MAX_NAME]
- nk_flags **flags**
- struct nk_rect **bounds**
- struct nk_scroll **scrollbar**
- struct nk_command_buffer **buffer**
- struct nk_panel * **layout**
- float **scrollbar_hiding_timer**
- struct nk_property_state **property**
- struct nk_popup_state **popup**
- struct nk_edit_state **edit**
- unsigned int **scrolled**
- struct nk_table * **tables**
- unsigned int **table_count**
- struct nk_window * **next**
- struct nk_window * **prev**
- struct nk_window * **parent**

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

- lib/glfw/deps/nuklear.h

26.212 option Struct Reference

Public Attributes

- const char * **name**
- int **has_arg**
- int * **flag**
- int **val**

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

- lib/glfw/deps/getopt.h

26.213 PARTICLE Struct Reference

Public Attributes

- float **x**
- float **y**
- float **z**
- float **vx**
- float **vy**
- float **vz**
- float **r**
- float **g**
- float **b**
- float **life**
- int **active**

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

- lib/glfw/examples/particles.c

26.214 Slot Struct Reference

Public Attributes

- GLFWwindow * **window**
- int **number**
- int **closeable**

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

- lib/glfw/tests/events.c

26.215 SwapchainBuffers Struct Reference

Public Attributes

- VkImage **image**
- VkCommandBuffer **cmd**
- VkImageView **view**

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

- lib/glfw/tests/triangle-vulkan.c

26.216 texture_object Struct Reference

Public Attributes

- VkSampler **sampler**
- VkImage **image**
- VkImageLayout **imageLayout**
- VkDeviceMemory **mem**
- VkImageView **view**
- int32_t **tex_width**
- int32_t **tex_height**

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

- lib/glfw/tests/triangle-vulkan.c

26.217 Thread Struct Reference

Public Attributes

- GLFWwindow * **window**
- const char * **title**
- float **r**
- float **g**
- float **b**
- thrd_t **id**

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

- lib/glfw/tests/threads.c

26.218 Vec3 Struct Reference

Public Attributes

- float **x**
- float **y**
- float **z**

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

- lib/glfw/examples/particles.c

26.219 Vertex Struct Reference

Public Attributes

- GLfloat **s**
- GLfloat **t**
- GLuint **rgba**
- GLfloat **x**
- GLfloat **y**
- GLfloat **z**
- vec2 **pos**
- vec3 **col**
- GLfloat **r**
- GLfloat **g**
- GLfloat **b**

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

- lib/glfw/examples/particles.c
- lib/glfw/examples/triangle-opengl.c
- lib/glfw/examples/triangle-opengles.c
- lib/glfw/examples/wave.c

26.220 vertex_t Struct Reference

Public Attributes

- float **x**
- float **y**
- float **z**

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

- lib/glfw/examples/boing.c

26.221 VkAcquireNextImageInfoKHR Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkSwapchainKHR **swapchain**
- uint64_t **timeout**
- VkSemaphore **semaphore**
- VkFence **fence**
- uint32_t **deviceMask**

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

- lib/glfw/deps/glad/vulkan.h

26.222 VkAllocationCallbacks Struct Reference

Public Attributes

- void * **pUserData**
- PFN_vkAllocationFunction **pfnAllocation**
- PFN_vkReallocationFunction **pfnReallocation**
- PFN_vkFreeFunction **pfnFree**
- PFN_vkInternalAllocationNotification **pfnInternalAllocation**
- PFN_vkInternalFreeNotification **pfnInternalFree**

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

- lib/glfw/deps/glad/vulkan.h

26.223 VkApplicationInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- const char * **pApplicationName**
- uint32_t **applicationVersion**
- const char * **pEngineName**
- uint32_t **engineVersion**
- uint32_t **apiVersion**

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

- lib/glfw/deps/glad/vulkan.h

26.224 VkAttachmentDescription Struct Reference

Public Attributes

- VkAttachmentDescriptionFlags **flags**
- VkFormat **format**
- VkSampleCountFlagBits **samples**
- VkAttachmentLoadOp **loadOp**
- VkAttachmentStoreOp **storeOp**
- VkAttachmentLoadOp **stencilLoadOp**
- VkAttachmentStoreOp **stencilStoreOp**
- VkImageLayout **initialLayout**
- VkImageLayout **finalLayout**

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

- lib/glfw/deps/glad/vulkan.h

26.225 VkAttachmentReference Struct Reference

Public Attributes

- uint32_t **attachment**
- VkImageLayout **layout**

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

- lib/glfw/deps/glad/vulkan.h

26.226 VkBaseInStructure Struct Reference

Public Attributes

- VkStructureType **sType**
- const struct VkBaseInStructure * **pNext**

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

- lib/glfw/deps/glad/vulkan.h

26.227 VkBaseOutStructure Struct Reference

Public Attributes

- VkStructureType **sType**
- struct VkBaseOutStructure * **pNext**

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

- lib/glfw/deps/glad/vulkan.h

26.228 VkBindBufferMemoryDeviceGroupInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- uint32_t **deviceIndexCount**
- const uint32_t * **pDeviceIndices**

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

- lib/glfw/deps/glad/vulkan.h

26.229 VkBindBufferMemoryInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkBuffer **buffer**
- VkDeviceMemory **memory**
- VkDeviceSize **memoryOffset**

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

- lib/glfw/deps/glad/vulkan.h

26.230 VkBindImageMemoryDeviceGroupInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- uint32_t **deviceIndexCount**
- const uint32_t * **pDeviceIndices**
- uint32_t **splitInstanceBindRegionCount**
- const VkRect2D * **pSplitInstanceBindRegions**

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

- lib/glfw/deps/glad/vulkan.h

26.231 VkBindImageMemoryInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkImage **image**
- VkDeviceMemory **memory**
- VkDeviceSize **memoryOffset**

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

- lib/glfw/deps/glad/vulkan.h

26.232 VkBindImageMemorySwapchainInfoKHR Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkSwapchainKHR **swapchain**
- uint32_t **imageIndex**

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

- lib/glfw/deps/glad/vulkan.h

26.233 VkBindImagePlaneMemoryInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkImageAspectFlagBits **planeAspect**

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

- lib/glfw/deps/glad/vulkan.h

26.234 VkBindSparseInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- uint32_t **waitSemaphoreCount**
- const VkSemaphore * **pWaitSemaphores**
- uint32_t **bufferBindCount**
- const VkSparseBufferMemoryBindInfo * **pBufferBinds**
- uint32_t **imageOpaqueBindCount**
- const VkSparseImageOpaqueMemoryBindInfo * **pImageOpaqueBinds**
- uint32_t **imageBindCount**
- const VkSparseImageMemoryBindInfo * **pImageBinds**
- uint32_t **signalSemaphoreCount**
- const VkSemaphore * **pSignalSemaphores**

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

- lib/glfw/deps/glad/vulkan.h

26.235 VkBufferCopy Struct Reference

Public Attributes

- VkDeviceSize **srcOffset**
- VkDeviceSize **dstOffset**
- VkDeviceSize **size**

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

- lib/glfw/deps/glad/vulkan.h

26.236 VkBufferCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkBufferCreateFlags **flags**
- VkDeviceSize **size**
- VkBufferUsageFlags **usage**
- VkSharingMode **sharingMode**
- uint32_t **queueFamilyIndexCount**
- const uint32_t * **pQueueFamilyIndices**

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

- lib/glfw/deps/glad/vulkan.h

26.237 VkBufferImageCopy Struct Reference

Public Attributes

- VkDeviceSize **bufferOffset**
- uint32_t **bufferRowLength**
- uint32_t **bufferImageHeight**
- VkImageSubresourceLayers **imageSubresource**
- VkOffset3D **imageOffset**
- VkExtent3D **imageExtent**

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

- lib/glfw/deps/glad/vulkan.h

26.238 VkBufferMemoryBarrier Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkAccessFlags **srcAccessMask**
- VkAccessFlags **dstAccessMask**
- uint32_t **srcQueueFamilyIndex**
- uint32_t **dstQueueFamilyIndex**
- VkBuffer **buffer**
- VkDeviceSize **offset**
- VkDeviceSize **size**

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

- lib/glfw/deps/glad/vulkan.h

26.239 VkBufferMemoryRequirementsInfo2 Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkBuffer **buffer**

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

- lib/glfw/deps/glad/vulkan.h

26.240 VkBufferViewCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkBufferViewCreateFlags **flags**
- VkBuffer **buffer**
- VkFormat **format**
- VkDeviceSize **offset**
- VkDeviceSize **range**

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

- lib/glfw/deps/glad/vulkan.h

26.241 VkClearAttachment Struct Reference

Public Attributes

- VkImageAspectFlags **aspectMask**
- uint32_t **colorAttachment**
- VkClearValue **clearValue**

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

- lib/glfw/deps/glad/vulkan.h

26.242 VkClearColorValue Union Reference

Public Attributes

- float **float32** [4]
- int32_t **int32** [4]
- uint32_t **uint32** [4]

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

- lib/glfw/deps/glad/vulkan.h

26.243 VkClearDepthStencilValue Struct Reference

Public Attributes

- float **depth**
- uint32_t **stencil**

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

- lib/glfw/deps/glad/vulkan.h

26.244 VkClearRect Struct Reference

Public Attributes

- VkRect2D **rect**
- uint32_t **baseArrayLayer**
- uint32_t **layerCount**

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

- lib/glfw/deps/glad/vulkan.h

26.245 VkClearValue Union Reference

Public Attributes

- VkClearColorValue **color**
- VkClearDepthStencilValue **depthStencil**

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

- lib/glfw/deps/glad/vulkan.h

26.246 VkCommandBufferAllocateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkCommandPool **commandPool**
- VkCommandBufferLevel **level**
- uint32_t **commandBufferCount**

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

- lib/glfw/deps/glad/vulkan.h

26.247 VkCommandBufferBeginInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkCommandBufferUsageFlags **flags**
- const VkCommandBufferInheritanceInfo * **pInheritanceInfo**

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

- lib/glfw/deps/glad/vulkan.h

26.248 VkCommandBufferInheritanceInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkRenderPass **renderPass**
- uint32_t **subpass**
- VkFramebuffer **framebuffer**
- VkBool32 **occlusionQueryEnable**
- VkQueryControlFlags **queryFlags**
- VkQueryPipelineStatisticFlags **pipelineStatistics**

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

- lib/glfw/deps/glad/vulkan.h

26.249 VkCommandPoolCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkCommandPoolCreateFlags **flags**
- uint32_t **queueFamilyIndex**

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

- lib/glfw/deps/glad/vulkan.h

26.250 VkComponentMapping Struct Reference

Public Attributes

- VkComponentSwizzle **r**
- VkComponentSwizzle **g**
- VkComponentSwizzle **b**
- VkComponentSwizzle **a**

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

- lib/glfw/deps/glad/vulkan.h

26.251 VkComputePipelineCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkPipelineCreateFlags **flags**
- VkPipelineShaderStageCreateInfo **stage**
- VkPipelineLayout **layout**
- VkPipeline **basePipelineHandle**
- int32_t **basePipelineIndex**

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

- lib/glfw/deps/glad/vulkan.h

26.252 VkCopyDescriptorSet Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkDescriptorSet **srcSet**
- uint32_t **srcBinding**
- uint32_t **srcArrayElement**
- VkDescriptorSet **dstSet**
- uint32_t **dstBinding**
- uint32_t **dstArrayElement**
- uint32_t **descriptorCount**

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

- lib/glfw/deps/glad/vulkan.h

26.253 VkDebugReportCallbackCreateInfoEXT Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkDebugReportFlagsEXT **flags**
- PFN_vkDebugReportCallbackEXT **pfnCallback**
- void * **pUserData**

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

- lib/glfw/deps/glad/vulkan.h

26.254 VkDescriptorBufferInfo Struct Reference

Public Attributes

- VkBuffer **buffer**
- VkDeviceSize **offset**
- VkDeviceSize **range**

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

- lib/glfw/deps/glad/vulkan.h

26.255 VkDescriptorImageInfo Struct Reference

Public Attributes

- VkSampler **sampler**
- VkImageView **imageView**
- VkImageLayout **imageLayout**

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

- lib/glfw/deps/glad/vulkan.h

26.256 VkDescriptorPoolCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkDescriptorPoolCreateFlags **flags**
- uint32_t **maxSets**
- uint32_t **poolSizeCount**
- const VkDescriptorPoolSize * **pPoolSizes**

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

- lib/glfw/deps/glad/vulkan.h

26.257 VkDescriptorPoolSize Struct Reference

Public Attributes

- VkDescriptorType **type**
- uint32_t **descriptorCount**

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

- lib/glfw/deps/glad/vulkan.h

26.258 VkDescriptorSetAllocateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkDescriptorPool **descriptorPool**
- uint32_t **descriptorSetCount**
- const VkDescriptorSetLayout * **pSetLayouts**

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

- lib/glfw/deps/glad/vulkan.h

26.259 VkDescriptorSetLayoutBinding Struct Reference

Public Attributes

- uint32_t **binding**
- VkDescriptorType **descriptorType**
- uint32_t **descriptorCount**
- VkShaderStageFlags **stageFlags**
- const VkSampler * **pImmutableSamplers**

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

- lib/glfw/deps/glad/vulkan.h

26.260 VkDescriptorSetLayoutCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkDescriptorSetLayoutCreateFlags **flags**
- uint32_t **bindingCount**
- const VkDescriptorSetLayoutBinding * **pBindings**

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

- lib/glfw/deps/glad/vulkan.h

26.261 VkDescriptorSetLayoutSupport Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkBool32 **supported**

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

- lib/glfw/deps/glad/vulkan.h

26.262 VkDescriptorUpdateTemplateCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkDescriptorUpdateTemplateCreateFlags **flags**
- uint32_t **descriptorUpdateEntryCount**
- const VkDescriptorUpdateTemplateEntry * **pDescriptorUpdateEntries**
- VkDescriptorUpdateTemplateType **templateType**
- VkDescriptorSetLayout **descriptorSetLayout**
- VkPipelineBindPoint **pipelineBindPoint**
- VkPipelineLayout **pipelineLayout**
- uint32_t **set**

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

- lib/glfw/deps/glad/vulkan.h

26.263 VkDescriptorUpdateTemplateEntry Struct Reference

Public Attributes

- uint32_t **dstBinding**
- uint32_t **dstArrayElement**
- uint32_t **descriptorCount**
- VkDescriptorType **descriptorType**
- size_t **offset**
- size_t **stride**

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

- lib/glfw/deps/glad/vulkan.h

26.264 VkDeviceCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkDeviceCreateFlags **flags**
- uint32_t **queueCreateInfoCount**
- const VkDeviceQueueCreateInfo * **pQueueCreateInfos**
- uint32_t **enabledLayerCount**
- const char *const * **ppEnabledLayerNames**
- uint32_t **enabledExtensionCount**
- const char *const * **ppEnabledExtensionNames**
- const VkPhysicalDeviceFeatures * **pEnabledFeatures**

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

- lib/glfw/deps/glad/vulkan.h

26.265 VkDeviceGroupBindSparseInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void ∗ **pNext**
- uint32_t **resourceDeviceIndex**
- uint32_t **memoryDeviceIndex**

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

- lib/glfw/deps/glad/vulkan.h

26.266 VkDeviceGroupCommandBufferBeginInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void ∗ **pNext**
- uint32_t **deviceMask**

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

- lib/glfw/deps/glad/vulkan.h

26.267 VkDeviceGroupDeviceCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void ∗ **pNext**
- uint32_t **physicalDeviceCount**
- const VkPhysicalDevice ∗ **pPhysicalDevices**

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

- lib/glfw/deps/glad/vulkan.h

26.268 VkDeviceGroupPresentCapabilitiesKHR Struct Reference

Public Attributes

- VkStructureType **sType**
- const void ∗ **pNext**
- uint32_t **presentMask** [VK_MAX_DEVICE_GROUP_SIZE]
- VkDeviceGroupPresentModeFlagsKHR **modes**

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

- lib/glfw/deps/glad/vulkan.h

26.269 VkDeviceGroupPresentInfoKHR Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- uint32_t **swapchainCount**
- const uint32_t * **pDeviceMasks**
- VkDeviceGroupPresentModeFlagBitsKHR **mode**

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

- lib/glfw/deps/glad/vulkan.h

26.270 VkDeviceGroupRenderPassBeginInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- uint32_t **deviceMask**
- uint32_t **deviceRenderAreaCount**
- const VkRect2D * **pDeviceRenderAreas**

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

- lib/glfw/deps/glad/vulkan.h

26.271 VkDeviceGroupSubmitInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- uint32_t **waitSemaphoreCount**
- const uint32_t * **pWaitSemaphoreDeviceIndices**
- uint32_t **commandBufferCount**
- const uint32_t * **pCommandBufferDeviceMasks**
- uint32_t **signalSemaphoreCount**
- const uint32_t * **pSignalSemaphoreDeviceIndices**

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

- lib/glfw/deps/glad/vulkan.h

26.272 VkDeviceGroupSwapchainCreateInfoKHR Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkDeviceGroupPresentModeFlagsKHR **modes**

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

- lib/glfw/deps/glad/vulkan.h

26.273 VkDeviceQueueCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkDeviceQueueCreateFlags **flags**
- uint32_t **queueFamilyIndex**
- uint32_t **queueCount**
- const float * **pQueuePriorities**

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

- lib/glfw/deps/glad/vulkan.h

26.274 VkDeviceQueueInfo2 Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkDeviceQueueCreateFlags **flags**
- uint32_t **queueFamilyIndex**
- uint32_t **queueIndex**

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

- lib/glfw/deps/glad/vulkan.h

26.275 VkDispatchIndirectCommand Struct Reference

Public Attributes

- uint32_t **x**
- uint32_t **y**
- uint32_t **z**

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

- lib/glfw/deps/glad/vulkan.h

26.276 VkDrawIndexedIndirectCommand Struct Reference

Public Attributes

- uint32_t **indexCount**
- uint32_t **instanceCount**
- uint32_t **firstIndex**
- int32_t **vertexOffset**
- uint32_t **firstInstance**

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

- lib/glfw/deps/glad/vulkan.h

26.277 VkDrawIndirectCommand Struct Reference

Public Attributes

- uint32_t **vertexCount**
- uint32_t **instanceCount**
- uint32_t **firstVertex**
- uint32_t **firstInstance**

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

- lib/glfw/deps/glad/vulkan.h

26.278 VkEventCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkEventCreateFlags **flags**

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

- lib/glfw/deps/glad/vulkan.h

26.279 VkExportFenceCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void ∗ **pNext**
- VkExternalFenceHandleTypeFlags **handleTypes**

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

- lib/glfw/deps/glad/vulkan.h

26.280 VkExportMemoryAllocateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void ∗ **pNext**
- VkExternalMemoryHandleTypeFlags **handleTypes**

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

- lib/glfw/deps/glad/vulkan.h

26.281 VkExportSemaphoreCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void ∗ **pNext**
- VkExternalSemaphoreHandleTypeFlags **handleTypes**

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

- lib/glfw/deps/glad/vulkan.h

26.282 VkExtensionProperties Struct Reference

Public Attributes

- char **extensionName** [VK_MAX_EXTENSION_NAME_SIZE]
- uint32_t **specVersion**

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

- lib/glfw/deps/glad/vulkan.h
- lib/glfw/src/internal.h

26.283 VkExtent2D Struct Reference

Public Attributes

- uint32_t **width**
- uint32_t **height**

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

- lib/glfw/deps/glad/vulkan.h

26.284 VkExtent3D Struct Reference

Public Attributes

- uint32_t **width**
- uint32_t **height**
- uint32_t **depth**

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

- lib/glfw/deps/glad/vulkan.h

26.285 VkExternalBufferProperties Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkExternalMemoryProperties **externalMemoryProperties**

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

- lib/glfw/deps/glad/vulkan.h

26.286 VkExternalFenceProperties Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkExternalFenceHandleTypeFlags **exportFromImportedHandleTypes**
- VkExternalFenceHandleTypeFlags **compatibleHandleTypes**
- VkExternalFenceFeatureFlags **externalFenceFeatures**

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

- lib/glfw/deps/glad/vulkan.h

26.287 VkExternalImageFormatProperties Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkExternalMemoryProperties **externalMemoryProperties**

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

- lib/glfw/deps/glad/vulkan.h

26.288 VkExternalMemoryBufferCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkExternalMemoryHandleTypeFlags **handleTypes**

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

- lib/glfw/deps/glad/vulkan.h

26.289 VkExternalMemoryImageCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkExternalMemoryHandleTypeFlags **handleTypes**

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

- lib/glfw/deps/glad/vulkan.h

26.290 VkExternalMemoryProperties Struct Reference

Public Attributes

- VkExternalMemoryFeatureFlags **externalMemoryFeatures**
- VkExternalMemoryHandleTypeFlags **exportFromImportedHandleTypes**
- VkExternalMemoryHandleTypeFlags **compatibleHandleTypes**

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

- lib/glfw/deps/glad/vulkan.h

26.291 VkExternalSemaphoreProperties Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkExternalSemaphoreHandleTypeFlags **exportFromImportedHandleTypes**
- VkExternalSemaphoreHandleTypeFlags **compatibleHandleTypes**
- VkExternalSemaphoreFeatureFlags **externalSemaphoreFeatures**

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

- lib/glfw/deps/glad/vulkan.h

26.292 VkFenceCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkFenceCreateFlags **flags**

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

- lib/glfw/deps/glad/vulkan.h

26.293 VkFormatProperties Struct Reference

Public Attributes

- VkFormatFeatureFlags **linearTilingFeatures**
- VkFormatFeatureFlags **optimalTilingFeatures**
- VkFormatFeatureFlags **bufferFeatures**

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

- lib/glfw/deps/glad/vulkan.h

26.294 VkFormatProperties2 Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkFormatProperties **formatProperties**

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

- lib/glfw/deps/glad/vulkan.h

26.295 VkFramebufferCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkFramebufferCreateFlags **flags**
- VkRenderPass **renderPass**
- uint32_t **attachmentCount**
- const VkImageView * **pAttachments**
- uint32_t **width**
- uint32_t **height**
- uint32_t **layers**

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

- lib/glfw/deps/glad/vulkan.h

26.296 VkGraphicsPipelineCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkPipelineCreateFlags **flags**
- uint32_t **stageCount**
- const VkPipelineShaderStageCreateInfo * **pStages**
- const VkPipelineVertexInputStateCreateInfo * **pVertexInputState**
- const VkPipelineInputAssemblyStateCreateInfo * **pInputAssemblyState**
- const VkPipelineTessellationStateCreateInfo * **pTessellationState**
- const VkPipelineViewportStateCreateInfo * **pViewportState**
- const VkPipelineRasterizationStateCreateInfo * **pRasterizationState**
- const VkPipelineMultisampleStateCreateInfo * **pMultisampleState**
- const VkPipelineDepthStencilStateCreateInfo * **pDepthStencilState**
- const VkPipelineColorBlendStateCreateInfo * **pColorBlendState**
- const VkPipelineDynamicStateCreateInfo * **pDynamicState**
- VkPipelineLayout **layout**
- VkRenderPass **renderPass**
- uint32_t **subpass**
- VkPipeline **basePipelineHandle**
- int32_t **basePipelineIndex**

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

- lib/glfw/deps/glad/vulkan.h

26.297 VkImageBlit Struct Reference

Public Attributes

- VkImageSubresourceLayers **srcSubresource**
- VkOffset3D **srcOffsets** [2]
- VkImageSubresourceLayers **dstSubresource**
- VkOffset3D **dstOffsets** [2]

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

- lib/glfw/deps/glad/vulkan.h

26.298 VkImageCopy Struct Reference

Public Attributes

- VkImageSubresourceLayers **srcSubresource**
- VkOffset3D **srcOffset**
- VkImageSubresourceLayers **dstSubresource**
- VkOffset3D **dstOffset**
- VkExtent3D **extent**

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

- lib/glfw/deps/glad/vulkan.h

26.299 VkImageCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkImageCreateFlags **flags**
- VkImageType **imageType**
- VkFormat **format**
- VkExtent3D **extent**
- uint32_t **mipLevels**
- uint32_t **arrayLayers**
- VkSampleCountFlagBits **samples**
- VkImageTiling **tiling**
- VkImageUsageFlags **usage**
- VkSharingMode **sharingMode**
- uint32_t **queueFamilyIndexCount**
- const uint32_t * **pQueueFamilyIndices**
- VkImageLayout **initialLayout**

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

- lib/glfw/deps/glad/vulkan.h

26.300 VkImageFormatProperties Struct Reference

Public Attributes

- VkExtent3D **maxExtent**
- uint32_t **maxMipLevels**
- uint32_t **maxArrayLayers**
- VkSampleCountFlags **sampleCounts**
- VkDeviceSize **maxResourceSize**

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

- lib/glfw/deps/glad/vulkan.h

26.301 VkImageFormatProperties2 Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkImageFormatProperties **imageFormatProperties**

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

- lib/glfw/deps/glad/vulkan.h

26.302 VkImageMemoryBarrier Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkAccessFlags **srcAccessMask**
- VkAccessFlags **dstAccessMask**
- VkImageLayout **oldLayout**
- VkImageLayout **newLayout**
- uint32_t **srcQueueFamilyIndex**
- uint32_t **dstQueueFamilyIndex**
- VkImage **image**
- VkImageSubresourceRange **subresourceRange**

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

- lib/glfw/deps/glad/vulkan.h

26.303 VkImageMemoryRequirementsInfo2 Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkImage **image**

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

- lib/glfw/deps/glad/vulkan.h

26.304 VkImagePlaneMemoryRequirementsInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkImageAspectFlagBits **planeAspect**

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

- lib/glfw/deps/glad/vulkan.h

26.305 VkImageResolve Struct Reference

Public Attributes

- VkImageSubresourceLayers **srcSubresource**
- VkOffset3D **srcOffset**
- VkImageSubresourceLayers **dstSubresource**
- VkOffset3D **dstOffset**
- VkExtent3D **extent**

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

- lib/glfw/deps/glad/vulkan.h

26.306 VkImageSparseMemoryRequirementsInfo2 Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkImage **image**

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

- lib/glfw/deps/glad/vulkan.h

26.307 VkImageSubresource Struct Reference

Public Attributes

- VkImageAspectFlags **aspectMask**
- uint32_t **mipLevel**
- uint32_t **arrayLayer**

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

- lib/glfw/deps/glad/vulkan.h

26.308 VkImageSubresourceLayers Struct Reference

Public Attributes

- VkImageAspectFlags **aspectMask**
- uint32_t **mipLevel**
- uint32_t **baseArrayLayer**
- uint32_t **layerCount**

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

- lib/glfw/deps/glad/vulkan.h

26.309 VkImageSubresourceRange Struct Reference

Public Attributes

- VkImageAspectFlags **aspectMask**
- uint32_t **baseMipLevel**
- uint32_t **levelCount**
- uint32_t **baseArrayLayer**
- uint32_t **layerCount**

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

- lib/glfw/deps/glad/vulkan.h

26.310 VkImageSwapchainCreateInfoKHR Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkSwapchainKHR **swapchain**

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

- lib/glfw/deps/glad/vulkan.h

26.311 VkImageViewCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkImageViewCreateFlags **flags**
- VkImage **image**
- VkImageViewType **viewType**
- VkFormat **format**
- VkComponentMapping **components**
- VkImageSubresourceRange **subresourceRange**

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

- lib/glfw/deps/glad/vulkan.h

26.312 VkImageViewUsageCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkImageUsageFlags **usage**

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

- lib/glfw/deps/glad/vulkan.h

26.313 VkInputAttachmentAspectReference Struct Reference

Public Attributes

- uint32_t **subpass**
- uint32_t **inputAttachmentIndex**
- VkImageAspectFlags **aspectMask**

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

- lib/glfw/deps/glad/vulkan.h

26.314 VkInstanceCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkInstanceCreateFlags **flags**
- const VkApplicationInfo * **pApplicationInfo**
- uint32_t **enabledLayerCount**
- const char *const * **ppEnabledLayerNames**
- uint32_t **enabledExtensionCount**
- const char *const * **ppEnabledExtensionNames**

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

- lib/glfw/deps/glad/vulkan.h

26.315 VkLayerProperties Struct Reference

Public Attributes

- char **layerName** [VK_MAX_EXTENSION_NAME_SIZE]
- uint32_t **specVersion**
- uint32_t **implementationVersion**
- char **description** [VK_MAX_DESCRIPTION_SIZE]

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

- lib/glfw/deps/glad/vulkan.h

26.316 VkMacOSSurfaceCreateInfoMVK Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkMacOSSurfaceCreateFlagsMVK **flags**
- const void * **pView**

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

- lib/glfw/src/cocoa_platform.h

26.317 VkMappedMemoryRange Struct Reference

Public Attributes

- VkStructureType **sType**
- const void ∗ **pNext**
- VkDeviceMemory **memory**
- VkDeviceSize **offset**
- VkDeviceSize **size**

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

- lib/glfw/deps/glad/vulkan.h

26.318 VkMemoryAllocateFlagsInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void ∗ **pNext**
- VkMemoryAllocateFlags **flags**
- uint32_t **deviceMask**

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

- lib/glfw/deps/glad/vulkan.h

26.319 VkMemoryAllocateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void ∗ **pNext**
- VkDeviceSize **allocationSize**
- uint32_t **memoryTypeIndex**

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

- lib/glfw/deps/glad/vulkan.h

26.320 VkMemoryBarrier Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkAccessFlags **srcAccessMask**
- VkAccessFlags **dstAccessMask**

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

- lib/glfw/deps/glad/vulkan.h

26.321 VkMemoryDedicatedAllocateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkImage **image**
- VkBuffer **buffer**

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

- lib/glfw/deps/glad/vulkan.h

26.322 VkMemoryDedicatedRequirements Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkBool32 **prefersDedicatedAllocation**
- VkBool32 **requiresDedicatedAllocation**

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

- lib/glfw/deps/glad/vulkan.h

26.323 VkMemoryHeap Struct Reference

Public Attributes

- VkDeviceSize **size**
- VkMemoryHeapFlags **flags**

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

- lib/glfw/deps/glad/vulkan.h

26.324 VkMemoryRequirements Struct Reference

Public Attributes

- VkDeviceSize **size**
- VkDeviceSize **alignment**
- uint32_t **memoryTypeBits**

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

- lib/glfw/deps/glad/vulkan.h

26.325 VkMemoryRequirements2 Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkMemoryRequirements **memoryRequirements**

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

- lib/glfw/deps/glad/vulkan.h

26.326 VkMemoryType Struct Reference

Public Attributes

- VkMemoryPropertyFlags **propertyFlags**
- uint32_t **heapIndex**

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

- lib/glfw/deps/glad/vulkan.h

26.327 VkMetalSurfaceCreateInfoEXT Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkMetalSurfaceCreateFlagsEXT **flags**
- const void * **pLayer**

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

- lib/glfw/src/cocoa_platform.h

26.328 VkOffset2D Struct Reference

Public Attributes

- int32_t **x**
- int32_t **y**

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

- lib/glfw/deps/glad/vulkan.h

26.329 VkOffset3D Struct Reference

Public Attributes

- int32_t **x**
- int32_t **y**
- int32_t **z**

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

- lib/glfw/deps/glad/vulkan.h

26.330 VkPhysicalDevice16BitStorageFeatures Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkBool32 **storageBuffer16BitAccess**
- VkBool32 **uniformAndStorageBuffer16BitAccess**
- VkBool32 **storagePushConstant16**
- VkBool32 **storageInputOutput16**

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

- lib/glfw/deps/glad/vulkan.h

26.331 VkPhysicalDeviceExternalBufferInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkBufferCreateFlags **flags**
- VkBufferUsageFlags **usage**
- VkExternalMemoryHandleTypeFlagBits **handleType**

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

- lib/glfw/deps/glad/vulkan.h

26.332 VkPhysicalDeviceExternalFenceInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkExternalFenceHandleTypeFlagBits **handleType**

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

- lib/glfw/deps/glad/vulkan.h

26.333 VkPhysicalDeviceExternalImageFormatInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkExternalMemoryHandleTypeFlagBits **handleType**

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

- lib/glfw/deps/glad/vulkan.h

26.334 VkPhysicalDeviceExternalSemaphoreInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkExternalSemaphoreHandleTypeFlagBits **handleType**

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

- lib/glfw/deps/glad/vulkan.h

26.335 VkPhysicalDeviceFeatures Struct Reference

Public Attributes

- VkBool32 **robustBufferAccess**
- VkBool32 **fullDrawIndexUint32**
- VkBool32 **imageCubeArray**
- VkBool32 **independentBlend**
- VkBool32 **geometryShader**
- VkBool32 **tessellationShader**
- VkBool32 **sampleRateShading**
- VkBool32 **dualSrcBlend**
- VkBool32 **logicOp**
- VkBool32 **multiDrawIndirect**
- VkBool32 **drawIndirectFirstInstance**
- VkBool32 **depthClamp**
- VkBool32 **depthBiasClamp**
- VkBool32 **fillModeNonSolid**
- VkBool32 **depthBounds**
- VkBool32 **wideLines**
- VkBool32 **largePoints**
- VkBool32 **alphaToOne**
- VkBool32 **multiViewport**
- VkBool32 **samplerAnisotropy**
- VkBool32 **textureCompressionETC2**
- VkBool32 **textureCompressionASTC_LDR**
- VkBool32 **textureCompressionBC**
- VkBool32 **occlusionQueryPrecise**
- VkBool32 **pipelineStatisticsQuery**
- VkBool32 **vertexPipelineStoresAndAtomics**
- VkBool32 **fragmentStoresAndAtomics**
- VkBool32 **shaderTessellationAndGeometryPointSize**
- VkBool32 **shaderImageGatherExtended**
- VkBool32 **shaderStorageImageExtendedFormats**
- VkBool32 **shaderStorageImageMultisample**
- VkBool32 **shaderStorageImageReadWithoutFormat**
- VkBool32 **shaderStorageImageWriteWithoutFormat**
- VkBool32 **shaderUniformBufferArrayDynamicIndexing**
- VkBool32 **shaderSampledImageArrayDynamicIndexing**
- VkBool32 **shaderStorageBufferArrayDynamicIndexing**
- VkBool32 **shaderStorageImageArrayDynamicIndexing**
- VkBool32 **shaderClipDistance**
- VkBool32 **shaderCullDistance**
- VkBool32 **shaderFloat64**
- VkBool32 **shaderInt64**
- VkBool32 **shaderInt16**
- VkBool32 **shaderResourceResidency**
- VkBool32 **shaderResourceMinLod**
- VkBool32 **sparseBinding**
- VkBool32 **sparseResidencyBuffer**
- VkBool32 **sparseResidencyImage2D**
- VkBool32 **sparseResidencyImage3D**
- VkBool32 **sparseResidency2Samples**
- VkBool32 **sparseResidency4Samples**

- VkBool32 **sparseResidency8Samples**
- VkBool32 **sparseResidency16Samples**
- VkBool32 **sparseResidencyAliased**
- VkBool32 **variableMultisampleRate**
- VkBool32 **inheritedQueries**

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

- lib/glfw/deps/glad/vulkan.h

26.336 VkPhysicalDeviceFeatures2 Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkPhysicalDeviceFeatures **features**

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

- lib/glfw/deps/glad/vulkan.h

26.337 VkPhysicalDeviceGroupProperties Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- uint32_t **physicalDeviceCount**
- VkPhysicalDevice **physicalDevices** [VK_MAX_DEVICE_GROUP_SIZE]
- VkBool32 **subsetAllocation**

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

- lib/glfw/deps/glad/vulkan.h

26.338 VkPhysicalDeviceIDProperties Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- uint8_t **deviceUUID** [VK_UUID_SIZE]
- uint8_t **driverUUID** [VK_UUID_SIZE]
- uint8_t **deviceLUID** [VK_LUID_SIZE]
- uint32_t **deviceNodeMask**
- VkBool32 **deviceLUIDValid**

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

- lib/glfw/deps/glad/vulkan.h

26.339 VkPhysicalDeviceImageFormatInfo2 Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkFormat **format**
- VkImageType **type**
- VkImageTiling **tiling**
- VkImageUsageFlags **usage**
- VkImageCreateFlags **flags**

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

- lib/glfw/deps/glad/vulkan.h

26.340 VkPhysicalDeviceLimits Struct Reference

Public Attributes

- uint32_t **maxImageDimension1D**
- uint32_t **maxImageDimension2D**
- uint32_t **maxImageDimension3D**
- uint32_t **maxImageDimensionCube**
- uint32_t **maxImageArrayLayers**
- uint32_t **maxTexelBufferElements**
- uint32_t **maxUniformBufferRange**
- uint32_t **maxStorageBufferRange**
- uint32_t **maxPushConstantsSize**
- uint32_t **maxMemoryAllocationCount**
- uint32_t **maxSamplerAllocationCount**
- VkDeviceSize **bufferImageGranularity**
- VkDeviceSize **sparseAddressSpaceSize**
- uint32_t **maxBoundDescriptorSets**
- uint32_t **maxPerStageDescriptorSamplers**
- uint32_t **maxPerStageDescriptorUniformBuffers**
- uint32_t **maxPerStageDescriptorStorageBuffers**
- uint32_t **maxPerStageDescriptorSampledImages**
- uint32_t **maxPerStageDescriptorStorageImages**
- uint32_t **maxPerStageDescriptorInputAttachments**
- uint32_t **maxPerStageResources**
- uint32_t **maxDescriptorSetSamplers**
- uint32_t **maxDescriptorSetUniformBuffers**
- uint32_t **maxDescriptorSetUniformBuffersDynamic**
- uint32_t **maxDescriptorSetStorageBuffers**
- uint32_t **maxDescriptorSetStorageBuffersDynamic**
- uint32_t **maxDescriptorSetSampledImages**
- uint32_t **maxDescriptorSetStorageImages**
- uint32_t **maxDescriptorSetInputAttachments**
- uint32_t **maxVertexInputAttributes**
- uint32_t **maxVertexInputBindings**

- uint32_t **maxVertexInputAttributeOffset**
- uint32_t **maxVertexInputBindingStride**
- uint32_t **maxVertexOutputComponents**
- uint32_t **maxTessellationGenerationLevel**
- uint32_t **maxTessellationPatchSize**
- uint32_t **maxTessellationControlPerVertexInputComponents**
- uint32_t **maxTessellationControlPerVertexOutputComponents**
- uint32_t **maxTessellationControlPerPatchOutputComponents**
- uint32_t **maxTessellationControlTotalOutputComponents**
- uint32_t **maxTessellationEvaluationInputComponents**
- uint32_t **maxTessellationEvaluationOutputComponents**
- uint32_t **maxGeometryShaderInvocations**
- uint32_t **maxGeometryInputComponents**
- uint32_t **maxGeometryOutputComponents**
- uint32_t **maxGeometryOutputVertices**
- uint32_t **maxGeometryTotalOutputComponents**
- uint32_t **maxFragmentInputComponents**
- uint32_t **maxFragmentOutputAttachments**
- uint32_t **maxFragmentDualSrcAttachments**
- uint32_t **maxFragmentCombinedOutputResources**
- uint32_t **maxComputeSharedMemorySize**
- uint32_t **maxComputeWorkGroupCount** [3]
- uint32_t **maxComputeWorkGroupInvocations**
- uint32_t **maxComputeWorkGroupSize** [3]
- uint32_t **subPixelPrecisionBits**
- uint32_t **subTexelPrecisionBits**
- uint32_t **mipmapPrecisionBits**
- uint32_t **maxDrawIndexedIndexValue**
- uint32_t **maxDrawIndirectCount**
- float **maxSamplerLodBias**
- float **maxSamplerAnisotropy**
- uint32_t **maxViewports**
- uint32_t **maxViewportDimensions** [2]
- float **viewportBoundsRange** [2]
- uint32_t **viewportSubPixelBits**
- size_t **minMemoryMapAlignment**
- VkDeviceSize **minTexelBufferOffsetAlignment**
- VkDeviceSize **minUniformBufferOffsetAlignment**
- VkDeviceSize **minStorageBufferOffsetAlignment**
- int32_t **minTexelOffset**
- uint32_t **maxTexelOffset**
- int32_t **minTexelGatherOffset**
- uint32_t **maxTexelGatherOffset**
- float **minInterpolationOffset**
- float **maxInterpolationOffset**
- uint32_t **subPixelInterpolationOffsetBits**
- uint32_t **maxFramebufferWidth**
- uint32_t **maxFramebufferHeight**
- uint32_t **maxFramebufferLayers**
- VkSampleCountFlags **framebufferColorSampleCounts**
- VkSampleCountFlags **framebufferDepthSampleCounts**
- VkSampleCountFlags **framebufferStencilSampleCounts**
- VkSampleCountFlags **framebufferNoAttachmentsSampleCounts**
- uint32_t **maxColorAttachments**
- VkSampleCountFlags **sampledImageColorSampleCounts**

- VkSampleCountFlags **sampledImageIntegerSampleCounts**
- VkSampleCountFlags **sampledImageDepthSampleCounts**
- VkSampleCountFlags **sampledImageStencilSampleCounts**
- VkSampleCountFlags **storageImageSampleCounts**
- uint32_t **maxSampleMaskWords**
- VkBool32 **timestampComputeAndGraphics**
- float **timestampPeriod**
- uint32_t **maxClipDistances**
- uint32_t **maxCullDistances**
- uint32_t **maxCombinedClipAndCullDistances**
- uint32_t **discreteQueuePriorities**
- float **pointSizeRange** [2]
- float **lineWidthRange** [2]
- float **pointSizeGranularity**
- float **lineWidthGranularity**
- VkBool32 **strictLines**
- VkBool32 **standardSampleLocations**
- VkDeviceSize **optimalBufferCopyOffsetAlignment**
- VkDeviceSize **optimalBufferCopyRowPitchAlignment**
- VkDeviceSize **nonCoherentAtomSize**

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

- lib/glfw/deps/glad/vulkan.h

26.341 VkPhysicalDeviceMaintenance3Properties Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- uint32_t **maxPerSetDescriptors**
- VkDeviceSize **maxMemoryAllocationSize**

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

- lib/glfw/deps/glad/vulkan.h

26.342 VkPhysicalDeviceMemoryProperties Struct Reference

Public Attributes

- uint32_t **memoryTypeCount**
- VkMemoryType **memoryTypes** [VK_MAX_MEMORY_TYPES]
- uint32_t **memoryHeapCount**
- VkMemoryHeap **memoryHeaps** [VK_MAX_MEMORY_HEAPS]

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

- lib/glfw/deps/glad/vulkan.h

26.343 VkPhysicalDeviceMemoryProperties2 Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkPhysicalDeviceMemoryProperties **memoryProperties**

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

- lib/glfw/deps/glad/vulkan.h

26.344 VkPhysicalDeviceMultiviewFeatures Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkBool32 **multiview**
- VkBool32 **multiviewGeometryShader**
- VkBool32 **multiviewTessellationShader**

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

- lib/glfw/deps/glad/vulkan.h

26.345 VkPhysicalDeviceMultiviewProperties Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- uint32_t **maxMultiviewViewCount**
- uint32_t **maxMultiviewInstanceIndex**

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

- lib/glfw/deps/glad/vulkan.h

26.346 VkPhysicalDevicePointClippingProperties Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkPointClippingBehavior **pointClippingBehavior**

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

- lib/glfw/deps/glad/vulkan.h

26.347 VkPhysicalDeviceProperties Struct Reference

Public Attributes

- uint32_t **apiVersion**
- uint32_t **driverVersion**
- uint32_t **vendorID**
- uint32_t **deviceID**
- VkPhysicalDeviceType **deviceType**
- char **deviceName** [VK_MAX_PHYSICAL_DEVICE_NAME_SIZE]
- uint8_t **pipelineCacheUUID** [VK_UUID_SIZE]
- VkPhysicalDeviceLimits **limits**
- VkPhysicalDeviceSparseProperties **sparseProperties**

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

- lib/glfw/deps/glad/vulkan.h

26.348 VkPhysicalDeviceProperties2 Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkPhysicalDeviceProperties **properties**

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

- lib/glfw/deps/glad/vulkan.h

26.349 VkPhysicalDeviceProtectedMemoryFeatures Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkBool32 **protectedMemory**

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

- lib/glfw/deps/glad/vulkan.h

26.350 VkPhysicalDeviceProtectedMemoryProperties Struct Reference

Public Attributes

- VkStructureType **sType**
- void ∗ **pNext**
- VkBool32 **protectedNoFault**

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

- lib/glfw/deps/glad/vulkan.h

26.351 VkPhysicalDeviceSamplerYcbcrConversionFeatures Struct Reference

Public Attributes

- VkStructureType **sType**
- void ∗ **pNext**
- VkBool32 **samplerYcbcrConversion**

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

- lib/glfw/deps/glad/vulkan.h

26.352 VkPhysicalDeviceShaderDrawParametersFeatures Struct Reference

Public Attributes

- VkStructureType **sType**
- void ∗ **pNext**
- VkBool32 **shaderDrawParameters**

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

- lib/glfw/deps/glad/vulkan.h

26.353 VkPhysicalDeviceSparseImageFormatInfo2 Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkFormat **format**
- VkImageType **type**
- VkSampleCountFlagBits **samples**
- VkImageUsageFlags **usage**
- VkImageTiling **tiling**

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

- lib/glfw/deps/glad/vulkan.h

26.354 VkPhysicalDeviceSparseProperties Struct Reference

Public Attributes

- VkBool32 **residencyStandard2DBlockShape**
- VkBool32 **residencyStandard2DMultisampleBlockShape**
- VkBool32 **residencyStandard3DBlockShape**
- VkBool32 **residencyAlignedMipSize**
- VkBool32 **residencyNonResidentStrict**

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

- lib/glfw/deps/glad/vulkan.h

26.355 VkPhysicalDeviceSubgroupProperties Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- uint32_t **subgroupSize**
- VkShaderStageFlags **supportedStages**
- VkSubgroupFeatureFlags **supportedOperations**
- VkBool32 **quadOperationsInAllStages**

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

- lib/glfw/deps/glad/vulkan.h

26.356 VkPhysicalDeviceVariablePointersFeatures Struct Reference

Public Attributes

- VkStructureType **sType**
- void ∗ **pNext**
- VkBool32 **variablePointersStorageBuffer**
- VkBool32 **variablePointers**

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

- lib/glfw/deps/glad/vulkan.h

26.357 VkPipelineCacheCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void ∗ **pNext**
- VkPipelineCacheCreateFlags **flags**
- size_t **initialDataSize**
- const void ∗ **pInitialData**

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

- lib/glfw/deps/glad/vulkan.h

26.358 VkPipelineColorBlendAttachmentState Struct Reference

Public Attributes

- VkBool32 **blendEnable**
- VkBlendFactor **srcColorBlendFactor**
- VkBlendFactor **dstColorBlendFactor**
- VkBlendOp **colorBlendOp**
- VkBlendFactor **srcAlphaBlendFactor**
- VkBlendFactor **dstAlphaBlendFactor**
- VkBlendOp **alphaBlendOp**
- VkColorComponentFlags **colorWriteMask**

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

- lib/glfw/deps/glad/vulkan.h

26.359 VkPipelineColorBlendStateCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkPipelineColorBlendStateCreateFlags **flags**
- VkBool32 **logicOpEnable**
- VkLogicOp **logicOp**
- uint32_t **attachmentCount**
- const VkPipelineColorBlendAttachmentState * **pAttachments**
- float **blendConstants** [4]

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

- lib/glfw/deps/glad/vulkan.h

26.360 VkPipelineDepthStencilStateCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkPipelineDepthStencilStateCreateFlags **flags**
- VkBool32 **depthTestEnable**
- VkBool32 **depthWriteEnable**
- VkCompareOp **depthCompareOp**
- VkBool32 **depthBoundsTestEnable**
- VkBool32 **stencilTestEnable**
- VkStencilOpState **front**
- VkStencilOpState **back**
- float **minDepthBounds**
- float **maxDepthBounds**

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

- lib/glfw/deps/glad/vulkan.h

26.361 VkPipelineDynamicStateCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkPipelineDynamicStateCreateFlags **flags**
- uint32_t **dynamicStateCount**
- const VkDynamicState * **pDynamicStates**

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

- lib/glfw/deps/glad/vulkan.h

26.362 VkPipelineInputAssemblyStateCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkPipelineInputAssemblyStateCreateFlags **flags**
- VkPrimitiveTopology **topology**
- VkBool32 **primitiveRestartEnable**

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

- lib/glfw/deps/glad/vulkan.h

26.363 VkPipelineLayoutCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkPipelineLayoutCreateFlags **flags**
- uint32_t **setLayoutCount**
- const VkDescriptorSetLayout * **pSetLayouts**
- uint32_t **pushConstantRangeCount**
- const VkPushConstantRange * **pPushConstantRanges**

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

- lib/glfw/deps/glad/vulkan.h

26.364 VkPipelineMultisampleStateCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkPipelineMultisampleStateCreateFlags **flags**
- VkSampleCountFlagBits **rasterizationSamples**
- VkBool32 **sampleShadingEnable**
- float **minSampleShading**
- const VkSampleMask * **pSampleMask**
- VkBool32 **alphaToCoverageEnable**
- VkBool32 **alphaToOneEnable**

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

- lib/glfw/deps/glad/vulkan.h

26.365 VkPipelineRasterizationStateCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkPipelineRasterizationStateCreateFlags **flags**
- VkBool32 **depthClampEnable**
- VkBool32 **rasterizerDiscardEnable**
- VkPolygonMode **polygonMode**
- VkCullModeFlags **cullMode**
- VkFrontFace **frontFace**
- VkBool32 **depthBiasEnable**
- float **depthBiasConstantFactor**
- float **depthBiasClamp**
- float **depthBiasSlopeFactor**
- float **lineWidth**

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

- lib/glfw/deps/glad/vulkan.h

26.366 VkPipelineShaderStageCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkPipelineShaderStageCreateFlags **flags**
- VkShaderStageFlagBits **stage**
- VkShaderModule **module**
- const char * **pName**
- const VkSpecializationInfo * **pSpecializationInfo**

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

- lib/glfw/deps/glad/vulkan.h

26.367 VkPipelineTessellationDomainOriginStateCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkTessellationDomainOrigin **domainOrigin**

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

- lib/glfw/deps/glad/vulkan.h

26.368 VkPipelineTessellationStateCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkPipelineTessellationStateCreateFlags **flags**
- uint32_t **patchControlPoints**

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

- lib/glfw/deps/glad/vulkan.h

26.369 VkPipelineVertexInputStateCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkPipelineVertexInputStateCreateFlags **flags**
- uint32_t **vertexBindingDescriptionCount**
- const VkVertexInputBindingDescription * **pVertexBindingDescriptions**
- uint32_t **vertexAttributeDescriptionCount**
- const VkVertexInputAttributeDescription * **pVertexAttributeDescriptions**

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

- lib/glfw/deps/glad/vulkan.h

26.370 VkPipelineViewportStateCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkPipelineViewportStateCreateFlags **flags**
- uint32_t **viewportCount**
- const VkViewport * **pViewports**
- uint32_t **scissorCount**
- const VkRect2D * **pScissors**

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

- lib/glfw/deps/glad/vulkan.h

26.371 VkPresentInfoKHR Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- uint32_t **waitSemaphoreCount**
- const VkSemaphore * **pWaitSemaphores**
- uint32_t **swapchainCount**
- const VkSwapchainKHR * **pSwapchains**
- const uint32_t * **pImageIndices**
- VkResult * **pResults**

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

- lib/glfw/deps/glad/vulkan.h

26.372 VkProtectedSubmitInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkBool32 **protectedSubmit**

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

- lib/glfw/deps/glad/vulkan.h

26.373 VkPushConstantRange Struct Reference

Public Attributes

- VkShaderStageFlags **stageFlags**
- uint32_t **offset**
- uint32_t **size**

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

- lib/glfw/deps/glad/vulkan.h

26.374 VkQueryPoolCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkQueryPoolCreateFlags **flags**
- VkQueryType **queryType**
- uint32_t **queryCount**
- VkQueryPipelineStatisticFlags **pipelineStatistics**

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

- lib/glfw/deps/glad/vulkan.h

26.375 VkQueueFamilyProperties Struct Reference

Public Attributes

- VkQueueFlags **queueFlags**
- uint32_t **queueCount**
- uint32_t **timestampValidBits**
- VkExtent3D **minImageTransferGranularity**

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

- lib/glfw/deps/glad/vulkan.h

26.376 VkQueueFamilyProperties2 Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkQueueFamilyProperties **queueFamilyProperties**

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

- lib/glfw/deps/glad/vulkan.h

26.377 VkRect2D Struct Reference

Public Attributes

- VkOffset2D **offset**
- VkExtent2D **extent**

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

- lib/glfw/deps/glad/vulkan.h

26.378 VkRenderPassBeginInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkRenderPass **renderPass**
- VkFramebuffer **framebuffer**
- VkRect2D **renderArea**
- uint32_t **clearValueCount**
- const VkClearValue * **pClearValues**

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

- lib/glfw/deps/glad/vulkan.h

26.379 VkRenderPassCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkRenderPassCreateFlags **flags**
- uint32_t **attachmentCount**
- const VkAttachmentDescription * **pAttachments**
- uint32_t **subpassCount**
- const VkSubpassDescription * **pSubpasses**
- uint32_t **dependencyCount**
- const VkSubpassDependency * **pDependencies**

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

- lib/glfw/deps/glad/vulkan.h

26.380 VkRenderPassInputAttachmentAspectCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- uint32_t **aspectReferenceCount**
- const VkInputAttachmentAspectReference * **pAspectReferences**

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

- lib/glfw/deps/glad/vulkan.h

26.381 VkRenderPassMultiviewCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- uint32_t **subpassCount**
- const uint32_t * **pViewMasks**
- uint32_t **dependencyCount**
- const int32_t * **pViewOffsets**
- uint32_t **correlationMaskCount**
- const uint32_t * **pCorrelationMasks**

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

- lib/glfw/deps/glad/vulkan.h

26.382 VkSamplerCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkSamplerCreateFlags **flags**
- VkFilter **magFilter**
- VkFilter **minFilter**
- VkSamplerMipmapMode **mipmapMode**
- VkSamplerAddressMode **addressModeU**
- VkSamplerAddressMode **addressModeV**
- VkSamplerAddressMode **addressModeW**
- float **mipLodBias**
- VkBool32 **anisotropyEnable**
- float **maxAnisotropy**
- VkBool32 **compareEnable**
- VkCompareOp **compareOp**
- float **minLod**
- float **maxLod**
- VkBorderColor **borderColor**
- VkBool32 **unnormalizedCoordinates**

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

- lib/glfw/deps/glad/vulkan.h

26.383 VkSamplerYcbcrConversionCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkFormat **format**
- VkSamplerYcbcrModelConversion **ycbcrModel**
- VkSamplerYcbcrRange **ycbcrRange**
- VkComponentMapping **components**
- VkChromaLocation **xChromaOffset**
- VkChromaLocation **yChromaOffset**
- VkFilter **chromaFilter**
- VkBool32 **forceExplicitReconstruction**

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

- lib/glfw/deps/glad/vulkan.h

26.384 VkSamplerYcbcrConversionImageFormatProperties Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- uint32_t **combinedImageSamplerDescriptorCount**

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

- lib/glfw/deps/glad/vulkan.h

26.385 VkSamplerYcbcrConversionInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkSamplerYcbcrConversion **conversion**

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

- lib/glfw/deps/glad/vulkan.h

26.386 VkSemaphoreCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkSemaphoreCreateFlags **flags**

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

- lib/glfw/deps/glad/vulkan.h

26.387 VkShaderModuleCreateInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkShaderModuleCreateFlags **flags**
- size_t **codeSize**
- const uint32_t * **pCode**

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

- lib/glfw/deps/glad/vulkan.h

26.388 VkSparseBufferMemoryBindInfo Struct Reference

Public Attributes

- VkBuffer **buffer**
- uint32_t **bindCount**
- const VkSparseMemoryBind * **pBinds**

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

- lib/glfw/deps/glad/vulkan.h

26.389 VkSparseImageFormatProperties Struct Reference

Public Attributes

- VkImageAspectFlags **aspectMask**
- VkExtent3D **imageGranularity**
- VkSparseImageFormatFlags **flags**

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

- lib/glfw/deps/glad/vulkan.h

26.390 VkSparseImageFormatProperties2 Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkSparseImageFormatProperties **properties**

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

- lib/glfw/deps/glad/vulkan.h

26.391 VkSparseImageMemoryBind Struct Reference

Public Attributes

- VkImageSubresource **subresource**
- VkOffset3D **offset**
- VkExtent3D **extent**
- VkDeviceMemory **memory**
- VkDeviceSize **memoryOffset**
- VkSparseMemoryBindFlags **flags**

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

- lib/glfw/deps/glad/vulkan.h

26.392 VkSparseImageMemoryBindInfo Struct Reference

Public Attributes

- VkImage **image**
- uint32_t **bindCount**
- const VkSparseImageMemoryBind * **pBinds**

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

- lib/glfw/deps/glad/vulkan.h

26.393 VkSparseImageMemoryRequirements Struct Reference

Public Attributes

- VkSparseImageFormatProperties **formatProperties**
- uint32_t **imageMipTailFirstLod**
- VkDeviceSize **imageMipTailSize**
- VkDeviceSize **imageMipTailOffset**
- VkDeviceSize **imageMipTailStride**

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

- lib/glfw/deps/glad/vulkan.h

26.394 VkSparseImageMemoryRequirements2 Struct Reference

Public Attributes

- VkStructureType **sType**
- void * **pNext**
- VkSparseImageMemoryRequirements **memoryRequirements**

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

- lib/glfw/deps/glad/vulkan.h

26.395 VkSparseImageOpaqueMemoryBindInfo Struct Reference

Public Attributes

- VkImage **image**
- uint32_t **bindCount**
- const VkSparseMemoryBind * **pBinds**

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

- lib/glfw/deps/glad/vulkan.h

26.396 VkSparseMemoryBind Struct Reference

Public Attributes

- VkDeviceSize **resourceOffset**
- VkDeviceSize **size**
- VkDeviceMemory **memory**
- VkDeviceSize **memoryOffset**
- VkSparseMemoryBindFlags **flags**

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

- lib/glfw/deps/glad/vulkan.h

26.397 VkSpecializationInfo Struct Reference

Public Attributes

- uint32_t **mapEntryCount**
- const VkSpecializationMapEntry * **pMapEntries**
- size_t **dataSize**
- const void * **pData**

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

- lib/glfw/deps/glad/vulkan.h

26.398 VkSpecializationMapEntry Struct Reference

Public Attributes

- uint32_t **constantID**
- uint32_t **offset**
- size_t **size**

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

- lib/glfw/deps/glad/vulkan.h

26.399 VkStencilOpState Struct Reference

Public Attributes

- VkStencilOp **failOp**
- VkStencilOp **passOp**
- VkStencilOp **depthFailOp**
- VkCompareOp **compareOp**
- uint32_t **compareMask**
- uint32_t **writeMask**
- uint32_t **reference**

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

- lib/glfw/deps/glad/vulkan.h

26.400 VkSubmitInfo Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- uint32_t **waitSemaphoreCount**
- const VkSemaphore * **pWaitSemaphores**
- const VkPipelineStageFlags * **pWaitDstStageMask**
- uint32_t **commandBufferCount**
- const VkCommandBuffer * **pCommandBuffers**
- uint32_t **signalSemaphoreCount**
- const VkSemaphore * **pSignalSemaphores**

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

- lib/glfw/deps/glad/vulkan.h

26.401 VkSubpassDependency Struct Reference

Public Attributes

- uint32_t **srcSubpass**
- uint32_t **dstSubpass**
- VkPipelineStageFlags **srcStageMask**
- VkPipelineStageFlags **dstStageMask**
- VkAccessFlags **srcAccessMask**
- VkAccessFlags **dstAccessMask**
- VkDependencyFlags **dependencyFlags**

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

- lib/glfw/deps/glad/vulkan.h

26.402 VkSubpassDescription Struct Reference

Public Attributes

- VkSubpassDescriptionFlags **flags**
- VkPipelineBindPoint **pipelineBindPoint**
- uint32_t **inputAttachmentCount**
- const VkAttachmentReference * **pInputAttachments**
- uint32_t **colorAttachmentCount**
- const VkAttachmentReference * **pColorAttachments**
- const VkAttachmentReference * **pResolveAttachments**
- const VkAttachmentReference * **pDepthStencilAttachment**
- uint32_t **preserveAttachmentCount**
- const uint32_t * **pPreserveAttachments**

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

- lib/glfw/deps/glad/vulkan.h

26.403 VkSubresourceLayout Struct Reference

Public Attributes

- VkDeviceSize **offset**
- VkDeviceSize **size**
- VkDeviceSize **rowPitch**
- VkDeviceSize **arrayPitch**
- VkDeviceSize **depthPitch**

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

- lib/glfw/deps/glad/vulkan.h

26.404 VkSurfaceCapabilitiesKHR Struct Reference

Public Attributes

- uint32_t **minImageCount**
- uint32_t **maxImageCount**
- VkExtent2D **currentExtent**
- VkExtent2D **minImageExtent**
- VkExtent2D **maxImageExtent**
- uint32_t **maxImageArrayLayers**
- VkSurfaceTransformFlagsKHR **supportedTransforms**
- VkSurfaceTransformFlagBitsKHR **currentTransform**
- VkCompositeAlphaFlagsKHR **supportedCompositeAlpha**
- VkImageUsageFlags **supportedUsageFlags**

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

- lib/glfw/deps/glad/vulkan.h

26.405 VkSurfaceFormatKHR Struct Reference

Public Attributes

- VkFormat **format**
- VkColorSpaceKHR **colorSpace**

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

- lib/glfw/deps/glad/vulkan.h

26.406 VkSwapchainCreateInfoKHR Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkSwapchainCreateFlagsKHR **flags**
- VkSurfaceKHR **surface**
- uint32_t **minImageCount**
- VkFormat **imageFormat**
- VkColorSpaceKHR **imageColorSpace**
- VkExtent2D **imageExtent**
- uint32_t **imageArrayLayers**
- VkImageUsageFlags **imageUsage**
- VkSharingMode **imageSharingMode**
- uint32_t **queueFamilyIndexCount**
- const uint32_t * **pQueueFamilyIndices**
- VkSurfaceTransformFlagBitsKHR **preTransform**
- VkCompositeAlphaFlagBitsKHR **compositeAlpha**
- VkPresentModeKHR **presentMode**
- VkBool32 **clipped**
- VkSwapchainKHR **oldSwapchain**

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

- lib/glfw/deps/glad/vulkan.h

26.407 VkVertexInputAttributeDescription Struct Reference

Public Attributes

- uint32_t **location**
- uint32_t **binding**
- VkFormat **format**
- uint32_t **offset**

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

- lib/glfw/deps/glad/vulkan.h

26.408 VkVertexInputBindingDescription Struct Reference

Public Attributes

- uint32_t **binding**
- uint32_t **stride**
- VkVertexInputRate **inputRate**

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

- lib/glfw/deps/glad/vulkan.h

26.409 VkViewport Struct Reference

Public Attributes

- float **x**
- float **y**
- float **width**
- float **height**
- float **minDepth**
- float **maxDepth**

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

- lib/glfw/deps/glad/vulkan.h

26.410 VkWaylandSurfaceCreateInfoKHR Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkWaylandSurfaceCreateFlagsKHR **flags**
- struct wl_display * **display**
- struct wl_surface * **surface**

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

- lib/glfw/src/wl_platform.h

26.411 VkWin32SurfaceCreateInfoKHR Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkWin32SurfaceCreateFlagsKHR **flags**
- HINSTANCE **hinstance**
- HWND **hwnd**

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

- lib/glfw/src/win32_platform.h

26.412 VkWriteDescriptorSet Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkDescriptorSet **dstSet**
- uint32_t **dstBinding**
- uint32_t **dstArrayElement**
- uint32_t **descriptorCount**
- VkDescriptorType **descriptorType**
- const VkDescriptorImageInfo * **pImageInfo**
- const VkDescriptorBufferInfo * **pBufferInfo**
- const VkBufferView * **pTexelBufferView**

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

- lib/glfw/deps/glad/vulkan.h

26.413 VkXcbSurfaceCreateInfoKHR Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkXcbSurfaceCreateFlagsKHR **flags**
- xcb_connection_t * **connection**
- xcb_window_t **window**

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

- lib/glfw/src/x11_platform.h

26.414 VkXlibSurfaceCreateInfoKHR Struct Reference

Public Attributes

- VkStructureType **sType**
- const void * **pNext**
- VkXlibSurfaceCreateFlagsKHR **flags**
- Display * **dpy**
- Window **window**

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

- lib/glfw/src/x11_platform.h

26.415 wl_cursor Struct Reference

Public Attributes

- unsigned int **image_count**
- struct wl_cursor_image ∗∗ **images**
- char ∗ **name**

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

- lib/glfw/src/wl_platform.h

26.416 wl_cursor_image Struct Reference

Public Attributes

- uint32_t **width**
- uint32_t **height**
- uint32_t **hotspot_x**
- uint32_t **hotspot_y**
- uint32_t **delay**

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

- lib/glfw/src/wl_platform.h

Chapter 27

File Documentation

27.1 getopt.h

```
1 /* Copyright (c) 2012, Kim Gräsman
2  * All rights reserved.
3  *
4  * Redistribution and use in source and binary forms, with or without
5  * modification, are permitted provided that the following conditions are met:
6  *  * Redistributions of source code must retain the above copyright notice,
7  *    this list of conditions and the following disclaimer.
8  *  * Redistributions in binary form must reproduce the above copyright notice,
9  *    this list of conditions and the following disclaimer in the documentation
10 *    and/or other materials provided with the distribution.
11 *  * Neither the name of Kim Gräsman nor the names of contributors may be used
12 *    to endorse or promote products derived from this software without specific
13 *    prior written permission.
14 *
15 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
16 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18 * ARE DISCLAIMED. IN NO EVENT SHALL KIM GRÄSMAN BE LIABLE FOR ANY DIRECT,
19 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
20 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
21 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
22 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
23 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
24 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
25 */
26
27 #ifndef INCLUDED_GETOPT_PORT_H
28 #define INCLUDED_GETOPT_PORT_H
29
30 #if defined(__cplusplus)
31 extern "C" {
32 #endif
33
34 extern const int no_argument;
35 extern const int required_argument;
36 extern const int optional_argument;
37
38 extern char* optarg;
39 extern int optind, opterr, optopt;
40
41 struct option {
42   const char* name;
43   int has_arg;
44   int* flag;
45   int val;
46 };
47
48 int getopt(int argc, char* const argv[], const char* optstring);
49
50 int getopt_long(int argc, char* const argv[],
51   const char* optstring, const struct option* longopts, int* longindex);
52
53 #if defined(__cplusplus)
54 }
55 #endif
56
57 #endif // INCLUDED_GETOPT_PORT_H
```

27.2 gl.h

```
1
28 #ifndef GLAD_GL_H_
29 #define GLAD_GL_H_
30
31 #ifdef __clang__
32 #pragma clang diagnostic push
33 #pragma clang diagnostic ignored "-Wreserved-id-macro"
34 #endif
35 #ifdef __gl_h_
36   #error OpenGL (gl.h) header already included (API: gl), remove previous include!
37 #endif
38 #define __gl_h_ 1
39 #ifdef __gl3_h_
40   #error OpenGL (gl3.h) header already included (API: gl), remove previous include!
41 #endif
42 #define __gl3_h_ 1
43 #ifdef __glext_h_
44   #error OpenGL (glext.h) header already included (API: gl), remove previous include!
45 #endif
46 #define __glext_h_ 1
47 #ifdef __gl3ext_h_
48   #error OpenGL (gl3ext.h) header already included (API: gl), remove previous include!
49 #endif
50 #define __gl3ext_h_ 1
51 #ifdef __clang__
52 #pragma clang diagnostic pop
53 #endif
54
55 #define GLAD_GL
56 #define GLAD_OPTION_GL_HEADER_ONLY
57
58 #ifdef __cplusplus
59 extern "C" {
60 #endif
61
62 #ifndef GLAD_PLATFORM_H_
63 #define GLAD_PLATFORM_H_
64
65 #ifndef GLAD_PLATFORM_WIN32
66   #if defined(_WIN32) || defined(__WIN32__) || defined(WIN32) || defined(__MINGW32__)
67     #define GLAD_PLATFORM_WIN32 1
68   #else
69     #define GLAD_PLATFORM_WIN32 0
70   #endif
71 #endif
72
73 #ifndef GLAD_PLATFORM_APPLE
74   #ifdef __APPLE__
75     #define GLAD_PLATFORM_APPLE 1
76   #else
77     #define GLAD_PLATFORM_APPLE 0
78   #endif
79 #endif
80
81 #ifndef GLAD_PLATFORM_EMSCRIPTEN
82   #ifdef __EMSCRIPTEN__
83     #define GLAD_PLATFORM_EMSCRIPTEN 1
84   #else
85     #define GLAD_PLATFORM_EMSCRIPTEN 0
86   #endif
87 #endif
88
89 #ifndef GLAD_PLATFORM_UWP
90   #if defined(_MSC_VER) && !defined(GLAD_INTERNAL_HAVE_WINAPIFAMILY)
91     #ifdef __has_include
92       #if __has_include(<winapifamily.h>)
93         #define GLAD_INTERNAL_HAVE_WINAPIFAMILY 1
94       #endif
95     #elif _MSC_VER >= 1700 && !_USING_V110_SDK71_
96       #define GLAD_INTERNAL_HAVE_WINAPIFAMILY 1
97     #endif
98   #endif
99
100   #ifdef GLAD_INTERNAL_HAVE_WINAPIFAMILY
101     #include <winapifamily.h>
102     #if !WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_DESKTOP) &&
      WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_APP)
103       #define GLAD_PLATFORM_UWP 1
104     #endif
105   #endif
106
107   #ifndef GLAD_PLATFORM_UWP
108     #define GLAD_PLATFORM_UWP 0
109   #endif
110 #endif
```

```
111
112 #ifdef __GNUC__
113   #define GLAD_GNUC_EXTENSION __extension__
114 #else
115   #define GLAD_GNUC_EXTENSION
116 #endif
117
118 #ifndef GLAD_API_CALL
119   #if defined(GLAD_API_CALL_EXPORT)
120     #if GLAD_PLATFORM_WIN32 || defined(__CYGWIN__)
121       #if defined(GLAD_API_CALL_EXPORT_BUILD)
122         #if defined(__GNUC__)
123           #define GLAD_API_CALL __attribute__ ((dllexport)) extern
124         #else
125           #define GLAD_API_CALL __declspec(dllexport) extern
126         #endif
127       #else
128         #if defined(__GNUC__)
129           #define GLAD_API_CALL __attribute__ ((dllimport)) extern
130         #else
131           #define GLAD_API_CALL __declspec(dllimport) extern
132         #endif
133       #endif
134     #elif defined(__GNUC__) && defined(GLAD_API_CALL_EXPORT_BUILD)
135       #define GLAD_API_CALL __attribute__ ((visibility ("default"))) extern
136     #else
137       #define GLAD_API_CALL extern
138     #endif
139   #else
140     #define GLAD_API_CALL extern
141   #endif
142 #endif
143
144 #ifdef APIENTRY
145   #define GLAD_API_PTR APIENTRY
146 #elif GLAD_PLATFORM_WIN32
147   #define GLAD_API_PTR __stdcall
148 #else
149   #define GLAD_API_PTR
150 #endif
151
152 #ifndef GLAPI
153 #define GLAPI GLAD_API_CALL
154 #endif
155
156 #ifndef GLAPIENTRY
157 #define GLAPIENTRY GLAD_API_PTR
158 #endif
159
160 #define GLAD_MAKE_VERSION(major, minor) (major * 10000 + minor)
161 #define GLAD_VERSION_MAJOR(version) (version / 10000)
162 #define GLAD_VERSION_MINOR(version) (version % 10000)
163
164 #define GLAD_GENERATOR_VERSION "2.0.0-beta"
165
166 typedef void (*GLADapiproc)(void);
167
168 typedef GLADapiproc (*GLADloadfunc)(const char *name);
169 typedef GLADapiproc (*GLADuserptrloadfunc)(void *userptr, const char *name);
170
171 typedef void (*GLADprecallback)(const char *name, GLADapiproc apiproc, int len_args, ...);
172 typedef void (*GLADpostcallback)(void *ret, const char *name, GLADapiproc apiproc, int len_args, ...);
173
174 #endif /* GLAD_PLATFORM_H_ */
175
176 #define GL_2D 0x0600
177 #define GL_2_BYTES 0x1407
178 #define GL_3D 0x0601
179 #define GL_3D_COLOR 0x0602
180 #define GL_3D_COLOR_TEXTURE 0x0603
181 #define GL_3_BYTES 0x1408
182 #define GL_4D_COLOR_TEXTURE 0x0604
183 #define GL_4_BYTES 0x1409
184 #define GL_ACCUM 0x0100
185 #define GL_ACCUM_ALPHA_BITS 0x0D5B
186 #define GL_ACCUM_BLUE_BITS 0x0D5A
187 #define GL_ACCUM_BUFFER_BIT 0x00000200
188 #define GL_ACCUM_CLEAR_VALUE 0x0B80
189 #define GL_ACCUM_GREEN_BITS 0x0D59
190 #define GL_ACCUM_RED_BITS 0x0D58
191 #define GL_ACTIVE_ATTRIBUTES 0x8B89
192 #define GL_ACTIVE_ATTRIBUTE_MAX_LENGTH 0x8B8A
193 #define GL_ACTIVE_TEXTURE 0x84E0
194 #define GL_ACTIVE_UNIFORMS 0x8B86
195 #define GL_ACTIVE_UNIFORM_BLOCKS 0x8A36
196 #define GL_ACTIVE_UNIFORM_BLOCK_MAX_NAME_LENGTH 0x8A35
197 #define GL_ACTIVE_UNIFORM_MAX_LENGTH 0x8B87
```

```
198 #define GL_ADD 0x0104
199 #define GL_ADD_SIGNED 0x8574
200 #define GL_ALIASED_LINE_WIDTH_RANGE 0x846E
201 #define GL_ALIASED_POINT_SIZE_RANGE 0x846D
202 #define GL_ALL_ATTRIB_BITS 0xFFFFFFFF
203 #define GL_ALPHA 0x1906
204 #define GL_ALPHA12 0x803D
205 #define GL_ALPHA16 0x803E
206 #define GL_ALPHA4 0x803B
207 #define GL_ALPHA8 0x803C
208 #define GL_ALPHA_BIAS 0x0D1D
209 #define GL_ALPHA_BITS 0x0D55
210 #define GL_ALPHA_INTEGER 0x8D97
211 #define GL_ALPHA_SCALE 0x0D1C
212 #define GL_ALPHA_TEST 0x0BC0
213 #define GL_ALPHA_TEST_FUNC 0x0BC1
214 #define GL_ALPHA_TEST_REF 0x0BC2
215 #define GL_ALREADY_SIGNALED 0x911A
216 #define GL_ALWAYS 0x0207
217 #define GL_AMBIENT 0x1200
218 #define GL_AMBIENT_AND_DIFFUSE 0x1602
219 #define GL_AND 0x1501
220 #define GL_AND_INVERTED 0x1504
221 #define GL_AND_REVERSE 0x1502
222 #define GL_ANY_SAMPLES_PASSED 0x8C2F
223 #define GL_ARRAY_BUFFER 0x8892
224 #define GL_ARRAY_BUFFER_BINDING 0x8894
225 #define GL_ATTACHED_SHADERS 0x8B85
226 #define GL_ATTRIB_STACK_DEPTH 0x0BB0
227 #define GL_AUTO_NORMAL 0x0D80
228 #define GL_AUX0 0x0409
229 #define GL_AUX1 0x040A
230 #define GL_AUX2 0x040B
231 #define GL_AUX3 0x040C
232 #define GL_AUX_BUFFERS 0x0C00
233 #define GL_BACK 0x0405
234 #define GL_BACK_LEFT 0x0402
235 #define GL_BACK_RIGHT 0x0403
236 #define GL_BGR 0x80E0
237 #define GL_BGRA 0x80E1
238 #define GL_BGRA_INTEGER 0x8D9B
239 #define GL_BGR_INTEGER 0x8D9A
240 #define GL_BITMAP 0x1A00
241 #define GL_BITMAP_TOKEN 0x0704
242 #define GL_BLEND 0x0BE2
243 #define GL_BLEND_COLOR 0x8005
244 #define GL_BLEND_DST 0x0BE0
245 #define GL_BLEND_DST_ALPHA 0x80CA
246 #define GL_BLEND_DST_RGB 0x80C8
247 #define GL_BLEND_EQUATION 0x8009
248 #define GL_BLEND_EQUATION_ALPHA 0x883D
249 #define GL_BLEND_EQUATION_RGB 0x8009
250 #define GL_BLEND_SRC 0x0BE1
251 #define GL_BLEND_SRC_ALPHA 0x80CB
252 #define GL_BLEND_SRC_RGB 0x80C9
253 #define GL_BLUE 0x1905
254 #define GL_BLUE_BIAS 0x0D1B
255 #define GL_BLUE_BITS 0x0D54
256 #define GL_BLUE_INTEGER 0x8D96
257 #define GL_BLUE_SCALE 0x0D1A
258 #define GL_BOOL 0x8B56
259 #define GL_BOOL_VEC2 0x8B57
260 #define GL_BOOL_VEC3 0x8B58
261 #define GL_BOOL_VEC4 0x8B59
262 #define GL_BUFFER 0x82E0
263 #define GL_BUFFER_ACCESS 0x88BB
264 #define GL_BUFFER_ACCESS_FLAGS 0x911F
265 #define GL_BUFFER_MAPPED 0x88BC
266 #define GL_BUFFER_MAP_LENGTH 0x9120
267 #define GL_BUFFER_MAP_OFFSET 0x9121
268 #define GL_BUFFER_MAP_POINTER 0x88BD
269 #define GL_BUFFER_SIZE 0x8764
270 #define GL_BUFFER_USAGE 0x8765
271 #define GL_BYTE 0x1400
272 #define GL_C3F_V3F 0x2A24
273 #define GL_C4F_N3F_V3F 0x2A26
274 #define GL_C4UB_V2F 0x2A22
275 #define GL_C4UB_V3F 0x2A23
276 #define GL_CCW 0x0901
277 #define GL_CLAMP 0x2900
278 #define GL_CLAMP_FRAGMENT_COLOR 0x891B
279 #define GL_CLAMP_READ_COLOR 0x891C
280 #define GL_CLAMP_TO_BORDER 0x812D
281 #define GL_CLAMP_TO_EDGE 0x812F
282 #define GL_CLAMP_VERTEX_COLOR 0x891A
283 #define GL_CLEAR 0x1500
284 #define GL_CLIENT_ACTIVE_TEXTURE 0x84E1
```

```
285 #define GL_CLIENT_ALL_ATTRIB_BITS 0xFFFFFFFF
286 #define GL_CLIENT_ATTRIB_STACK_DEPTH 0x0BB1
287 #define GL_CLIENT_PIXEL_STORE_BIT 0x00000001
288 #define GL_CLIENT_VERTEX_ARRAY_BIT 0x00000002
289 #define GL_CLIP_DISTANCE0 0x3000
290 #define GL_CLIP_DISTANCE1 0x3001
291 #define GL_CLIP_DISTANCE2 0x3002
292 #define GL_CLIP_DISTANCE3 0x3003
293 #define GL_CLIP_DISTANCE4 0x3004
294 #define GL_CLIP_DISTANCE5 0x3005
295 #define GL_CLIP_DISTANCE6 0x3006
296 #define GL_CLIP_DISTANCE7 0x3007
297 #define GL_CLIP_PLANE0 0x3000
298 #define GL_CLIP_PLANE1 0x3001
299 #define GL_CLIP_PLANE2 0x3002
300 #define GL_CLIP_PLANE3 0x3003
301 #define GL_CLIP_PLANE4 0x3004
302 #define GL_CLIP_PLANE5 0x3005
303 #define GL_COEFF 0x0A00
304 #define GL_COLOR 0x1800
305 #define GL_COLOR_ARRAY 0x8076
306 #define GL_COLOR_ARRAY_BUFFER_BINDING 0x8898
307 #define GL_COLOR_ARRAY_POINTER 0x8090
308 #define GL_COLOR_ARRAY_SIZE 0x8081
309 #define GL_COLOR_ARRAY_STRIDE 0x8083
310 #define GL_COLOR_ARRAY_TYPE 0x8082
311 #define GL_COLOR_ATTACHMENT0 0x8CE0
312 #define GL_COLOR_ATTACHMENT1 0x8CE1
313 #define GL_COLOR_ATTACHMENT10 0x8CEA
314 #define GL_COLOR_ATTACHMENT11 0x8CEB
315 #define GL_COLOR_ATTACHMENT12 0x8CEC
316 #define GL_COLOR_ATTACHMENT13 0x8CED
317 #define GL_COLOR_ATTACHMENT14 0x8CEE
318 #define GL_COLOR_ATTACHMENT15 0x8CEF
319 #define GL_COLOR_ATTACHMENT16 0x8CF0
320 #define GL_COLOR_ATTACHMENT17 0x8CF1
321 #define GL_COLOR_ATTACHMENT18 0x8CF2
322 #define GL_COLOR_ATTACHMENT19 0x8CF3
323 #define GL_COLOR_ATTACHMENT2 0x8CE2
324 #define GL_COLOR_ATTACHMENT20 0x8CF4
325 #define GL_COLOR_ATTACHMENT21 0x8CF5
326 #define GL_COLOR_ATTACHMENT22 0x8CF6
327 #define GL_COLOR_ATTACHMENT23 0x8CF7
328 #define GL_COLOR_ATTACHMENT24 0x8CF8
329 #define GL_COLOR_ATTACHMENT25 0x8CF9
330 #define GL_COLOR_ATTACHMENT26 0x8CFA
331 #define GL_COLOR_ATTACHMENT27 0x8CFB
332 #define GL_COLOR_ATTACHMENT28 0x8CFC
333 #define GL_COLOR_ATTACHMENT29 0x8CFD
334 #define GL_COLOR_ATTACHMENT3 0x8CE3
335 #define GL_COLOR_ATTACHMENT30 0x8CFE
336 #define GL_COLOR_ATTACHMENT31 0x8CFF
337 #define GL_COLOR_ATTACHMENT4 0x8CE4
338 #define GL_COLOR_ATTACHMENT5 0x8CE5
339 #define GL_COLOR_ATTACHMENT6 0x8CE6
340 #define GL_COLOR_ATTACHMENT7 0x8CE7
341 #define GL_COLOR_ATTACHMENT8 0x8CE8
342 #define GL_COLOR_ATTACHMENT9 0x8CE9
343 #define GL_COLOR_BUFFER_BIT 0x00004000
344 #define GL_COLOR_CLEAR_VALUE 0x0C22
345 #define GL_COLOR_INDEX 0x1900
346 #define GL_COLOR_INDEXES 0x1603
347 #define GL_COLOR_LOGIC_OP 0x0BF2
348 #define GL_COLOR_MATERIAL 0x0B57
349 #define GL_COLOR_MATERIAL_FACE 0x0B55
350 #define GL_COLOR_MATERIAL_PARAMETER 0x0B56
351 #define GL_COLOR_SUM 0x8458
352 #define GL_COLOR_WRITEMASK 0x0C23
353 #define GL_COMBINE 0x8570
354 #define GL_COMBINE_ALPHA 0x8572
355 #define GL_COMBINE_RGB 0x8571
356 #define GL_COMPARE_REF_TO_TEXTURE 0x884E
357 #define GL_COMPARE_R_TO_TEXTURE 0x884E
358 #define GL_COMPILE 0x1300
359 #define GL_COMPILE_AND_EXECUTE 0x1301
360 #define GL_COMPILE_STATUS 0x8B81
361 #define GL_COMPRESSED_ALPHA 0x84E9
362 #define GL_COMPRESSED_INTENSITY 0x84EC
363 #define GL_COMPRESSED_LUMINANCE 0x84EA
364 #define GL_COMPRESSED_LUMINANCE_ALPHA 0x84EB
365 #define GL_COMPRESSED_RED 0x8225
366 #define GL_COMPRESSED_RED_RGTC1 0x8DBB
367 #define GL_COMPRESSED_RG 0x8226
368 #define GL_COMPRESSED_RGB 0x84ED
369 #define GL_COMPRESSED_RGBA 0x84EE
370 #define GL_COMPRESSED_RG_RGTC2 0x8DBD
371 #define GL_COMPRESSED_SIGNED_RED_RGTC1 0x8DBC
```

```
372 #define GL_COMPRESSED_SIGNED_RG_RGTC2 0x8DBE
373 #define GL_COMPRESSED_SLUMINANCE 0x8C4A
374 #define GL_COMPRESSED_SLUMINANCE_ALPHA 0x8C4B
375 #define GL_COMPRESSED_SRGB 0x8C48
376 #define GL_COMPRESSED_SRGB_ALPHA 0x8C49
377 #define GL_COMPRESSED_TEXTURE_FORMATS 0x86A3
378 #define GL_CONDITION_SATISFIED 0x911C
379 #define GL_CONSTANT 0x8576
380 #define GL_CONSTANT_ALPHA 0x8003
381 #define GL_CONSTANT_ATTENUATION 0x1207
382 #define GL_CONSTANT_COLOR 0x8001
383 #define GL_CONTEXT_COMPATIBILITY_PROFILE_BIT 0x00000002
384 #define GL_CONTEXT_CORE_PROFILE_BIT 0x00000001
385 #define GL_CONTEXT_FLAGS 0x821E
386 #define GL_CONTEXT_FLAG_DEBUG_BIT 0x00000002
387 #define GL_CONTEXT_FLAG_FORWARD_COMPATIBLE_BIT 0x00000001
388 #define GL_CONTEXT_FLAG_ROBUST_ACCESS_BIT_ARB 0x00000004
389 #define GL_CONTEXT_PROFILE_MASK 0x9126
390 #define GL_COORD_REPLACE 0x8862
391 #define GL_COPY 0x1503
392 #define GL_COPY_INVERTED 0x150C
393 #define GL_COPY_PIXEL_TOKEN 0x0706
394 #define GL_COPY_READ_BUFFER 0x8F36
395 #define GL_COPY_WRITE_BUFFER 0x8F37
396 #define GL_CULL_FACE 0x0B44
397 #define GL_CULL_FACE_MODE 0x0B45
398 #define GL_CURRENT_BIT 0x00000001
399 #define GL_CURRENT_COLOR 0x0B00
400 #define GL_CURRENT_FOG_COORD 0x8453
401 #define GL_CURRENT_FOG_COORDINATE 0x8453
402 #define GL_CURRENT_INDEX 0x0B01
403 #define GL_CURRENT_NORMAL 0x0B02
404 #define GL_CURRENT_PROGRAM 0x8B8D
405 #define GL_CURRENT_QUERY 0x8865
406 #define GL_CURRENT_RASTER_COLOR 0x0B04
407 #define GL_CURRENT_RASTER_DISTANCE 0x0B09
408 #define GL_CURRENT_RASTER_INDEX 0x0B05
409 #define GL_CURRENT_RASTER_POSITION 0x0B07
410 #define GL_CURRENT_RASTER_POSITION_VALID 0x0B08
411 #define GL_CURRENT_RASTER_SECONDARY_COLOR 0x845F
412 #define GL_CURRENT_RASTER_TEXTURE_COORDS 0x0B06
413 #define GL_CURRENT_SECONDARY_COLOR 0x8459
414 #define GL_CURRENT_TEXTURE_COORDS 0x0B03
415 #define GL_CURRENT_VERTEX_ATTRIB 0x8626
416 #define GL_CW 0x0900
417 #define GL_DEBUG_CALLBACK_FUNCTION 0x8244
418 #define GL_DEBUG_CALLBACK_USER_PARAM 0x8245
419 #define GL_DEBUG_GROUP_STACK_DEPTH 0x826D
420 #define GL_DEBUG_LOGGED_MESSAGES 0x9145
421 #define GL_DEBUG_NEXT_LOGGED_MESSAGE_LENGTH 0x8243
422 #define GL_DEBUG_OUTPUT 0x92E0
423 #define GL_DEBUG_OUTPUT_SYNCHRONOUS 0x8242
424 #define GL_DEBUG_SEVERITY_HIGH 0x9146
425 #define GL_DEBUG_SEVERITY_LOW 0x9148
426 #define GL_DEBUG_SEVERITY_MEDIUM 0x9147
427 #define GL_DEBUG_SEVERITY_NOTIFICATION 0x826B
428 #define GL_DEBUG_SOURCE_API 0x8246
429 #define GL_DEBUG_SOURCE_APPLICATION 0x824A
430 #define GL_DEBUG_SOURCE_OTHER 0x824B
431 #define GL_DEBUG_SOURCE_SHADER_COMPILER 0x8248
432 #define GL_DEBUG_SOURCE_THIRD_PARTY 0x8249
433 #define GL_DEBUG_SOURCE_WINDOW_SYSTEM 0x8247
434 #define GL_DEBUG_TYPE_DEPRECATED_BEHAVIOR 0x824D
435 #define GL_DEBUG_TYPE_ERROR 0x824C
436 #define GL_DEBUG_TYPE_MARKER 0x8268
437 #define GL_DEBUG_TYPE_OTHER 0x8251
438 #define GL_DEBUG_TYPE_PERFORMANCE 0x8250
439 #define GL_DEBUG_TYPE_POP_GROUP 0x826A
440 #define GL_DEBUG_TYPE_PORTABILITY 0x824F
441 #define GL_DEBUG_TYPE_PUSH_GROUP 0x8269
442 #define GL_DEBUG_TYPE_UNDEFINED_BEHAVIOR 0x824E
443 #define GL_DECAL 0x2101
444 #define GL_DECR 0x1E03
445 #define GL_DECR_WRAP 0x8508
446 #define GL_DELETE_STATUS 0x8B80
447 #define GL_DEPTH 0x1801
448 #define GL_DEPTH24_STENCIL8 0x88F0
449 #define GL_DEPTH32F_STENCIL8 0x8CAD
450 #define GL_DEPTH_ATTACHMENT 0x8D00
451 #define GL_DEPTH_BIAS 0x0D1F
452 #define GL_DEPTH_BITS 0x0D56
453 #define GL_DEPTH_BUFFER_BIT 0x00000100
454 #define GL_DEPTH_CLAMP 0x864F
455 #define GL_DEPTH_CLEAR_VALUE 0x0B73
456 #define GL_DEPTH_COMPONENT 0x1902
457 #define GL_DEPTH_COMPONENT16 0x81A5
458 #define GL_DEPTH_COMPONENT24 0x81A6
```

```
459 #define GL_DEPTH_COMPONENT32 0x81A7
460 #define GL_DEPTH_COMPONENT32F 0x8CAC
461 #define GL_DEPTH_FUNC 0x0B74
462 #define GL_DEPTH_RANGE 0x0B70
463 #define GL_DEPTH_SCALE 0x0D1E
464 #define GL_DEPTH_STENCIL 0x84F9
465 #define GL_DEPTH_STENCIL_ATTACHMENT 0x821A
466 #define GL_DEPTH_TEST 0x0B71
467 #define GL_DEPTH_TEXTURE_MODE 0x884B
468 #define GL_DEPTH_WRITEMASK 0x0B72
469 #define GL_DIFFUSE 0x1201
470 #define GL_DISPLAY_LIST 0x82E7
471 #define GL_DITHER 0x0BD0
472 #define GL_DOMAIN 0x0A02
473 #define GL_DONT_CARE 0x1100
474 #define GL_DOT3_RGB 0x86AE
475 #define GL_DOT3_RGBA 0x86AF
476 #define GL_DOUBLE 0x140A
477 #define GL_DOUBLEBUFFER 0x0C32
478 #define GL_DRAW_BUFFER 0x0C01
479 #define GL_DRAW_BUFFER0 0x8825
480 #define GL_DRAW_BUFFER1 0x8826
481 #define GL_DRAW_BUFFER10 0x882F
482 #define GL_DRAW_BUFFER11 0x8830
483 #define GL_DRAW_BUFFER12 0x8831
484 #define GL_DRAW_BUFFER13 0x8832
485 #define GL_DRAW_BUFFER14 0x8833
486 #define GL_DRAW_BUFFER15 0x8834
487 #define GL_DRAW_BUFFER2 0x8827
488 #define GL_DRAW_BUFFER3 0x8828
489 #define GL_DRAW_BUFFER4 0x8829
490 #define GL_DRAW_BUFFER5 0x882A
491 #define GL_DRAW_BUFFER6 0x882B
492 #define GL_DRAW_BUFFER7 0x882C
493 #define GL_DRAW_BUFFER8 0x882D
494 #define GL_DRAW_BUFFER9 0x882E
495 #define GL_DRAW_FRAMEBUFFER 0x8CA9
496 #define GL_DRAW_FRAMEBUFFER_BINDING 0x8CA6
497 #define GL_DRAW_PIXEL_TOKEN 0x0705
498 #define GL_DST_ALPHA 0x0304
499 #define GL_DST_COLOR 0x0306
500 #define GL_DYNAMIC_COPY 0x88EA
501 #define GL_DYNAMIC_DRAW 0x88E8
502 #define GL_DYNAMIC_READ 0x88E9
503 #define GL_EDGE_FLAG 0x0B43
504 #define GL_EDGE_FLAG_ARRAY 0x8079
505 #define GL_EDGE_FLAG_ARRAY_BUFFER_BINDING 0x889B
506 #define GL_EDGE_FLAG_ARRAY_POINTER 0x8093
507 #define GL_EDGE_FLAG_ARRAY_STRIDE 0x808C
508 #define GL_ELEMENT_ARRAY_BUFFER 0x8893
509 #define GL_ELEMENT_ARRAY_BUFFER_BINDING 0x8895
510 #define GL_EMISSION 0x1600
511 #define GL_ENABLE_BIT 0x00002000
512 #define GL_EQUAL 0x0202
513 #define GL_EQUIV 0x1509
514 #define GL_EVAL_BIT 0x00010000
515 #define GL_EXP 0x0800
516 #define GL_EXP2 0x0801
517 #define GL_EXTENSIONS 0x1F03
518 #define GL_EYE_LINEAR 0x2400
519 #define GL_EYE_PLANE 0x2502
520 #define GL_FALSE 0
521 #define GL_FASTEST 0x1101
522 #define GL_FEEDBACK 0x1C01
523 #define GL_FEEDBACK_BUFFER_POINTER 0x0DF0
524 #define GL_FEEDBACK_BUFFER_SIZE 0x0DF1
525 #define GL_FEEDBACK_BUFFER_TYPE 0x0DF2
526 #define GL_FILL 0x1B02
527 #define GL_FIRST_VERTEX_CONVENTION 0x8E4D
528 #define GL_FIXED_ONLY 0x891D
529 #define GL_FLAT 0x1D00
530 #define GL_FLOAT 0x1406
531 #define GL_FLOAT_32_UNSIGNED_INT_24_8_REV 0x8DAD
532 #define GL_FLOAT_MAT2 0x8B5A
533 #define GL_FLOAT_MAT2x3 0x8B65
534 #define GL_FLOAT_MAT2x4 0x8B66
535 #define GL_FLOAT_MAT3 0x8B5B
536 #define GL_FLOAT_MAT3x2 0x8B67
537 #define GL_FLOAT_MAT3x4 0x8B68
538 #define GL_FLOAT_MAT4 0x8B5C
539 #define GL_FLOAT_MAT4x2 0x8B69
540 #define GL_FLOAT_MAT4x3 0x8B6A
541 #define GL_FLOAT_VEC2 0x8B50
542 #define GL_FLOAT_VEC3 0x8B51
543 #define GL_FLOAT_VEC4 0x8B52
544 #define GL_FOG 0x0B60
545 #define GL_FOG_BIT 0x00000080
```

```
546 #define GL_FOG_COLOR 0x0B66
547 #define GL_FOG_COORD 0x8451
548 #define GL_FOG_COORDINATE 0x8451
549 #define GL_FOG_COORDINATE_ARRAY 0x8457
550 #define GL_FOG_COORDINATE_ARRAY_BUFFER_BINDING 0x889D
551 #define GL_FOG_COORDINATE_ARRAY_POINTER 0x8456
552 #define GL_FOG_COORDINATE_ARRAY_STRIDE 0x8455
553 #define GL_FOG_COORDINATE_ARRAY_TYPE 0x8454
554 #define GL_FOG_COORDINATE_SOURCE 0x8450
555 #define GL_FOG_COORD_ARRAY 0x8457
556 #define GL_FOG_COORD_ARRAY_BUFFER_BINDING 0x889D
557 #define GL_FOG_COORD_ARRAY_POINTER 0x8456
558 #define GL_FOG_COORD_ARRAY_STRIDE 0x8455
559 #define GL_FOG_COORD_ARRAY_TYPE 0x8454
560 #define GL_FOG_COORD_SRC 0x8450
561 #define GL_FOG_DENSITY 0x0B62
562 #define GL_FOG_END 0x0B64
563 #define GL_FOG_HINT 0x0C54
564 #define GL_FOG_INDEX 0x0B61
565 #define GL_FOG_MODE 0x0B65
566 #define GL_FOG_START 0x0B63
567 #define GL_FRAGMENT_DEPTH 0x8452
568 #define GL_FRAGMENT_SHADER 0x8B30
569 #define GL_FRAGMENT_SHADER_DERIVATIVE_HINT 0x8B8B
570 #define GL_FRAMEBUFFER 0x8D40
571 #define GL_FRAMEBUFFER_ATTACHMENT_ALPHA_SIZE 0x8215
572 #define GL_FRAMEBUFFER_ATTACHMENT_BLUE_SIZE 0x8214
573 #define GL_FRAMEBUFFER_ATTACHMENT_COLOR_ENCODING 0x8210
574 #define GL_FRAMEBUFFER_ATTACHMENT_COMPONENT_TYPE 0x8211
575 #define GL_FRAMEBUFFER_ATTACHMENT_DEPTH_SIZE 0x8216
576 #define GL_FRAMEBUFFER_ATTACHMENT_GREEN_SIZE 0x8213
577 #define GL_FRAMEBUFFER_ATTACHMENT_LAYERED 0x8DA7
578 #define GL_FRAMEBUFFER_ATTACHMENT_OBJECT_NAME 0x8CD1
579 #define GL_FRAMEBUFFER_ATTACHMENT_OBJECT_TYPE 0x8CD0
580 #define GL_FRAMEBUFFER_ATTACHMENT_RED_SIZE 0x8212
581 #define GL_FRAMEBUFFER_ATTACHMENT_STENCIL_SIZE 0x8217
582 #define GL_FRAMEBUFFER_ATTACHMENT_TEXTURE_CUBE_MAP_FACE 0x8CD3
583 #define GL_FRAMEBUFFER_ATTACHMENT_TEXTURE_LAYER 0x8CD4
584 #define GL_FRAMEBUFFER_ATTACHMENT_TEXTURE_LEVEL 0x8CD2
585 #define GL_FRAMEBUFFER_BINDING 0x8CA6
586 #define GL_FRAMEBUFFER_COMPLETE 0x8CD5
587 #define GL_FRAMEBUFFER_DEFAULT 0x8218
588 #define GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT 0x8CD6
589 #define GL_FRAMEBUFFER_INCOMPLETE_DRAW_BUFFER 0x8CDB
590 #define GL_FRAMEBUFFER_INCOMPLETE_LAYER_TARGETS 0x8DA8
591 #define GL_FRAMEBUFFER_INCOMPLETE_MISSING_ATTACHMENT 0x8CD7
592 #define GL_FRAMEBUFFER_INCOMPLETE_MULTISAMPLE 0x8D56
593 #define GL_FRAMEBUFFER_INCOMPLETE_READ_BUFFER 0x8CDC
594 #define GL_FRAMEBUFFER_SRGB 0x8DB9
595 #define GL_FRAMEBUFFER_UNDEFINED 0x8219
596 #define GL_FRAMEBUFFER_UNSUPPORTED 0x8CDD
597 #define GL_FRONT 0x0404
598 #define GL_FRONT_AND_BACK 0x0408
599 #define GL_FRONT_FACE 0x0B46
600 #define GL_FRONT_LEFT 0x0400
601 #define GL_FRONT_RIGHT 0x0401
602 #define GL_FUNC_ADD 0x8006
603 #define GL_FUNC_REVERSE_SUBTRACT 0x800B
604 #define GL_FUNC_SUBTRACT 0x800A
605 #define GL_GENERATE_MIPMAP 0x8191
606 #define GL_GENERATE_MIPMAP_HINT 0x8192
607 #define GL_GEOMETRY_INPUT_TYPE 0x8917
608 #define GL_GEOMETRY_OUTPUT_TYPE 0x8918
609 #define GL_GEOMETRY_SHADER 0x8DD9
610 #define GL_GEOMETRY_VERTICES_OUT 0x8916
611 #define GL_GEQUAL 0x0206
612 #define GL_GREATER 0x0204
613 #define GL_GREEN 0x1904
614 #define GL_GREEN_BIAS 0x0D19
615 #define GL_GREEN_BITS 0x0D53
616 #define GL_GREEN_INTEGER 0x8D95
617 #define GL_GREEN_SCALE 0x0D18
618 #define GL_GUILTY_CONTEXT_RESET_ARB 0x8253
619 #define GL_HALF_FLOAT 0x140B
620 #define GL_HINT_BIT 0x00008000
621 #define GL_INCR 0x1E02
622 #define GL_INCR_WRAP 0x8507
623 #define GL_INDEX 0x8222
624 #define GL_INDEX_ARRAY 0x8077
625 #define GL_INDEX_ARRAY_BUFFER_BINDING 0x8899
626 #define GL_INDEX_ARRAY_POINTER 0x8091
627 #define GL_INDEX_ARRAY_STRIDE 0x8086
628 #define GL_INDEX_ARRAY_TYPE 0x8085
629 #define GL_INDEX_BITS 0x0D51
630 #define GL_INDEX_CLEAR_VALUE 0x0C20
631 #define GL_INDEX_LOGIC_OP 0x0BF1
632 #define GL_INDEX_MODE 0x0C30
```

```
633 #define GL_INDEX_OFFSET 0x0D13
634 #define GL_INDEX_SHIFT 0x0D12
635 #define GL_INDEX_WRITEMASK 0x0C21
636 #define GL_INFO_LOG_LENGTH 0x8B84
637 #define GL_INNOCENT_CONTEXT_RESET_ARB 0x8254
638 #define GL_INT 0x1404
639 #define GL_INTENSITY 0x8049
640 #define GL_INTENSITY12 0x804C
641 #define GL_INTENSITY16 0x804D
642 #define GL_INTENSITY4 0x804A
643 #define GL_INTENSITY8 0x804B
644 #define GL_INTERLEAVED_ATTRIBS 0x8C8C
645 #define GL_INTERPOLATE 0x8575
646 #define GL_INT_2_10_10_10_REV 0x8D9F
647 #define GL_INT_SAMPLER_1D 0x8DC9
648 #define GL_INT_SAMPLER_1D_ARRAY 0x8DCE
649 #define GL_INT_SAMPLER_2D 0x8DCA
650 #define GL_INT_SAMPLER_2D_ARRAY 0x8DCF
651 #define GL_INT_SAMPLER_2D_MULTISAMPLE 0x9109
652 #define GL_INT_SAMPLER_2D_MULTISAMPLE_ARRAY 0x910C
653 #define GL_INT_SAMPLER_2D_RECT 0x8DCD
654 #define GL_INT_SAMPLER_3D 0x8DCB
655 #define GL_INT_SAMPLER_BUFFER 0x8DD0
656 #define GL_INT_SAMPLER_CUBE 0x8DCC
657 #define GL_INT_VEC2 0x8B53
658 #define GL_INT_VEC3 0x8B54
659 #define GL_INT_VEC4 0x8B55
660 #define GL_INVALID_ENUM 0x0500
661 #define GL_INVALID_FRAMEBUFFER_OPERATION 0x0506
662 #define GL_INVALID_INDEX 0xFFFFFFFF
663 #define GL_INVALID_OPERATION 0x0502
664 #define GL_INVALID_VALUE 0x0501
665 #define GL_INVERT 0x150A
666 #define GL_KEEP 0x1E00
667 #define GL_LAST_VERTEX_CONVENTION 0x8E4E
668 #define GL_LEFT 0x0406
669 #define GL_LEQUAL 0x0203
670 #define GL_LESS 0x0201
671 #define GL_LIGHT0 0x4000
672 #define GL_LIGHT1 0x4001
673 #define GL_LIGHT2 0x4002
674 #define GL_LIGHT3 0x4003
675 #define GL_LIGHT4 0x4004
676 #define GL_LIGHT5 0x4005
677 #define GL_LIGHT6 0x4006
678 #define GL_LIGHT7 0x4007
679 #define GL_LIGHTING 0x0B50
680 #define GL_LIGHTING_BIT 0x00000040
681 #define GL_LIGHT_MODEL_AMBIENT 0x0B53
682 #define GL_LIGHT_MODEL_COLOR_CONTROL 0x81F8
683 #define GL_LIGHT_MODEL_LOCAL_VIEWER 0x0B51
684 #define GL_LIGHT_MODEL_TWO_SIDE 0x0B52
685 #define GL_LINE 0x1B01
686 #define GL_LINEAR 0x2601
687 #define GL_LINEAR_ATTENUATION 0x1208
688 #define GL_LINEAR_MIPMAP_LINEAR 0x2703
689 #define GL_LINEAR_MIPMAP_NEAREST 0x2701
690 #define GL_LINES 0x0001
691 #define GL_LINES_ADJACENCY 0x000A
692 #define GL_LINE_BIT 0x00000004
693 #define GL_LINE_LOOP 0x0002
694 #define GL_LINE_RESET_TOKEN 0x0707
695 #define GL_LINE_SMOOTH 0x0B20
696 #define GL_LINE_SMOOTH_HINT 0x0C52
697 #define GL_LINE_STIPPLE 0x0B24
698 #define GL_LINE_STIPPLE_PATTERN 0x0B25
699 #define GL_LINE_STIPPLE_REPEAT 0x0B26
700 #define GL_LINE_STRIP 0x0003
701 #define GL_LINE_STRIP_ADJACENCY 0x000B
702 #define GL_LINE_TOKEN 0x0702
703 #define GL_LINE_WIDTH 0x0B21
704 #define GL_LINE_WIDTH_GRANULARITY 0x0B23
705 #define GL_LINE_WIDTH_RANGE 0x0B22
706 #define GL_LINK_STATUS 0x8B82
707 #define GL_LIST_BASE 0x0B32
708 #define GL_LIST_BIT 0x00020000
709 #define GL_LIST_INDEX 0x0B33
710 #define GL_LIST_MODE 0x0B30
711 #define GL_LOAD 0x0101
712 #define GL_LOGIC_OP 0x0BF1
713 #define GL_LOGIC_OP_MODE 0x0BF0
714 #define GL_LOSE_CONTEXT_ON_RESET_ARB 0x8252
715 #define GL_LOWER_LEFT 0x8CA1
716 #define GL_LUMINANCE 0x1909
717 #define GL_LUMINANCE12 0x8041
718 #define GL_LUMINANCE12_ALPHA12 0x8047
719 #define GL_LUMINANCE12_ALPHA4 0x8046
```

```
720 #define GL_LUMINANCE16 0x8042
721 #define GL_LUMINANCE16_ALPHA16 0x8048
722 #define GL_LUMINANCE4 0x803F
723 #define GL_LUMINANCE4_ALPHA4 0x8043
724 #define GL_LUMINANCE6_ALPHA2 0x8044
725 #define GL_LUMINANCE8 0x8040
726 #define GL_LUMINANCE8_ALPHA8 0x8045
727 #define GL_LUMINANCE_ALPHA 0x190A
728 #define GL_MAJOR_VERSION 0x821B
729 #define GL_MAP1_COLOR_4 0x0D90
730 #define GL_MAP1_GRID_DOMAIN 0x0DD0
731 #define GL_MAP1_GRID_SEGMENTS 0x0DD1
732 #define GL_MAP1_INDEX 0x0D91
733 #define GL_MAP1_NORMAL 0x0D92
734 #define GL_MAP1_TEXTURE_COORD_1 0x0D93
735 #define GL_MAP1_TEXTURE_COORD_2 0x0D94
736 #define GL_MAP1_TEXTURE_COORD_3 0x0D95
737 #define GL_MAP1_TEXTURE_COORD_4 0x0D96
738 #define GL_MAP1_VERTEX_3 0x0D97
739 #define GL_MAP1_VERTEX_4 0x0D98
740 #define GL_MAP2_COLOR_4 0x0DB0
741 #define GL_MAP2_GRID_DOMAIN 0x0DD2
742 #define GL_MAP2_GRID_SEGMENTS 0x0DD3
743 #define GL_MAP2_INDEX 0x0DB1
744 #define GL_MAP2_NORMAL 0x0DB2
745 #define GL_MAP2_TEXTURE_COORD_1 0x0DB3
746 #define GL_MAP2_TEXTURE_COORD_2 0x0DB4
747 #define GL_MAP2_TEXTURE_COORD_3 0x0DB5
748 #define GL_MAP2_TEXTURE_COORD_4 0x0DB6
749 #define GL_MAP2_VERTEX_3 0x0DB7
750 #define GL_MAP2_VERTEX_4 0x0DB8
751 #define GL_MAP_COLOR 0x0D10
752 #define GL_MAP_FLUSH_EXPLICIT_BIT 0x0010
753 #define GL_MAP_INVALIDATE_BUFFER_BIT 0x0008
754 #define GL_MAP_INVALIDATE_RANGE_BIT 0x0004
755 #define GL_MAP_READ_BIT 0x0001
756 #define GL_MAP_STENCIL 0x0D11
757 #define GL_MAP_UNSYNCHRONIZED_BIT 0x0020
758 #define GL_MAP_WRITE_BIT 0x0002
759 #define GL_MATRIX_MODE 0x0BA0
760 #define GL_MAX 0x8008
761 #define GL_MAX_3D_TEXTURE_SIZE 0x8073
762 #define GL_MAX_ARRAY_TEXTURE_LAYERS 0x88FF
763 #define GL_MAX_ATTRIB_STACK_DEPTH 0x0D35
764 #define GL_MAX_CLIENT_ATTRIB_STACK_DEPTH 0x0D3B
765 #define GL_MAX_CLIP_DISTANCES 0x0D32
766 #define GL_MAX_CLIP_PLANES 0x0D32
767 #define GL_MAX_COLOR_ATTACHMENTS 0x8CDF
768 #define GL_MAX_COLOR_TEXTURE_SAMPLES 0x910E
769 #define GL_MAX_COMBINED_FRAGMENT_UNIFORM_COMPONENTS 0x8A33
770 #define GL_MAX_COMBINED_GEOMETRY_UNIFORM_COMPONENTS 0x8A32
771 #define GL_MAX_COMBINED_TEXTURE_IMAGE_UNITS 0x8B4D
772 #define GL_MAX_COMBINED_UNIFORM_BLOCKS 0x8A2E
773 #define GL_MAX_COMBINED_VERTEX_UNIFORM_COMPONENTS 0x8A31
774 #define GL_MAX_CUBE_MAP_TEXTURE_SIZE 0x851C
775 #define GL_MAX_DEBUG_GROUP_STACK_DEPTH 0x826C
776 #define GL_MAX_DEBUG_LOGGED_MESSAGES 0x9144
777 #define GL_MAX_DEBUG_MESSAGE_LENGTH 0x9143
778 #define GL_MAX_DEPTH_TEXTURE_SAMPLES 0x910F
779 #define GL_MAX_DRAW_BUFFERS 0x8824
780 #define GL_MAX_DUAL_SOURCE_DRAW_BUFFERS 0x88FC
781 #define GL_MAX_ELEMENTS_INDICES 0x80E9
782 #define GL_MAX_ELEMENTS_VERTICES 0x80E8
783 #define GL_MAX_EVAL_ORDER 0x0D30
784 #define GL_MAX_FRAGMENT_INPUT_COMPONENTS 0x9125
785 #define GL_MAX_FRAGMENT_UNIFORM_BLOCKS 0x8A2D
786 #define GL_MAX_FRAGMENT_UNIFORM_COMPONENTS 0x8B49
787 #define GL_MAX_GEOMETRY_INPUT_COMPONENTS 0x9123
788 #define GL_MAX_GEOMETRY_OUTPUT_COMPONENTS 0x9124
789 #define GL_MAX_GEOMETRY_OUTPUT_VERTICES 0x8DE0
790 #define GL_MAX_GEOMETRY_TEXTURE_IMAGE_UNITS 0x8C29
791 #define GL_MAX_GEOMETRY_TOTAL_OUTPUT_COMPONENTS 0x8DE1
792 #define GL_MAX_GEOMETRY_UNIFORM_BLOCKS 0x8A2C
793 #define GL_MAX_GEOMETRY_UNIFORM_COMPONENTS 0x8DDF
794 #define GL_MAX_INTEGER_SAMPLES 0x9110
795 #define GL_MAX_LABEL_LENGTH 0x82E8
796 #define GL_MAX_LIGHTS 0x0D31
797 #define GL_MAX_LIST_NESTING 0x0B31
798 #define GL_MAX_MODELVIEW_STACK_DEPTH 0x0D36
799 #define GL_MAX_NAME_STACK_DEPTH 0x0D37
800 #define GL_MAX_PIXEL_MAP_TABLE 0x0D34
801 #define GL_MAX_PROGRAM_TEXEL_OFFSET 0x8905
802 #define GL_MAX_PROJECTION_STACK_DEPTH 0x0D38
803 #define GL_MAX_RECTANGLE_TEXTURE_SIZE 0x84F8
804 #define GL_MAX_RENDERBUFFER_SIZE 0x84E8
805 #define GL_MAX_SAMPLES 0x8D57
806 #define GL_MAX_SAMPLE_MASK_WORDS 0x8E59
```

```
807 #define GL_MAX_SERVER_WAIT_TIMEOUT 0x9111
808 #define GL_MAX_TEXTURE_BUFFER_SIZE 0x8C2B
809 #define GL_MAX_TEXTURE_COORDS 0x8871
810 #define GL_MAX_TEXTURE_IMAGE_UNITS 0x8872
811 #define GL_MAX_TEXTURE_LOD_BIAS 0x84FD
812 #define GL_MAX_TEXTURE_SIZE 0x0D33
813 #define GL_MAX_TEXTURE_STACK_DEPTH 0x0D39
814 #define GL_MAX_TEXTURE_UNITS 0x84E2
815 #define GL_MAX_TRANSFORM_FEEDBACK_INTERLEAVED_COMPONENTS 0x8C8A
816 #define GL_MAX_TRANSFORM_FEEDBACK_SEPARATE_ATTRIBS 0x8C8B
817 #define GL_MAX_TRANSFORM_FEEDBACK_SEPARATE_COMPONENTS 0x8C80
818 #define GL_MAX_UNIFORM_BLOCK_SIZE 0x8A30
819 #define GL_MAX_UNIFORM_BUFFER_BINDINGS 0x8A2F
820 #define GL_MAX_VARYING_COMPONENTS 0x8B4B
821 #define GL_MAX_VARYING_FLOATS 0x8B4B
822 #define GL_MAX_VERTEX_ATTRIBS 0x8869
823 #define GL_MAX_VERTEX_OUTPUT_COMPONENTS 0x9122
824 #define GL_MAX_VERTEX_TEXTURE_IMAGE_UNITS 0x8B4C
825 #define GL_MAX_VERTEX_UNIFORM_BLOCKS 0x8A2B
826 #define GL_MAX_VERTEX_UNIFORM_COMPONENTS 0x8B4A
827 #define GL_MAX_VIEWPORT_DIMS 0x0D3A
828 #define GL_MIN 0x8007
829 #define GL_MINOR_VERSION 0x821C
830 #define GL_MIN_PROGRAM_TEXEL_OFFSET 0x8904
831 #define GL_MIRRORED_REPEAT 0x8370
832 #define GL_MODELVIEW 0x1700
833 #define GL_MODELVIEW_MATRIX 0x0BA6
834 #define GL_MODELVIEW_STACK_DEPTH 0x0BA3
835 #define GL_MODULATE 0x2100
836 #define GL_MULT 0x0103
837 #define GL_MULTISAMPLE 0x809D
838 #define GL_MULTISAMPLE_ARB 0x809D
839 #define GL_MULTISAMPLE_BIT 0x20000000
840 #define GL_MULTISAMPLE_BIT_ARB 0x20000000
841 #define GL_N3F_V3F 0x2A25
842 #define GL_NAME_STACK_DEPTH 0x0D70
843 #define GL_NAND 0x150E
844 #define GL_NEAREST 0x2600
845 #define GL_NEAREST_MIPMAP_LINEAR 0x2702
846 #define GL_NEAREST_MIPMAP_NEAREST 0x2700
847 #define GL_NEVER 0x0200
848 #define GL_NICEST 0x1102
849 #define GL_NONE 0
850 #define GL_NOOP 0x1505
851 #define GL_NOR 0x1508
852 #define GL_NORMALIZE 0x0BA1
853 #define GL_NORMAL_ARRAY 0x8075
854 #define GL_NORMAL_ARRAY_BUFFER_BINDING 0x8897
855 #define GL_NORMAL_ARRAY_POINTER 0x808F
856 #define GL_NORMAL_ARRAY_STRIDE 0x807F
857 #define GL_NORMAL_ARRAY_TYPE 0x807E
858 #define GL_NORMAL_MAP 0x8511
859 #define GL_NOTEQUAL 0x0205
860 #define GL_NO_ERROR 0
861 #define GL_NO_RESET_NOTIFICATION_ARB 0x8261
862 #define GL_NUM_COMPRESSED_TEXTURE_FORMATS 0x86A2
863 #define GL_NUM_EXTENSIONS 0x821D
864 #define GL_OBJECT_LINEAR 0x2401
865 #define GL_OBJECT_PLANE 0x2501
866 #define GL_OBJECT_TYPE 0x9112
867 #define GL_ONE 1
868 #define GL_ONE_MINUS_CONSTANT_ALPHA 0x8004
869 #define GL_ONE_MINUS_CONSTANT_COLOR 0x8002
870 #define GL_ONE_MINUS_DST_ALPHA 0x0305
871 #define GL_ONE_MINUS_DST_COLOR 0x0307
872 #define GL_ONE_MINUS_SRC1_ALPHA 0x88FB
873 #define GL_ONE_MINUS_SRC1_COLOR 0x88FA
874 #define GL_ONE_MINUS_SRC_ALPHA 0x0303
875 #define GL_ONE_MINUS_SRC_COLOR 0x0301
876 #define GL_OPERAND0_ALPHA 0x8598
877 #define GL_OPERAND0_RGB 0x8590
878 #define GL_OPERAND1_ALPHA 0x8599
879 #define GL_OPERAND1_RGB 0x8591
880 #define GL_OPERAND2_ALPHA 0x859A
881 #define GL_OPERAND2_RGB 0x8592
882 #define GL_OR 0x1507
883 #define GL_ORDER 0x0A01
884 #define GL_OR_INVERTED 0x150D
885 #define GL_OR_REVERSE 0x150B
886 #define GL_OUT_OF_MEMORY 0x0505
887 #define GL_PACK_ALIGNMENT 0x0D05
888 #define GL_PACK_IMAGE_HEIGHT 0x806C
889 #define GL_PACK_LSB_FIRST 0x0D01
890 #define GL_PACK_ROW_LENGTH 0x0D02
891 #define GL_PACK_SKIP_IMAGES 0x806B
892 #define GL_PACK_SKIP_PIXELS 0x0D04
893 #define GL_PACK_SKIP_ROWS 0x0D03
```

```
894 #define GL_PACK_SWAP_BYTES 0x0D00
895 #define GL_PASS_THROUGH_TOKEN 0x0700
896 #define GL_PERSPECTIVE_CORRECTION_HINT 0x0C50
897 #define GL_PIXEL_MAP_A_TO_A 0x0C79
898 #define GL_PIXEL_MAP_A_TO_A_SIZE 0x0CB9
899 #define GL_PIXEL_MAP_B_TO_B 0x0C78
900 #define GL_PIXEL_MAP_B_TO_B_SIZE 0x0CB8
901 #define GL_PIXEL_MAP_G_TO_G 0x0C77
902 #define GL_PIXEL_MAP_G_TO_G_SIZE 0x0CB7
903 #define GL_PIXEL_MAP_I_TO_A 0x0C75
904 #define GL_PIXEL_MAP_I_TO_A_SIZE 0x0CB5
905 #define GL_PIXEL_MAP_I_TO_B 0x0C74
906 #define GL_PIXEL_MAP_I_TO_B_SIZE 0x0CB4
907 #define GL_PIXEL_MAP_I_TO_G 0x0C73
908 #define GL_PIXEL_MAP_I_TO_G_SIZE 0x0CB3
909 #define GL_PIXEL_MAP_I_TO_I 0x0C70
910 #define GL_PIXEL_MAP_I_TO_I_SIZE 0x0CB0
911 #define GL_PIXEL_MAP_I_TO_R 0x0C72
912 #define GL_PIXEL_MAP_I_TO_R_SIZE 0x0CB2
913 #define GL_PIXEL_MAP_R_TO_R 0x0C76
914 #define GL_PIXEL_MAP_R_TO_R_SIZE 0x0CB6
915 #define GL_PIXEL_MAP_S_TO_S 0x0C71
916 #define GL_PIXEL_MAP_S_TO_S_SIZE 0x0CB1
917 #define GL_PIXEL_MODE_BIT 0x00000020
918 #define GL_PIXEL_PACK_BUFFER 0x88EB
919 #define GL_PIXEL_PACK_BUFFER_BINDING 0x88ED
920 #define GL_PIXEL_UNPACK_BUFFER 0x88EC
921 #define GL_PIXEL_UNPACK_BUFFER_BINDING 0x88EF
922 #define GL_POINT 0x1B00
923 #define GL_POINTS 0x0000
924 #define GL_POINT_BIT 0x00000002
925 #define GL_POINT_DISTANCE_ATTENUATION 0x8129
926 #define GL_POINT_FADE_THRESHOLD_SIZE 0x8128
927 #define GL_POINT_SIZE 0x0B11
928 #define GL_POINT_SIZE_GRANULARITY 0x0B13
929 #define GL_POINT_SIZE_MAX 0x8127
930 #define GL_POINT_SIZE_MIN 0x8126
931 #define GL_POINT_SIZE_RANGE 0x0B12
932 #define GL_POINT_SMOOTH 0x0B10
933 #define GL_POINT_SMOOTH_HINT 0x0C51
934 #define GL_POINT_SPRITE 0x8861
935 #define GL_POINT_SPRITE_COORD_ORIGIN 0x8CA0
936 #define GL_POINT_TOKEN 0x0701
937 #define GL_POLYGON 0x0009
938 #define GL_POLYGON_BIT 0x00000008
939 #define GL_POLYGON_MODE 0x0B40
940 #define GL_POLYGON_OFFSET_FACTOR 0x8038
941 #define GL_POLYGON_OFFSET_FILL 0x8037
942 #define GL_POLYGON_OFFSET_LINE 0x2A02
943 #define GL_POLYGON_OFFSET_POINT 0x2A01
944 #define GL_POLYGON_OFFSET_UNITS 0x2A00
945 #define GL_POLYGON_SMOOTH 0x0B41
946 #define GL_POLYGON_SMOOTH_HINT 0x0C53
947 #define GL_POLYGON_STIPPLE 0x0B42
948 #define GL_POLYGON_STIPPLE_BIT 0x00000010
949 #define GL_POLYGON_TOKEN 0x0703
950 #define GL_POSITION 0x1203
951 #define GL_PREVIOUS 0x8578
952 #define GL_PRIMARY_COLOR 0x8577
953 #define GL_PRIMITIVES_GENERATED 0x8C87
954 #define GL_PRIMITIVE_RESTART 0x8F9D
955 #define GL_PRIMITIVE_RESTART_INDEX 0x8F9E
956 #define GL_PROGRAM 0x82E2
957 #define GL_PROGRAM_PIPELINE 0x82E4
958 #define GL_PROGRAM_POINT_SIZE 0x8642
959 #define GL_PROJECTION 0x1701
960 #define GL_PROJECTION_MATRIX 0x0BA7
961 #define GL_PROJECTION_STACK_DEPTH 0x0BA4
962 #define GL_PROVOKING_VERTEX 0x8E4F
963 #define GL_PROXY_TEXTURE_1D 0x8063
964 #define GL_PROXY_TEXTURE_1D_ARRAY 0x8C19
965 #define GL_PROXY_TEXTURE_2D 0x8064
966 #define GL_PROXY_TEXTURE_2D_ARRAY 0x8C1B
967 #define GL_PROXY_TEXTURE_2D_MULTISAMPLE 0x9101
968 #define GL_PROXY_TEXTURE_2D_MULTISAMPLE_ARRAY 0x9103
969 #define GL_PROXY_TEXTURE_3D 0x8070
970 #define GL_PROXY_TEXTURE_CUBE_MAP 0x851B
971 #define GL_PROXY_TEXTURE_RECTANGLE 0x84F7
972 #define GL_Q 0x2003
973 #define GL_QUADRATIC_ATTENUATION 0x1209
974 #define GL_QUADS 0x0007
975 #define GL_QUADS_FOLLOW_PROVOKING_VERTEX_CONVENTION 0x8E4C
976 #define GL_QUAD_STRIP 0x0008
977 #define GL_QUERY 0x82E3
978 #define GL_QUERY_BY_REGION_NO_WAIT 0x8E16
979 #define GL_QUERY_BY_REGION_WAIT 0x8E15
980 #define GL_QUERY_COUNTER_BITS 0x8864
```

```
981 #define GL_QUERY_NO_WAIT 0x8E14
982 #define GL_QUERY_RESULT 0x8866
983 #define GL_QUERY_RESULT_AVAILABLE 0x8867
984 #define GL_QUERY_WAIT 0x8E13
985 #define GL_R 0x2002
986 #define GL_R11F_G11F_B10F 0x8C3A
987 #define GL_R16 0x822A
988 #define GL_R16F 0x822D
989 #define GL_R16I 0x8233
990 #define GL_R16UI 0x8234
991 #define GL_R16_SNORM 0x8F98
992 #define GL_R32F 0x822E
993 #define GL_R32I 0x8235
994 #define GL_R32UI 0x8236
995 #define GL_R3_G3_B2 0x2A10
996 #define GL_R8 0x8229
997 #define GL_R8I 0x8231
998 #define GL_R8UI 0x8232
999 #define GL_R8_SNORM 0x8F94
1000 #define GL_RASTERIZER_DISCARD 0x8C89
1001 #define GL_READ_BUFFER 0x0C02
1002 #define GL_READ_FRAMEBUFFER 0x8CA8
1003 #define GL_READ_FRAMEBUFFER_BINDING 0x8CAA
1004 #define GL_READ_ONLY 0x88B8
1005 #define GL_READ_WRITE 0x88BA
1006 #define GL_RED 0x1903
1007 #define GL_RED_BIAS 0x0D15
1008 #define GL_RED_BITS 0x0D52
1009 #define GL_RED_INTEGER 0x8D94
1010 #define GL_RED_SCALE 0x0D14
1011 #define GL_REFLECTION_MAP 0x8512
1012 #define GL_RENDER 0x1C00
1013 #define GL_RENDERBUFFER 0x8D41
1014 #define GL_RENDERBUFFER_ALPHA_SIZE 0x8D53
1015 #define GL_RENDERBUFFER_BINDING 0x8CA7
1016 #define GL_RENDERBUFFER_BLUE_SIZE 0x8D52
1017 #define GL_RENDERBUFFER_DEPTH_SIZE 0x8D54
1018 #define GL_RENDERBUFFER_GREEN_SIZE 0x8D51
1019 #define GL_RENDERBUFFER_HEIGHT 0x8D43
1020 #define GL_RENDERBUFFER_INTERNAL_FORMAT 0x8D44
1021 #define GL_RENDERBUFFER_RED_SIZE 0x8D50
1022 #define GL_RENDERBUFFER_SAMPLES 0x8CAB
1023 #define GL_RENDERBUFFER_STENCIL_SIZE 0x8D55
1024 #define GL_RENDERBUFFER_WIDTH 0x8D42
1025 #define GL_RENDERER 0x1F01
1026 #define GL_RENDER_MODE 0x0C40
1027 #define GL_REPEAT 0x2901
1028 #define GL_REPLACE 0x1E01
1029 #define GL_RESCALE_NORMAL 0x803A
1030 #define GL_RESET_NOTIFICATION_STRATEGY_ARB 0x8256
1031 #define GL_RETURN 0x0102
1032 #define GL_RG 0x8227
1033 #define GL_RG16 0x822C
1034 #define GL_RG16F 0x822F
1035 #define GL_RG16I 0x8239
1036 #define GL_RG16UI 0x823A
1037 #define GL_RG16_SNORM 0x8F99
1038 #define GL_RG32F 0x8230
1039 #define GL_RG32I 0x823B
1040 #define GL_RG32UI 0x823C
1041 #define GL_RG8 0x822B
1042 #define GL_RG8I 0x8237
1043 #define GL_RG8UI 0x8238
1044 #define GL_RG8_SNORM 0x8F95
1045 #define GL_RGB 0x1907
1046 #define GL_RGB10 0x8052
1047 #define GL_RGB10_A2 0x8059
1048 #define GL_RGB10_A2UI 0x906F
1049 #define GL_RGB12 0x8053
1050 #define GL_RGB16 0x8054
1051 #define GL_RGB16F 0x881B
1052 #define GL_RGB16I 0x8D89
1053 #define GL_RGB16UI 0x8D77
1054 #define GL_RGB16_SNORM 0x8F9A
1055 #define GL_RGB32F 0x8815
1056 #define GL_RGB32I 0x8D83
1057 #define GL_RGB32UI 0x8D71
1058 #define GL_RGB4 0x804F
1059 #define GL_RGB5 0x8050
1060 #define GL_RGB5_A1 0x8057
1061 #define GL_RGB8 0x8051
1062 #define GL_RGB8I 0x8D8F
1063 #define GL_RGB8UI 0x8D7D
1064 #define GL_RGB8_SNORM 0x8F96
1065 #define GL_RGB9_E5 0x8C3D
1066 #define GL_RGBA 0x1908
1067 #define GL_RGBA12 0x805A
```

```
1068 #define GL_RGBA16 0x805B
1069 #define GL_RGBA16F 0x881A
1070 #define GL_RGBA16I 0x8D88
1071 #define GL_RGBA16UI 0x8D76
1072 #define GL_RGBA16_SNORM 0x8F9B
1073 #define GL_RGBA2 0x8055
1074 #define GL_RGBA32F 0x8814
1075 #define GL_RGBA32I 0x8D82
1076 #define GL_RGBA32UI 0x8D70
1077 #define GL_RGBA4 0x8056
1078 #define GL_RGBA8 0x8058
1079 #define GL_RGBA8I 0x8D8E
1080 #define GL_RGBA8UI 0x8D7C
1081 #define GL_RGBA8_SNORM 0x8F97
1082 #define GL_RGBA_INTEGER 0x8D99
1083 #define GL_RGBA_MODE 0x0C31
1084 #define GL_RGB_INTEGER 0x8D98
1085 #define GL_RGB_SCALE 0x8573
1086 #define GL_RG_INTEGER 0x8228
1087 #define GL_RIGHT 0x0407
1088 #define GL_S 0x2000
1089 #define GL_SAMPLER 0x82E6
1090 #define GL_SAMPLER_1D 0x8B5D
1091 #define GL_SAMPLER_1D_ARRAY 0x8DC0
1092 #define GL_SAMPLER_1D_ARRAY_SHADOW 0x8DC3
1093 #define GL_SAMPLER_1D_SHADOW 0x8B61
1094 #define GL_SAMPLER_2D 0x8B5E
1095 #define GL_SAMPLER_2D_ARRAY 0x8DC1
1096 #define GL_SAMPLER_2D_ARRAY_SHADOW 0x8DC4
1097 #define GL_SAMPLER_2D_MULTISAMPLE 0x9108
1098 #define GL_SAMPLER_2D_MULTISAMPLE_ARRAY 0x910B
1099 #define GL_SAMPLER_2D_RECT 0x8B63
1100 #define GL_SAMPLER_2D_RECT_SHADOW 0x8B64
1101 #define GL_SAMPLER_2D_SHADOW 0x8B62
1102 #define GL_SAMPLER_3D 0x8B5F
1103 #define GL_SAMPLER_BINDING 0x8919
1104 #define GL_SAMPLER_BUFFER 0x8DC2
1105 #define GL_SAMPLER_CUBE 0x8B60
1106 #define GL_SAMPLER_CUBE_SHADOW 0x8DC5
1107 #define GL_SAMPLES 0x80A9
1108 #define GL_SAMPLES_ARB 0x80A9
1109 #define GL_SAMPLES_PASSED 0x8914
1110 #define GL_SAMPLE_ALPHA_TO_COVERAGE 0x809E
1111 #define GL_SAMPLE_ALPHA_TO_COVERAGE_ARB 0x809E
1112 #define GL_SAMPLE_ALPHA_TO_ONE 0x809F
1113 #define GL_SAMPLE_ALPHA_TO_ONE_ARB 0x809F
1114 #define GL_SAMPLE_BUFFERS 0x80A8
1115 #define GL_SAMPLE_BUFFERS_ARB 0x80A8
1116 #define GL_SAMPLE_COVERAGE 0x80A0
1117 #define GL_SAMPLE_COVERAGE_ARB 0x80A0
1118 #define GL_SAMPLE_COVERAGE_INVERT 0x80AB
1119 #define GL_SAMPLE_COVERAGE_INVERT_ARB 0x80AB
1120 #define GL_SAMPLE_COVERAGE_VALUE 0x80AA
1121 #define GL_SAMPLE_COVERAGE_VALUE_ARB 0x80AA
1122 #define GL_SAMPLE_MASK 0x8E51
1123 #define GL_SAMPLE_MASK_VALUE 0x8E52
1124 #define GL_SAMPLE_POSITION 0x8E50
1125 #define GL_SCISSOR_BIT 0x00080000
1126 #define GL_SCISSOR_BOX 0x0C10
1127 #define GL_SCISSOR_TEST 0x0C11
1128 #define GL_SECONDARY_COLOR_ARRAY 0x845E
1129 #define GL_SECONDARY_COLOR_ARRAY_BUFFER_BINDING 0x889C
1130 #define GL_SECONDARY_COLOR_ARRAY_POINTER 0x845D
1131 #define GL_SECONDARY_COLOR_ARRAY_SIZE 0x845A
1132 #define GL_SECONDARY_COLOR_ARRAY_STRIDE 0x845C
1133 #define GL_SECONDARY_COLOR_ARRAY_TYPE 0x845B
1134 #define GL_SELECT 0x1C02
1135 #define GL_SELECTION_BUFFER_POINTER 0x0DF3
1136 #define GL_SELECTION_BUFFER_SIZE 0x0DF4
1137 #define GL_SEPARATE_ATTRIBS 0x8C8D
1138 #define GL_SEPARATE_SPECULAR_COLOR 0x81FA
1139 #define GL_SET 0x150F
1140 #define GL_SHADER 0x82E1
1141 #define GL_SHADER_SOURCE_LENGTH 0x8B88
1142 #define GL_SHADER_TYPE 0x8B4F
1143 #define GL_SHADE_MODEL 0x0B54
1144 #define GL_SHADING_LANGUAGE_VERSION 0x8B8C
1145 #define GL_SHININESS 0x1601
1146 #define GL_SHORT 0x1402
1147 #define GL_SIGNALED 0x9119
1148 #define GL_SIGNED_NORMALIZED 0x8F9C
1149 #define GL_SINGLE_COLOR 0x81F9
1150 #define GL_SLUMINANCE 0x8C46
1151 #define GL_SLUMINANCE8 0x8C47
1152 #define GL_SLUMINANCE8_ALPHA8 0x8C45
1153 #define GL_SLUMINANCE_ALPHA 0x8C44
1154 #define GL_SMOOTH 0x1D01
```

```
1155 #define GL_SMOOTH_LINE_WIDTH_GRANULARITY 0x0B23
1156 #define GL_SMOOTH_LINE_WIDTH_RANGE 0x0B22
1157 #define GL_SMOOTH_POINT_SIZE_GRANULARITY 0x0B13
1158 #define GL_SMOOTH_POINT_SIZE_RANGE 0x0B12
1159 #define GL_SOURCE0_ALPHA 0x8588
1160 #define GL_SOURCE0_RGB 0x8580
1161 #define GL_SOURCE1_ALPHA 0x8589
1162 #define GL_SOURCE1_RGB 0x8581
1163 #define GL_SOURCE2_ALPHA 0x858A
1164 #define GL_SOURCE2_RGB 0x8582
1165 #define GL_SPECULAR 0x1202
1166 #define GL_SPHERE_MAP 0x2402
1167 #define GL_SPOT_CUTOFF 0x1206
1168 #define GL_SPOT_DIRECTION 0x1204
1169 #define GL_SPOT_EXPONENT 0x1205
1170 #define GL_SRC0_ALPHA 0x8588
1171 #define GL_SRC0_RGB 0x8580
1172 #define GL_SRC1_ALPHA 0x8589
1173 #define GL_SRC1_COLOR 0x88F9
1174 #define GL_SRC1_RGB 0x8581
1175 #define GL_SRC2_ALPHA 0x858A
1176 #define GL_SRC2_RGB 0x8582
1177 #define GL_SRC_ALPHA 0x0302
1178 #define GL_SRC_ALPHA_SATURATE 0x0308
1179 #define GL_SRC_COLOR 0x0300
1180 #define GL_SRGB 0x8C40
1181 #define GL_SRGB8 0x8C41
1182 #define GL_SRGB8_ALPHA8 0x8C43
1183 #define GL_SRGB_ALPHA 0x8C42
1184 #define GL_STACK_OVERFLOW 0x0503
1185 #define GL_STACK_UNDERFLOW 0x0504
1186 #define GL_STATIC_COPY 0x88E6
1187 #define GL_STATIC_DRAW 0x88E4
1188 #define GL_STATIC_READ 0x88E5
1189 #define GL_STENCIL 0x1802
1190 #define GL_STENCIL_ATTACHMENT 0x8D20
1191 #define GL_STENCIL_BACK_FAIL 0x8801
1192 #define GL_STENCIL_BACK_FUNC 0x8800
1193 #define GL_STENCIL_BACK_PASS_DEPTH_FAIL 0x8802
1194 #define GL_STENCIL_BACK_PASS_DEPTH_PASS 0x8803
1195 #define GL_STENCIL_BACK_REF 0x8CA3
1196 #define GL_STENCIL_BACK_VALUE_MASK 0x8CA4
1197 #define GL_STENCIL_BACK_WRITEMASK 0x8CA5
1198 #define GL_STENCIL_BITS 0x0D57
1199 #define GL_STENCIL_BUFFER_BIT 0x00000400
1200 #define GL_STENCIL_CLEAR_VALUE 0x0B91
1201 #define GL_STENCIL_FAIL 0x0B94
1202 #define GL_STENCIL_FUNC 0x0B92
1203 #define GL_STENCIL_INDEX 0x1901
1204 #define GL_STENCIL_INDEX1 0x8D46
1205 #define GL_STENCIL_INDEX16 0x8D49
1206 #define GL_STENCIL_INDEX4 0x8D47
1207 #define GL_STENCIL_INDEX8 0x8D48
1208 #define GL_STENCIL_PASS_DEPTH_FAIL 0x0B95
1209 #define GL_STENCIL_PASS_DEPTH_PASS 0x0B96
1210 #define GL_STENCIL_REF 0x0B97
1211 #define GL_STENCIL_TEST 0x0B90
1212 #define GL_STENCIL_VALUE_MASK 0x0B93
1213 #define GL_STENCIL_WRITEMASK 0x0B98
1214 #define GL_STEREO 0x0C33
1215 #define GL_STREAM_COPY 0x88E2
1216 #define GL_STREAM_DRAW 0x88E0
1217 #define GL_STREAM_READ 0x88E1
1218 #define GL_SUBPIXEL_BITS 0x0D50
1219 #define GL_SUBTRACT 0x84E7
1220 #define GL_SYNC_CONDITION 0x9113
1221 #define GL_SYNC_FENCE 0x9116
1222 #define GL_SYNC_FLAGS 0x9115
1223 #define GL_SYNC_FLUSH_COMMANDS_BIT 0x00000001
1224 #define GL_SYNC_GPU_COMMANDS_COMPLETE 0x9117
1225 #define GL_SYNC_STATUS 0x9114
1226 #define GL_T 0x2001
1227 #define GL_T2F_C3F_V3F 0x2A2A
1228 #define GL_T2F_C4F_N3F_V3F 0x2A2C
1229 #define GL_T2F_C4UB_V3F 0x2A29
1230 #define GL_T2F_N3F_V3F 0x2A2B
1231 #define GL_T2F_V3F 0x2A27
1232 #define GL_T4F_C4F_N3F_V4F 0x2A2D
1233 #define GL_T4F_V4F 0x2A28
1234 #define GL_TEXTURE 0x1702
1235 #define GL_TEXTURE0 0x84C0
1236 #define GL_TEXTURE1 0x84C1
1237 #define GL_TEXTURE10 0x84CA
1238 #define GL_TEXTURE11 0x84CB
1239 #define GL_TEXTURE12 0x84CC
1240 #define GL_TEXTURE13 0x84CD
1241 #define GL_TEXTURE14 0x84CE
```

```
1242 #define GL_TEXTURE15 0x84CF
1243 #define GL_TEXTURE16 0x84D0
1244 #define GL_TEXTURE17 0x84D1
1245 #define GL_TEXTURE18 0x84D2
1246 #define GL_TEXTURE19 0x84D3
1247 #define GL_TEXTURE2 0x84C2
1248 #define GL_TEXTURE20 0x84D4
1249 #define GL_TEXTURE21 0x84D5
1250 #define GL_TEXTURE22 0x84D6
1251 #define GL_TEXTURE23 0x84D7
1252 #define GL_TEXTURE24 0x84D8
1253 #define GL_TEXTURE25 0x84D9
1254 #define GL_TEXTURE26 0x84DA
1255 #define GL_TEXTURE27 0x84DB
1256 #define GL_TEXTURE28 0x84DC
1257 #define GL_TEXTURE29 0x84DD
1258 #define GL_TEXTURE3 0x84C3
1259 #define GL_TEXTURE30 0x84DE
1260 #define GL_TEXTURE31 0x84DF
1261 #define GL_TEXTURE4 0x84C4
1262 #define GL_TEXTURE5 0x84C5
1263 #define GL_TEXTURE6 0x84C6
1264 #define GL_TEXTURE7 0x84C7
1265 #define GL_TEXTURE8 0x84C8
1266 #define GL_TEXTURE9 0x84C9
1267 #define GL_TEXTURE_1D 0x0DE0
1268 #define GL_TEXTURE_1D_ARRAY 0x8C18
1269 #define GL_TEXTURE_2D 0x0DE1
1270 #define GL_TEXTURE_2D_ARRAY 0x8C1A
1271 #define GL_TEXTURE_2D_MULTISAMPLE 0x9100
1272 #define GL_TEXTURE_2D_MULTISAMPLE_ARRAY 0x9102
1273 #define GL_TEXTURE_3D 0x806F
1274 #define GL_TEXTURE_ALPHA_SIZE 0x805F
1275 #define GL_TEXTURE_ALPHA_TYPE 0x8C13
1276 #define GL_TEXTURE_BASE_LEVEL 0x813C
1277 #define GL_TEXTURE_BINDING_1D 0x8068
1278 #define GL_TEXTURE_BINDING_1D_ARRAY 0x8C1C
1279 #define GL_TEXTURE_BINDING_2D 0x8069
1280 #define GL_TEXTURE_BINDING_2D_ARRAY 0x8C1D
1281 #define GL_TEXTURE_BINDING_2D_MULTISAMPLE 0x9104
1282 #define GL_TEXTURE_BINDING_2D_MULTISAMPLE_ARRAY 0x9105
1283 #define GL_TEXTURE_BINDING_3D 0x806A
1284 #define GL_TEXTURE_BINDING_BUFFER 0x8C2C
1285 #define GL_TEXTURE_BINDING_CUBE_MAP 0x8514
1286 #define GL_TEXTURE_BINDING_RECTANGLE 0x84F6
1287 #define GL_TEXTURE_BIT 0x00040000
1288 #define GL_TEXTURE_BLUE_SIZE 0x805E
1289 #define GL_TEXTURE_BLUE_TYPE 0x8C12
1290 #define GL_TEXTURE_BORDER 0x1005
1291 #define GL_TEXTURE_BORDER_COLOR 0x1004
1292 #define GL_TEXTURE_BUFFER 0x8C2A
1293 #define GL_TEXTURE_BUFFER_DATA_STORE_BINDING 0x8C2D
1294 #define GL_TEXTURE_COMPARE_FUNC 0x884D
1295 #define GL_TEXTURE_COMPARE_MODE 0x884C
1296 #define GL_TEXTURE_COMPONENTS 0x1003
1297 #define GL_TEXTURE_COMPRESSED 0x86A1
1298 #define GL_TEXTURE_COMPRESSED_IMAGE_SIZE 0x86A0
1299 #define GL_TEXTURE_COMPRESSION_HINT 0x84EF
1300 #define GL_TEXTURE_COORD_ARRAY 0x8078
1301 #define GL_TEXTURE_COORD_ARRAY_BUFFER_BINDING 0x889A
1302 #define GL_TEXTURE_COORD_ARRAY_POINTER 0x8092
1303 #define GL_TEXTURE_COORD_ARRAY_SIZE 0x8088
1304 #define GL_TEXTURE_COORD_ARRAY_STRIDE 0x808A
1305 #define GL_TEXTURE_COORD_ARRAY_TYPE 0x8089
1306 #define GL_TEXTURE_CUBE_MAP 0x8513
1307 #define GL_TEXTURE_CUBE_MAP_NEGATIVE_X 0x8516
1308 #define GL_TEXTURE_CUBE_MAP_NEGATIVE_Y 0x8518
1309 #define GL_TEXTURE_CUBE_MAP_NEGATIVE_Z 0x851A
1310 #define GL_TEXTURE_CUBE_MAP_POSITIVE_X 0x8515
1311 #define GL_TEXTURE_CUBE_MAP_POSITIVE_Y 0x8517
1312 #define GL_TEXTURE_CUBE_MAP_POSITIVE_Z 0x8519
1313 #define GL_TEXTURE_CUBE_MAP_SEAMLESS 0x884F
1314 #define GL_TEXTURE_DEPTH 0x8071
1315 #define GL_TEXTURE_DEPTH_SIZE 0x884A
1316 #define GL_TEXTURE_DEPTH_TYPE 0x8C16
1317 #define GL_TEXTURE_ENV 0x2300
1318 #define GL_TEXTURE_ENV_COLOR 0x2201
1319 #define GL_TEXTURE_ENV_MODE 0x2200
1320 #define GL_TEXTURE_FILTER_CONTROL 0x8500
1321 #define GL_TEXTURE_FIXED_SAMPLE_LOCATIONS 0x9107
1322 #define GL_TEXTURE_GEN_MODE 0x2500
1323 #define GL_TEXTURE_GEN_Q 0x0C63
1324 #define GL_TEXTURE_GEN_R 0x0C62
1325 #define GL_TEXTURE_GEN_S 0x0C60
1326 #define GL_TEXTURE_GEN_T 0x0C61
1327 #define GL_TEXTURE_GREEN_SIZE 0x805D
1328 #define GL_TEXTURE_GREEN_TYPE 0x8C11
```

```
1329 #define GL_TEXTURE_HEIGHT 0x1001
1330 #define GL_TEXTURE_INTENSITY_SIZE 0x8061
1331 #define GL_TEXTURE_INTENSITY_TYPE 0x8C15
1332 #define GL_TEXTURE_INTERNAL_FORMAT 0x1003
1333 #define GL_TEXTURE_LOD_BIAS 0x8501
1334 #define GL_TEXTURE_LUMINANCE_SIZE 0x8060
1335 #define GL_TEXTURE_LUMINANCE_TYPE 0x8C14
1336 #define GL_TEXTURE_MAG_FILTER 0x2800
1337 #define GL_TEXTURE_MATRIX 0x0BA8
1338 #define GL_TEXTURE_MAX_LEVEL 0x813D
1339 #define GL_TEXTURE_MAX_LOD 0x813B
1340 #define GL_TEXTURE_MIN_FILTER 0x2801
1341 #define GL_TEXTURE_MIN_LOD 0x813A
1342 #define GL_TEXTURE_PRIORITY 0x8066
1343 #define GL_TEXTURE_RECTANGLE 0x84F5
1344 #define GL_TEXTURE_RED_SIZE 0x805C
1345 #define GL_TEXTURE_RED_TYPE 0x8C10
1346 #define GL_TEXTURE_RESIDENT 0x8067
1347 #define GL_TEXTURE_SAMPLES 0x9106
1348 #define GL_TEXTURE_SHARED_SIZE 0x8C3F
1349 #define GL_TEXTURE_STACK_DEPTH 0x0BA5
1350 #define GL_TEXTURE_STENCIL_SIZE 0x88F1
1351 #define GL_TEXTURE_SWIZZLE_A 0x8E45
1352 #define GL_TEXTURE_SWIZZLE_B 0x8E44
1353 #define GL_TEXTURE_SWIZZLE_G 0x8E43
1354 #define GL_TEXTURE_SWIZZLE_R 0x8E42
1355 #define GL_TEXTURE_SWIZZLE_RGBA 0x8E46
1356 #define GL_TEXTURE_WIDTH 0x1000
1357 #define GL_TEXTURE_WRAP_R 0x8072
1358 #define GL_TEXTURE_WRAP_S 0x2802
1359 #define GL_TEXTURE_WRAP_T 0x2803
1360 #define GL_TIMEOUT_EXPIRED 0x911B
1361 #define GL_TIMEOUT_IGNORED 0xFFFFFFFFFFFFFFFF
1362 #define GL_TIMESTAMP 0x8E28
1363 #define GL_TIME_ELAPSED 0x88BF
1364 #define GL_TRANSFORM_BIT 0x00001000
1365 #define GL_TRANSFORM_FEEDBACK_BUFFER 0x8C8E
1366 #define GL_TRANSFORM_FEEDBACK_BUFFER_BINDING 0x8C8F
1367 #define GL_TRANSFORM_FEEDBACK_BUFFER_MODE 0x8C7F
1368 #define GL_TRANSFORM_FEEDBACK_BUFFER_SIZE 0x8C85
1369 #define GL_TRANSFORM_FEEDBACK_BUFFER_START 0x8C84
1370 #define GL_TRANSFORM_FEEDBACK_PRIMITIVES_WRITTEN 0x8C88
1371 #define GL_TRANSFORM_FEEDBACK_VARYINGS 0x8C83
1372 #define GL_TRANSFORM_FEEDBACK_VARYING_MAX_LENGTH 0x8C76
1373 #define GL_TRANSPOSE_COLOR_MATRIX 0x84E6
1374 #define GL_TRANSPOSE_MODELVIEW_MATRIX 0x84E3
1375 #define GL_TRANSPOSE_PROJECTION_MATRIX 0x84E4
1376 #define GL_TRANSPOSE_TEXTURE_MATRIX 0x84E5
1377 #define GL_TRIANGLES 0x0004
1378 #define GL_TRIANGLES_ADJACENCY 0x000C
1379 #define GL_TRIANGLE_FAN 0x0006
1380 #define GL_TRIANGLE_STRIP 0x0005
1381 #define GL_TRIANGLE_STRIP_ADJACENCY 0x000D
1382 #define GL_TRUE 1
1383 #define GL_UNIFORM_ARRAY_STRIDE 0x8A3C
1384 #define GL_UNIFORM_BLOCK_ACTIVE_UNIFORMS 0x8A42
1385 #define GL_UNIFORM_BLOCK_ACTIVE_UNIFORM_INDICES 0x8A43
1386 #define GL_UNIFORM_BLOCK_BINDING 0x8A3F
1387 #define GL_UNIFORM_BLOCK_DATA_SIZE 0x8A40
1388 #define GL_UNIFORM_BLOCK_INDEX 0x8A3A
1389 #define GL_UNIFORM_BLOCK_NAME_LENGTH 0x8A41
1390 #define GL_UNIFORM_BLOCK_REFERENCED_BY_FRAGMENT_SHADER 0x8A46
1391 #define GL_UNIFORM_BLOCK_REFERENCED_BY_GEOMETRY_SHADER 0x8A45
1392 #define GL_UNIFORM_BLOCK_REFERENCED_BY_VERTEX_SHADER 0x8A44
1393 #define GL_UNIFORM_BUFFER 0x8A11
1394 #define GL_UNIFORM_BUFFER_BINDING 0x8A28
1395 #define GL_UNIFORM_BUFFER_OFFSET_ALIGNMENT 0x8A34
1396 #define GL_UNIFORM_BUFFER_SIZE 0x8A2A
1397 #define GL_UNIFORM_BUFFER_START 0x8A29
1398 #define GL_UNIFORM_IS_ROW_MAJOR 0x8A3E
1399 #define GL_UNIFORM_MATRIX_STRIDE 0x8A3D
1400 #define GL_UNIFORM_NAME_LENGTH 0x8A39
1401 #define GL_UNIFORM_OFFSET 0x8A3B
1402 #define GL_UNIFORM_SIZE 0x8A38
1403 #define GL_UNIFORM_TYPE 0x8A37
1404 #define GL_UNKNOWN_CONTEXT_RESET_ARB 0x8255
1405 #define GL_UNPACK_ALIGNMENT 0x0CF5
1406 #define GL_UNPACK_IMAGE_HEIGHT 0x806E
1407 #define GL_UNPACK_LSB_FIRST 0x0CF1
1408 #define GL_UNPACK_ROW_LENGTH 0x0CF2
1409 #define GL_UNPACK_SKIP_IMAGES 0x806D
1410 #define GL_UNPACK_SKIP_PIXELS 0x0CF4
1411 #define GL_UNPACK_SKIP_ROWS 0x0CF3
1412 #define GL_UNPACK_SWAP_BYTES 0x0CF0
1413 #define GL_UNSIGNALED 0x9118
1414 #define GL_UNSIGNED_BYTE 0x1401
1415 #define GL_UNSIGNED_BYTE_2_3_3_REV 0x8362
```

```
1416 #define GL_UNSIGNED_BYTE_3_3_2 0x8032
1417 #define GL_UNSIGNED_INT 0x1405
1418 #define GL_UNSIGNED_INT_10F_11F_11F_REV 0x8C3B
1419 #define GL_UNSIGNED_INT_10_10_10_2 0x8036
1420 #define GL_UNSIGNED_INT_24_8 0x84FA
1421 #define GL_UNSIGNED_INT_2_10_10_10_REV 0x8368
1422 #define GL_UNSIGNED_INT_5_9_9_9_REV 0x8C3E
1423 #define GL_UNSIGNED_INT_8_8_8_8 0x8035
1424 #define GL_UNSIGNED_INT_8_8_8_8_REV 0x8367
1425 #define GL_UNSIGNED_INT_SAMPLER_1D 0x8DD1
1426 #define GL_UNSIGNED_INT_SAMPLER_1D_ARRAY 0x8DD6
1427 #define GL_UNSIGNED_INT_SAMPLER_2D 0x8DD2
1428 #define GL_UNSIGNED_INT_SAMPLER_2D_ARRAY 0x8DD7
1429 #define GL_UNSIGNED_INT_SAMPLER_2D_MULTISAMPLE 0x910A
1430 #define GL_UNSIGNED_INT_SAMPLER_2D_MULTISAMPLE_ARRAY 0x910D
1431 #define GL_UNSIGNED_INT_SAMPLER_2D_RECT 0x8DD5
1432 #define GL_UNSIGNED_INT_SAMPLER_3D 0x8DD3
1433 #define GL_UNSIGNED_INT_SAMPLER_BUFFER 0x8DD8
1434 #define GL_UNSIGNED_INT_SAMPLER_CUBE 0x8DD4
1435 #define GL_UNSIGNED_INT_VEC2 0x8DC6
1436 #define GL_UNSIGNED_INT_VEC3 0x8DC7
1437 #define GL_UNSIGNED_INT_VEC4 0x8DC8
1438 #define GL_UNSIGNED_NORMALIZED 0x8C17
1439 #define GL_UNSIGNED_SHORT 0x1403
1440 #define GL_UNSIGNED_SHORT_1_5_5_5_REV 0x8366
1441 #define GL_UNSIGNED_SHORT_4_4_4_4 0x8033
1442 #define GL_UNSIGNED_SHORT_4_4_4_4_REV 0x8365
1443 #define GL_UNSIGNED_SHORT_5_5_5_1 0x8034
1444 #define GL_UNSIGNED_SHORT_5_6_5 0x8363
1445 #define GL_UNSIGNED_SHORT_5_6_5_REV 0x8364
1446 #define GL_UPPER_LEFT 0x8CA2
1447 #define GL_V2F 0x2A20
1448 #define GL_V3F 0x2A21
1449 #define GL_VALIDATE_STATUS 0x8B83
1450 #define GL_VENDOR 0x1F00
1451 #define GL_VERSION 0x1F02
1452 #define GL_VERTEX_ARRAY 0x8074
1453 #define GL_VERTEX_ARRAY_BINDING 0x85B5
1454 #define GL_VERTEX_ARRAY_BUFFER_BINDING 0x8896
1455 #define GL_VERTEX_ARRAY_POINTER 0x808E
1456 #define GL_VERTEX_ARRAY_SIZE 0x807A
1457 #define GL_VERTEX_ARRAY_STRIDE 0x807C
1458 #define GL_VERTEX_ARRAY_TYPE 0x807B
1459 #define GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING 0x889F
1460 #define GL_VERTEX_ATTRIB_ARRAY_DIVISOR 0x88FE
1461 #define GL_VERTEX_ATTRIB_ARRAY_ENABLED 0x8622
1462 #define GL_VERTEX_ATTRIB_ARRAY_INTEGER 0x88FD
1463 #define GL_VERTEX_ATTRIB_ARRAY_NORMALIZED 0x886A
1464 #define GL_VERTEX_ATTRIB_ARRAY_POINTER 0x8645
1465 #define GL_VERTEX_ATTRIB_ARRAY_SIZE 0x8623
1466 #define GL_VERTEX_ATTRIB_ARRAY_STRIDE 0x8624
1467 #define GL_VERTEX_ATTRIB_ARRAY_TYPE 0x8625
1468 #define GL_VERTEX_PROGRAM_POINT_SIZE 0x8642
1469 #define GL_VERTEX_PROGRAM_TWO_SIDE 0x8643
1470 #define GL_VERTEX_SHADER 0x8B31
1471 #define GL_VIEWPORT 0x0BA2
1472 #define GL_VIEWPORT_BIT 0x00000800
1473 #define GL_WAIT_FAILED 0x911D
1474 #define GL_WEIGHT_ARRAY_BUFFER_BINDING 0x889E
1475 #define GL_WRITE_ONLY 0x88B9
1476 #define GL_XOR 0x1506
1477 #define GL_ZERO 0
1478 #define GL_ZOOM_X 0x0D16
1479 #define GL_ZOOM_Y 0x0D17
1480
1481
1482 #ifndef __khrplatform_h_
1483 #define __khrplatform_h_
1484
1485 /*
1486 ** Copyright (c) 2008-2018 The Khronos Group Inc.
1487 **
1488 ** Permission is hereby granted, free of charge, to any person obtaining a
1489 ** copy of this software and/or associated documentation files (the
1490 ** "Materials"), to deal in the Materials without restriction, including
1491 ** without limitation the rights to use, copy, modify, merge, publish,
1492 ** distribute, sublicense, and/or sell copies of the Materials, and to
1493 ** permit persons to whom the Materials are furnished to do so, subject to
1494 ** the following conditions:
1495 **
1496 ** The above copyright notice and this permission notice shall be included
1497 ** in all copies or substantial portions of the Materials.
1498 **
1499 ** THE MATERIALS ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
1500 ** EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
1501 ** MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
1502 ** IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
```

```
1503 ** CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
1504 ** TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
1505 ** MATERIALS OR THE USE OR OTHER DEALINGS IN THE MATERIALS.
1506 */
1507
1508 /* Khronos platform-specific types and definitions.
1509  *
1510  * The master copy of khrplatform.h is maintained in the Khronos EGL
1511  * Registry repository at https://github.com/KhronosGroup/EGL-Registry
1512  * The last semantic modification to khrplatform.h was at commit ID:
1513  *      67a3e0864c2d75ea5287b9f3d2eb74a745936692
1514  *
1515  * Adopters may modify this file to suit their platform. Adopters are
1516  * encouraged to submit platform specific modifications to the Khronos
1517  * group so that they can be included in future versions of this file.
1518  * Please submit changes by filing pull requests or issues on
1519  * the EGL Registry repository linked above.
1520  *
1521  *
1522  * See the Implementer's Guidelines for information about where this file
1523  * should be located on your system and for more details of its use:
1524  *    http://www.khronos.org/registry/implementers_guide.pdf
1525  *
1526  * This file should be included as
1527  *        #include <KHR/khrplatform.h>
1528  * by Khronos client API header files that use its types and defines.
1529  *
1530  * The types in khrplatform.h should only be used to define API-specific types.
1531  *
1532  * Types defined in khrplatform.h:
1533  *    khronos_int8_t              signed   8  bit
1534  *    khronos_uint8_t             unsigned 8  bit
1535  *    khronos_int16_t             signed   16 bit
1536  *    khronos_uint16_t            unsigned 16 bit
1537  *    khronos_int32_t             signed   32 bit
1538  *    khronos_uint32_t            unsigned 32 bit
1539  *    khronos_int64_t             signed   64 bit
1540  *    khronos_uint64_t            unsigned 64 bit
1541  *    khronos_intptr_t            signed   same number of bits as a pointer
1542  *    khronos_uintptr_t           unsigned same number of bits as a pointer
1543  *    khronos_ssize_t             signed   size
1544  *    khronos_usize_t             unsigned size
1545  *    khronos_float_t             signed   32 bit floating point
1546  *    khronos_time_ns_t           unsigned 64 bit time in nanoseconds
1547  *    khronos_utime_nanoseconds_t unsigned time interval or absolute time in
1548  *                                         nanoseconds
1549  *    khronos_stime_nanoseconds_t signed time interval in nanoseconds
1550  *    khronos_boolean_enum_t      enumerated boolean type. This should
1551  *      only be used as a base type when a client API's boolean type is
1552  *      an enum. Client APIs which use an integer or other type for
1553  *      booleans cannot use this as the base type for their boolean.
1554  *
1555  * Tokens defined in khrplatform.h:
1556  *
1557  *    KHRONOS_FALSE, KHRONOS_TRUE Enumerated boolean false/true values.
1558  *
1559  *    KHRONOS_SUPPORT_INT64 is 1 if 64 bit integers are supported; otherwise 0.
1560  *    KHRONOS_SUPPORT_FLOAT is 1 if floats are supported; otherwise 0.
1561  *
1562  * Calling convention macros defined in this file:
1563  *    KHRONOS_APICALL
1564  *    KHRONOS_GLAD_API_PTR
1565  *    KHRONOS_APIATTRIBUTES
1566  *
1567  * These may be used in function prototypes as:
1568  *
1569  *      KHRONOS_APICALL void KHRONOS_GLAD_API_PTR funcname(
1570  *                                  int arg1,
1571  *                                  int arg2) KHRONOS_APIATTRIBUTES;
1572  */
1573
1574 #if defined(__SCITECH_SNAP__) && !defined(KHRONOS_STATIC)
1575 #   define KHRONOS_STATIC 1
1576 #endif
1577
1578 /*-------------------------------------------------------------------------
1579  * Definition of KHRONOS_APICALL
1580  *-------------------------------------------------------------------------
1581  * This precedes the return type of the function in the function prototype.
1582  */
1583 #if defined(KHRONOS_STATIC)
1584     /* If the preprocessor constant KHRONOS_STATIC is defined, make the
1585      * header compatible with static linking. */
1586 #   define KHRONOS_APICALL
1587 #elif defined(_WIN32)
1588 #   define KHRONOS_APICALL __declspec(dllimport)
1589 #elif defined (__SYMBIAN32__)
```

```
1590 #   define KHRONOS_APICALL IMPORT_C
1591 #elif defined(__ANDROID__)
1592 #   define KHRONOS_APICALL __attribute__((visibility("default")))
1593 #else
1594 #   define KHRONOS_APICALL
1595 #endif
1596
1597 /*-------------------------------------------------------------------------
1598  * Definition of KHRONOS_GLAD_API_PTR
1599  *-------------------------------------------------------------------------
1600  * This follows the return type of the function  and precedes the function
1601  * name in the function prototype.
1602  */
1603 #if defined(_WIN32) && !defined(_WIN32_WCE) && !defined(__SCITECH_SNAP__)
1604     /* Win32 but not WinCE */
1605 #   define KHRONOS_GLAD_API_PTR __stdcall
1606 #else
1607 #   define KHRONOS_GLAD_API_PTR
1608 #endif
1609
1610 /*-------------------------------------------------------------------------
1611  * Definition of KHRONOS_APIATTRIBUTES
1612  *-------------------------------------------------------------------------
1613  * This follows the closing parenthesis of the function prototype arguments.
1614  */
1615 #if defined (__ARMCC_2__)
1616 #define KHRONOS_APIATTRIBUTES __softfp
1617 #else
1618 #define KHRONOS_APIATTRIBUTES
1619 #endif
1620
1621 /*-------------------------------------------------------------------------
1622  * basic type definitions
1623  *-----------------------------------------------------------------------*/
1624 #if (defined(__STDC_VERSION__) && __STDC_VERSION__ >= 199901L) || defined(__GNUC__) || defined(__SCO__)
      || defined(__USLC__)
1625
1626
1627 /*
1628  * Using <stdint.h>
1629  */
1630 #include <stdint.h>
1631 typedef int32_t                 khronos_int32_t;
1632 typedef uint32_t                khronos_uint32_t;
1633 typedef int64_t                 khronos_int64_t;
1634 typedef uint64_t                khronos_uint64_t;
1635 #define KHRONOS_SUPPORT_INT64   1
1636 #define KHRONOS_SUPPORT_FLOAT   1
1637
1638 #elif defined(__VMS ) || defined(__sgi)
1639
1640 /*
1641  * Using <inttypes.h>
1642  */
1643 #include <inttypes.h>
1644 typedef int32_t                 khronos_int32_t;
1645 typedef uint32_t                khronos_uint32_t;
1646 typedef int64_t                 khronos_int64_t;
1647 typedef uint64_t                khronos_uint64_t;
1648 #define KHRONOS_SUPPORT_INT64   1
1649 #define KHRONOS_SUPPORT_FLOAT   1
1650
1651 #elif defined(_WIN32) && !defined(__SCITECH_SNAP__)
1652
1653 /*
1654  * Win32
1655  */
1656 typedef __int32                 khronos_int32_t;
1657 typedef unsigned __int32        khronos_uint32_t;
1658 typedef __int64                 khronos_int64_t;
1659 typedef unsigned __int64        khronos_uint64_t;
1660 #define KHRONOS_SUPPORT_INT64   1
1661 #define KHRONOS_SUPPORT_FLOAT   1
1662
1663 #elif defined(__sun__) || defined(__digital__)
1664
1665 /*
1666  * Sun or Digital
1667  */
1668 typedef int                     khronos_int32_t;
1669 typedef unsigned int            khronos_uint32_t;
1670 #if defined(__arch64__) || defined(_LP64)
1671 typedef long int                khronos_int64_t;
1672 typedef unsigned long int       khronos_uint64_t;
1673 #else
1674 typedef long long int           khronos_int64_t;
1675 typedef unsigned long long int  khronos_uint64_t;
```

```
1676 #endif /* __arch64__ */
1677 #define KHRONOS_SUPPORT_INT64   1
1678 #define KHRONOS_SUPPORT_FLOAT   1
1679
1680 #elif 0
1681
1682 /*
1683  * Hypothetical platform with no float or int64 support
1684  */
1685 typedef int                     khronos_int32_t;
1686 typedef unsigned int            khronos_uint32_t;
1687 #define KHRONOS_SUPPORT_INT64   0
1688 #define KHRONOS_SUPPORT_FLOAT   0
1689
1690 #else
1691
1692 /*
1693  * Generic fallback
1694  */
1695 #include <stdint.h>
1696 typedef int32_t                 khronos_int32_t;
1697 typedef uint32_t                khronos_uint32_t;
1698 typedef int64_t                 khronos_int64_t;
1699 typedef uint64_t                khronos_uint64_t;
1700 #define KHRONOS_SUPPORT_INT64   1
1701 #define KHRONOS_SUPPORT_FLOAT   1
1702
1703 #endif
1704
1705
1706 /*
1707  * Types that are (so far) the same on all platforms
1708  */
1709 typedef signed   char          khronos_int8_t;
1710 typedef unsigned char          khronos_uint8_t;
1711 typedef signed   short int     khronos_int16_t;
1712 typedef unsigned short int     khronos_uint16_t;
1713
1714 /*
1715  * Types that differ between LLP64 and LP64 architectures - in LLP64,
1716  * pointers are 64 bits, but 'long' is still 32 bits. Win64 appears
1717  * to be the only LLP64 architecture in current use.
1718  */
1719 #ifdef _WIN64
1720 typedef signed   long long int khronos_intptr_t;
1721 typedef unsigned long long int khronos_uintptr_t;
1722 typedef signed   long long int khronos_ssize_t;
1723 typedef unsigned long long int khronos_usize_t;
1724 #else
1725 typedef signed   long  int     khronos_intptr_t;
1726 typedef unsigned long  int     khronos_uintptr_t;
1727 typedef signed   long  int     khronos_ssize_t;
1728 typedef unsigned long  int     khronos_usize_t;
1729 #endif
1730
1731 #if KHRONOS_SUPPORT_FLOAT
1732 /*
1733  * Float type
1734  */
1735 typedef          float         khronos_float_t;
1736 #endif
1737
1738 #if KHRONOS_SUPPORT_INT64
1739 /* Time types
1740  *
1741  * These types can be used to represent a time interval in nanoseconds or
1742  * an absolute Unadjusted System Time.  Unadjusted System Time is the number
1743  * of nanoseconds since some arbitrary system event (e.g. since the last
1744  * time the system booted).  The Unadjusted System Time is an unsigned
1745  * 64 bit value that wraps back to 0 every 584 years.  Time intervals
1746  * may be either signed or unsigned.
1747  */
1748 typedef khronos_uint64_t       khronos_utime_nanoseconds_t;
1749 typedef khronos_int64_t        khronos_stime_nanoseconds_t;
1750 #endif
1751
1752 /*
1753  * Dummy value used to pad enum types to 32 bits.
1754  */
1755 #ifndef KHRONOS_MAX_ENUM
1756 #define KHRONOS_MAX_ENUM 0x7FFFFFFF
1757 #endif
1758
1759 /*
1760  * Enumerated boolean type
1761  *
1762  * Values other than zero should be considered to be true.  Therefore
```

```
1763  * comparisons should not be made against KHRONOS_TRUE.
1764  */
1765 typedef enum {
1766     KHRONOS_FALSE = 0,
1767     KHRONOS_TRUE  = 1,
1768     KHRONOS_BOOLEAN_ENUM_FORCE_SIZE = KHRONOS_MAX_ENUM
1769 } khronos_boolean_enum_t;
1770
1771 #endif /* __khrplatform_h_ */
1772
1773 typedef unsigned int GLenum;
1774
1775 typedef unsigned char GLboolean;
1776
1777 typedef unsigned int GLbitfield;
1778
1779 typedef void GLvoid;
1780
1781 typedef khronos_int8_t GLbyte;
1782
1783 typedef khronos_uint8_t GLubyte;
1784
1785 typedef khronos_int16_t GLshort;
1786
1787 typedef khronos_uint16_t GLushort;
1788
1789 typedef int GLint;
1790
1791 typedef unsigned int GLuint;
1792
1793 typedef khronos_int32_t GLclampx;
1794
1795 typedef int GLsizei;
1796
1797 typedef khronos_float_t GLfloat;
1798
1799 typedef khronos_float_t GLclampf;
1800
1801 typedef double GLdouble;
1802
1803 typedef double GLclampd;
1804
1805 typedef void *GLeglClientBufferEXT;
1806
1807 typedef void *GLeglImageOES;
1808
1809 typedef char GLchar;
1810
1811 typedef char GLcharARB;
1812
1813 #ifdef __APPLE__
1814 typedef void *GLhandleARB;
1815 #else
1816 typedef unsigned int GLhandleARB;
1817 #endif
1818
1819 typedef khronos_uint16_t GLhalf;
1820
1821 typedef khronos_uint16_t GLhalfARB;
1822
1823 typedef khronos_int32_t GLfixed;
1824
1825 #if defined(__ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__) &&
       (__ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__ > 1060)
1826 typedef khronos_intptr_t GLintptr;
1827 #else
1828 typedef khronos_intptr_t GLintptr;
1829 #endif
1830
1831 #if defined(__ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__) &&
       (__ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__ > 1060)
1832 typedef khronos_intptr_t GLintptrARB;
1833 #else
1834 typedef khronos_intptr_t GLintptrARB;
1835 #endif
1836
1837 #if defined(__ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__) &&
       (__ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__ > 1060)
1838 typedef khronos_ssize_t GLsizeiptr;
1839 #else
1840 typedef khronos_ssize_t GLsizeiptr;
1841 #endif
1842
1843 #if defined(__ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__) &&
       (__ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__ > 1060)
1844 typedef khronos_ssize_t GLsizeiptrARB;
1845 #else
```

```
1846 typedef khronos_ssize_t GLsizeiptrARB;
1847 #endif
1848
1849 typedef khronos_int64_t GLint64;
1850
1851 typedef khronos_int64_t GLint64EXT;
1852
1853 typedef khronos_uint64_t GLuint64;
1854
1855 typedef khronos_uint64_t GLuint64EXT;
1856
1857 typedef struct __GLsync *GLsync;
1858
1859 struct _cl_context;
1860
1861 struct _cl_event;
1862
1863 typedef void (GLAD_API_PTR *GLDEBUGPROC)(GLenum source,GLenum type,GLuint id,GLenum severity,GLsizei
      length,const GLchar *message,const void *userParam);
1864
1865 typedef void (GLAD_API_PTR *GLDEBUGPROCARB)(GLenum source,GLenum type,GLuint id,GLenum severity,GLsizei
      length,const GLchar *message,const void *userParam);
1866
1867 typedef void (GLAD_API_PTR *GLDEBUGPROCKHR)(GLenum source,GLenum type,GLuint id,GLenum severity,GLsizei
      length,const GLchar *message,const void *userParam);
1868
1869 typedef void (GLAD_API_PTR *GLDEBUGPROCAMD)(GLuint id,GLenum category,GLenum severity,GLsizei
      length,const GLchar *message,void *userParam);
1870
1871 typedef unsigned short GLhalfNV;
1872
1873 typedef GLintptr GLvdpauSurfaceNV;
1874
1875 typedef void (GLAD_API_PTR *GLVULKANPROCNV)(void);
1876
1877
1878
1879 #define GL_VERSION_1_0 1
1880 GLAD_API_CALL int GLAD_GL_VERSION_1_0;
1881 #define GL_VERSION_1_1 1
1882 GLAD_API_CALL int GLAD_GL_VERSION_1_1;
1883 #define GL_VERSION_1_2 1
1884 GLAD_API_CALL int GLAD_GL_VERSION_1_2;
1885 #define GL_VERSION_1_3 1
1886 GLAD_API_CALL int GLAD_GL_VERSION_1_3;
1887 #define GL_VERSION_1_4 1
1888 GLAD_API_CALL int GLAD_GL_VERSION_1_4;
1889 #define GL_VERSION_1_5 1
1890 GLAD_API_CALL int GLAD_GL_VERSION_1_5;
1891 #define GL_VERSION_2_0 1
1892 GLAD_API_CALL int GLAD_GL_VERSION_2_0;
1893 #define GL_VERSION_2_1 1
1894 GLAD_API_CALL int GLAD_GL_VERSION_2_1;
1895 #define GL_VERSION_3_0 1
1896 GLAD_API_CALL int GLAD_GL_VERSION_3_0;
1897 #define GL_VERSION_3_1 1
1898 GLAD_API_CALL int GLAD_GL_VERSION_3_1;
1899 #define GL_VERSION_3_2 1
1900 GLAD_API_CALL int GLAD_GL_VERSION_3_2;
1901 #define GL_VERSION_3_3 1
1902 GLAD_API_CALL int GLAD_GL_VERSION_3_3;
1903 #define GL_ARB_multisample 1
1904 GLAD_API_CALL int GLAD_GL_ARB_multisample;
1905 #define GL_ARB_robustness 1
1906 GLAD_API_CALL int GLAD_GL_ARB_robustness;
1907 #define GL_KHR_debug 1
1908 GLAD_API_CALL int GLAD_GL_KHR_debug;
1909
1910
1911 typedef void (GLAD_API_PTR *PFNGLACCUMPROC)(GLenum op, GLfloat value);
1912 typedef void (GLAD_API_PTR *PFNGLACTIVETEXTUREPROC)(GLenum texture);
1913 typedef void (GLAD_API_PTR *PFNGLALPHAFUNCPROC)(GLenum func, GLfloat ref);
1914 typedef GLboolean (GLAD_API_PTR *PFNGLARETEXTURESRESIDENTPROC)(GLsizei n, const GLuint * textures,
      GLboolean * residences);
1915 typedef void (GLAD_API_PTR *PFNGLARRAYELEMENTPROC)(GLint i);
1916 typedef void (GLAD_API_PTR *PFNGLATTACHSHADERPROC)(GLuint program, GLuint shader);
1917 typedef void (GLAD_API_PTR *PFNGLBEGINPROC)(GLenum mode);
1918 typedef void (GLAD_API_PTR *PFNGLBEGINCONDITIONALRENDERPROC)(GLuint id, GLenum mode);
1919 typedef void (GLAD_API_PTR *PFNGLBEGINQUERYPROC)(GLenum target, GLuint id);
1920 typedef void (GLAD_API_PTR *PFNGLBEGINTRANSFORMFEEDBACKPROC)(GLenum primitiveMode);
1921 typedef void (GLAD_API_PTR *PFNGLBINDATTRIBLOCATIONPROC)(GLuint program, GLuint index, const GLchar *
      name);
1922 typedef void (GLAD_API_PTR *PFNGLBINDBUFFERPROC)(GLenum target, GLuint buffer);
1923 typedef void (GLAD_API_PTR *PFNGLBINDBUFFERBASEPROC)(GLenum target, GLuint index, GLuint buffer);
1924 typedef void (GLAD_API_PTR *PFNGLBINDBUFFERRANGEPROC)(GLenum target, GLuint index, GLuint buffer,
      GLintptr offset, GLsizeiptr size);
1925 typedef void (GLAD_API_PTR *PFNGLBINDFRAGDATALOCATIONPROC)(GLuint program, GLuint color, const GLchar *
```

```
    name);
1926 typedef void (GLAD_API_PTR *PFNGLBINDFRAGDATALOCATIONINDEXEDPROC)(GLuint program, GLuint colorNumber,
    GLuint index, const GLchar * name);
1927 typedef void (GLAD_API_PTR *PFNGLBINDFRAMEBUFFERPROC)(GLenum target, GLuint framebuffer);
1928 typedef void (GLAD_API_PTR *PFNGLBINDRENDERBUFFERPROC)(GLenum target, GLuint renderbuffer);
1929 typedef void (GLAD_API_PTR *PFNGLBINDSAMPLERPROC)(GLuint unit, GLuint sampler);
1930 typedef void (GLAD_API_PTR *PFNGLBINDTEXTUREPROC)(GLenum target, GLuint texture);
1931 typedef void (GLAD_API_PTR *PFNGLBINDVERTEXARRAYPROC)(GLuint array);
1932 typedef void (GLAD_API_PTR *PFNGLBITMAPPROC)(GLsizei width, GLsizei height, GLfloat xorig, GLfloat
    yorig, GLfloat xmove, GLfloat ymove, const GLubyte * bitmap);
1933 typedef void (GLAD_API_PTR *PFNGLBLENDCOLORPROC)(GLfloat red, GLfloat green, GLfloat blue, GLfloat
    alpha);
1934 typedef void (GLAD_API_PTR *PFNGLBLENDEQUATIONPROC)(GLenum mode);
1935 typedef void (GLAD_API_PTR *PFNGLBLENDEQUATIONSEPARATEPROC)(GLenum modeRGB, GLenum modeAlpha);
1936 typedef void (GLAD_API_PTR *PFNGLBLENDFUNCPROC)(GLenum sfactor, GLenum dfactor);
1937 typedef void (GLAD_API_PTR *PFNGLBLENDFUNCSEPARATEPROC)(GLenum sfactorRGB, GLenum dfactorRGB, GLenum
    sfactorAlpha, GLenum dfactorAlpha);
1938 typedef void (GLAD_API_PTR *PFNGLBLITFRAMEBUFFERPROC)(GLint srcX0, GLint srcY0, GLint srcX1, GLint
    srcY1, GLint dstX0, GLint dstY0, GLint dstX1, GLint dstY1, GLbitfield mask, GLenum filter);
1939 typedef void (GLAD_API_PTR *PFNGLBUFFERDATAPROC)(GLenum target, GLsizeiptr size, const void * data,
    GLenum usage);
1940 typedef void (GLAD_API_PTR *PFNGLBUFFERSUBDATAPROC)(GLenum target, GLintptr offset, GLsizeiptr size,
    const void * data);
1941 typedef void (GLAD_API_PTR *PFNGLCALLLISTPROC)(GLuint list);
1942 typedef void (GLAD_API_PTR *PFNGLCALLLISTSPROC)(GLsizei n, GLenum type, const void * lists);
1943 typedef GLenum (GLAD_API_PTR *PFNGLCHECKFRAMEBUFFERSTATUSPROC)(GLenum target);
1944 typedef void (GLAD_API_PTR *PFNGLCLAMPCOLORPROC)(GLenum target, GLenum clamp);
1945 typedef void (GLAD_API_PTR *PFNGLCLEARPROC)(GLbitfield mask);
1946 typedef void (GLAD_API_PTR *PFNGLCLEARACCUMPROC)(GLfloat red, GLfloat green, GLfloat blue, GLfloat
    alpha);
1947 typedef void (GLAD_API_PTR *PFNGLCLEARBUFFERFIPROC)(GLenum buffer, GLint drawbuffer, GLfloat depth,
    GLint stencil);
1948 typedef void (GLAD_API_PTR *PFNGLCLEARBUFFERFVPROC)(GLenum buffer, GLint drawbuffer, const GLfloat *
    value);
1949 typedef void (GLAD_API_PTR *PFNGLCLEARBUFFERIVPROC)(GLenum buffer, GLint drawbuffer, const GLint *
    value);
1950 typedef void (GLAD_API_PTR *PFNGLCLEARBUFFERUIVPROC)(GLenum buffer, GLint drawbuffer, const GLuint *
    value);
1951 typedef void (GLAD_API_PTR *PFNGLCLEARCOLORPROC)(GLfloat red, GLfloat green, GLfloat blue, GLfloat
    alpha);
1952 typedef void (GLAD_API_PTR *PFNGLCLEARDEPTHPROC)(GLdouble depth);
1953 typedef void (GLAD_API_PTR *PFNGLCLEARINDEXPROC)(GLfloat c);
1954 typedef void (GLAD_API_PTR *PFNGLCLEARSTENCILPROC)(GLint s);
1955 typedef void (GLAD_API_PTR *PFNGLCLIENTACTIVETEXTUREPROC)(GLenum texture);
1956 typedef GLenum (GLAD_API_PTR *PFNGLCLIENTWAITSYNCPROC)(GLsync sync, GLbitfield flags, GLuint64
    timeout);
1957 typedef void (GLAD_API_PTR *PFNGLCLIPPLANEPROC)(GLenum plane, const GLdouble * equation);
1958 typedef void (GLAD_API_PTR *PFNGLCOLOR3BPROC)(GLbyte red, GLbyte green, GLbyte blue);
1959 typedef void (GLAD_API_PTR *PFNGLCOLOR3BVPROC)(const GLbyte * v);
1960 typedef void (GLAD_API_PTR *PFNGLCOLOR3DPROC)(GLdouble red, GLdouble green, GLdouble blue);
1961 typedef void (GLAD_API_PTR *PFNGLCOLOR3DVPROC)(const GLdouble * v);
1962 typedef void (GLAD_API_PTR *PFNGLCOLOR3FPROC)(GLfloat red, GLfloat green, GLfloat blue);
1963 typedef void (GLAD_API_PTR *PFNGLCOLOR3FVPROC)(const GLfloat * v);
1964 typedef void (GLAD_API_PTR *PFNGLCOLOR3IPROC)(GLint red, GLint green, GLint blue);
1965 typedef void (GLAD_API_PTR *PFNGLCOLOR3IVPROC)(const GLint * v);
1966 typedef void (GLAD_API_PTR *PFNGLCOLOR3SPROC)(GLshort red, GLshort green, GLshort blue);
1967 typedef void (GLAD_API_PTR *PFNGLCOLOR3SVPROC)(const GLshort * v);
1968 typedef void (GLAD_API_PTR *PFNGLCOLOR3UBPROC)(GLubyte red, GLubyte green, GLubyte blue);
1969 typedef void (GLAD_API_PTR *PFNGLCOLOR3UBVPROC)(const GLubyte * v);
1970 typedef void (GLAD_API_PTR *PFNGLCOLOR3UIPROC)(GLuint red, GLuint green, GLuint blue);
1971 typedef void (GLAD_API_PTR *PFNGLCOLOR3UIVPROC)(const GLuint * v);
1972 typedef void (GLAD_API_PTR *PFNGLCOLOR3USPROC)(GLushort red, GLushort green, GLushort blue);
1973 typedef void (GLAD_API_PTR *PFNGLCOLOR3USVPROC)(const GLushort * v);
1974 typedef void (GLAD_API_PTR *PFNGLCOLOR4BPROC)(GLbyte red, GLbyte green, GLbyte blue, GLbyte alpha);
1975 typedef void (GLAD_API_PTR *PFNGLCOLOR4BVPROC)(const GLbyte * v);
1976 typedef void (GLAD_API_PTR *PFNGLCOLOR4DPROC)(GLdouble red, GLdouble green, GLdouble blue, GLdouble
    alpha);
1977 typedef void (GLAD_API_PTR *PFNGLCOLOR4DVPROC)(const GLdouble * v);
1978 typedef void (GLAD_API_PTR *PFNGLCOLOR4FPROC)(GLfloat red, GLfloat green, GLfloat blue, GLfloat alpha);
1979 typedef void (GLAD_API_PTR *PFNGLCOLOR4FVPROC)(const GLfloat * v);
1980 typedef void (GLAD_API_PTR *PFNGLCOLOR4IPROC)(GLint red, GLint green, GLint blue, GLint alpha);
1981 typedef void (GLAD_API_PTR *PFNGLCOLOR4IVPROC)(const GLint * v);
1982 typedef void (GLAD_API_PTR *PFNGLCOLOR4SPROC)(GLshort red, GLshort green, GLshort blue, GLshort alpha);
1983 typedef void (GLAD_API_PTR *PFNGLCOLOR4SVPROC)(const GLshort * v);
1984 typedef void (GLAD_API_PTR *PFNGLCOLOR4UBPROC)(GLubyte red, GLubyte green, GLubyte blue, GLubyte
    alpha);
1985 typedef void (GLAD_API_PTR *PFNGLCOLOR4UBVPROC)(const GLubyte * v);
1986 typedef void (GLAD_API_PTR *PFNGLCOLOR4UIPROC)(GLuint red, GLuint green, GLuint blue, GLuint alpha);
1987 typedef void (GLAD_API_PTR *PFNGLCOLOR4UIVPROC)(const GLuint * v);
1988 typedef void (GLAD_API_PTR *PFNGLCOLOR4USPROC)(GLushort red, GLushort green, GLushort blue, GLushort
    alpha);
1989 typedef void (GLAD_API_PTR *PFNGLCOLOR4USVPROC)(const GLushort * v);
1990 typedef void (GLAD_API_PTR *PFNGLCOLORMASKPROC)(GLboolean red, GLboolean green, GLboolean blue,
    GLboolean alpha);
1991 typedef void (GLAD_API_PTR *PFNGLCOLORMASKIPROC)(GLuint index, GLboolean r, GLboolean g, GLboolean b,
    GLboolean a);
1992 typedef void (GLAD_API_PTR *PFNGLCOLORMATERIALPROC)(GLenum face, GLenum mode);
```

```
1993 typedef void (GLAD_API_PTR *PFNGLCOLORP3UIPROC)(GLenum type, GLuint color);
1994 typedef void (GLAD_API_PTR *PFNGLCOLORP3UIVPROC)(GLenum type, const GLuint * color);
1995 typedef void (GLAD_API_PTR *PFNGLCOLORP4UIPROC)(GLenum type, GLuint color);
1996 typedef void (GLAD_API_PTR *PFNGLCOLORP4UIVPROC)(GLenum type, const GLuint * color);
1997 typedef void (GLAD_API_PTR *PFNGLCOLORPOINTERPROC)(GLint size, GLenum type, GLsizei stride, const void
       * pointer);
1998 typedef void (GLAD_API_PTR *PFNGLCOMPILESHADERPROC)(GLuint shader);
1999 typedef void (GLAD_API_PTR *PFNGLCOMPRESSEDTEXIMAGE1DPROC)(GLenum target, GLint level, GLenum
       internalformat, GLsizei width, GLint border, GLsizei imageSize, const void * data);
2000 typedef void (GLAD_API_PTR *PFNGLCOMPRESSEDTEXIMAGE2DPROC)(GLenum target, GLint level, GLenum
       internalformat, GLsizei width, GLsizei height, GLint border, GLsizei imageSize, const void * data);
2001 typedef void (GLAD_API_PTR *PFNGLCOMPRESSEDTEXIMAGE3DPROC)(GLenum target, GLint level, GLenum
       internalformat, GLsizei width, GLsizei height, GLsizei depth, GLint border, GLsizei imageSize, const
       void * data);
2002 typedef void (GLAD_API_PTR *PFNGLCOMPRESSEDTEXSUBIMAGE1DPROC)(GLenum target, GLint level, GLint
       xoffset, GLsizei width, GLenum format, GLsizei imageSize, const void * data);
2003 typedef void (GLAD_API_PTR *PFNGLCOMPRESSEDTEXSUBIMAGE2DPROC)(GLenum target, GLint level, GLint
       xoffset, GLint yoffset, GLsizei width, GLsizei height, GLenum format, GLsizei imageSize, const void *
       data);
2004 typedef void (GLAD_API_PTR *PFNGLCOMPRESSEDTEXSUBIMAGE3DPROC)(GLenum target, GLint level, GLint
       xoffset, GLint yoffset, GLint zoffset, GLsizei width, GLsizei height, GLsizei depth, GLenum format,
       GLsizei imageSize, const void * data);
2005 typedef void (GLAD_API_PTR *PFNGLCOPYBUFFERSUBDATAPROC)(GLenum readTarget, GLenum writeTarget, GLintptr
       readOffset, GLintptr writeOffset, GLsizeiptr size);
2006 typedef void (GLAD_API_PTR *PFNGLCOPYPIXELSPROC)(GLint x, GLint y, GLsizei width, GLsizei height,
       GLenum type);
2007 typedef void (GLAD_API_PTR *PFNGLCOPYTEXIMAGE1DPROC)(GLenum target, GLint level, GLenum internalformat,
       GLint x, GLint y, GLsizei width, GLint border);
2008 typedef void (GLAD_API_PTR *PFNGLCOPYTEXIMAGE2DPROC)(GLenum target, GLint level, GLenum internalformat,
       GLint x, GLint y, GLsizei width, GLsizei height, GLint border);
2009 typedef void (GLAD_API_PTR *PFNGLCOPYTEXSUBIMAGE1DPROC)(GLenum target, GLint level, GLint xoffset,
       GLint x, GLint y, GLsizei width);
2010 typedef void (GLAD_API_PTR *PFNGLCOPYTEXSUBIMAGE2DPROC)(GLenum target, GLint level, GLint xoffset,
       GLint yoffset, GLint x, GLint y, GLsizei width, GLsizei height);
2011 typedef void (GLAD_API_PTR *PFNGLCOPYTEXSUBIMAGE3DPROC)(GLenum target, GLint level, GLint xoffset,
       GLint yoffset, GLint zoffset, GLint x, GLint y, GLsizei width, GLsizei height);
2012 typedef GLuint (GLAD_API_PTR *PFNGLCREATEPROGRAMPROC)(void);
2013 typedef GLuint (GLAD_API_PTR *PFNGLCREATESHADERPROC)(GLenum type);
2014 typedef void (GLAD_API_PTR *PFNGLCULLFACEPROC)(GLenum mode);
2015 typedef void (GLAD_API_PTR *PFNGLDEBUGMESSAGECALLBACKPROC)(GLDEBUGPROC callback, const void *
       userParam);
2016 typedef void (GLAD_API_PTR *PFNGLDEBUGMESSAGECONTROLPROC)(GLenum source, GLenum type, GLenum severity,
       GLsizei count, const GLuint * ids, GLboolean enabled);
2017 typedef void (GLAD_API_PTR *PFNGLDEBUGMESSAGEINSERTPROC)(GLenum source, GLenum type, GLuint id, GLenum
       severity, GLsizei length, const GLchar * buf);
2018 typedef void (GLAD_API_PTR *PFNGLDELETEBUFFERSPROC)(GLsizei n, const GLuint * buffers);
2019 typedef void (GLAD_API_PTR *PFNGLDELETEFRAMEBUFFERSPROC)(GLsizei n, const GLuint * framebuffers);
2020 typedef void (GLAD_API_PTR *PFNGLDELETELISTSPROC)(GLuint list, GLsizei range);
2021 typedef void (GLAD_API_PTR *PFNGLDELETEPROGRAMPROC)(GLuint program);
2022 typedef void (GLAD_API_PTR *PFNGLDELETEQUERIESPROC)(GLsizei n, const GLuint * ids);
2023 typedef void (GLAD_API_PTR *PFNGLDELETERENDERBUFFERSPROC)(GLsizei n, const GLuint * renderbuffers);
2024 typedef void (GLAD_API_PTR *PFNGLDELETESAMPLERSPROC)(GLsizei count, const GLuint * samplers);
2025 typedef void (GLAD_API_PTR *PFNGLDELETESHADERPROC)(GLuint shader);
2026 typedef void (GLAD_API_PTR *PFNGLDELETESYNCPROC)(GLsync sync);
2027 typedef void (GLAD_API_PTR *PFNGLDELETETEXTURESPROC)(GLsizei n, const GLuint * textures);
2028 typedef void (GLAD_API_PTR *PFNGLDELETEVERTEXARRAYSPROC)(GLsizei n, const GLuint * arrays);
2029 typedef void (GLAD_API_PTR *PFNGLDEPTHFUNCPROC)(GLenum func);
2030 typedef void (GLAD_API_PTR *PFNGLDEPTHMASKPROC)(GLboolean flag);
2031 typedef void (GLAD_API_PTR *PFNGLDEPTHRANGEPROC)(GLdouble n, GLdouble f);
2032 typedef void (GLAD_API_PTR *PFNGLDETACHSHADERPROC)(GLuint program, GLuint shader);
2033 typedef void (GLAD_API_PTR *PFNGLDISABLEPROC)(GLenum cap);
2034 typedef void (GLAD_API_PTR *PFNGLDISABLECLIENTSTATEPROC)(GLenum array);
2035 typedef void (GLAD_API_PTR *PFNGLDISABLEVERTEXATTRIBARRAYPROC)(GLuint index);
2036 typedef void (GLAD_API_PTR *PFNGLDISABLEIPROC)(GLenum target, GLuint index);
2037 typedef void (GLAD_API_PTR *PFNGLDRAWARRAYSPROC)(GLenum mode, GLint first, GLsizei count);
2038 typedef void (GLAD_API_PTR *PFNGLDRAWARRAYSINSTANCEDPROC)(GLenum mode, GLint first, GLsizei count,
       GLsizei instancecount);
2039 typedef void (GLAD_API_PTR *PFNGLDRAWBUFFERPROC)(GLenum buf);
2040 typedef void (GLAD_API_PTR *PFNGLDRAWBUFFERSPROC)(GLsizei n, const GLenum * bufs);
2041 typedef void (GLAD_API_PTR *PFNGLDRAWELEMENTSPROC)(GLenum mode, GLsizei count, GLenum type, const void
       * indices);
2042 typedef void (GLAD_API_PTR *PFNGLDRAWELEMENTSBASEVERTEXPROC)(GLenum mode, GLsizei count, GLenum type,
       const void * indices, GLint basevertex);
2043 typedef void (GLAD_API_PTR *PFNGLDRAWELEMENTSINSTANCEDPROC)(GLenum mode, GLsizei count, GLenum type,
       const void * indices, GLsizei instancecount);
2044 typedef void (GLAD_API_PTR *PFNGLDRAWELEMENTSINSTANCEDBASEVERTEXPROC)(GLenum mode, GLsizei count,
       GLenum type, const void * indices, GLsizei instancecount, GLint basevertex);
2045 typedef void (GLAD_API_PTR *PFNGLDRAWPIXELSPROC)(GLsizei width, GLsizei height, GLenum format, GLenum
       type, const void * pixels);
2046 typedef void (GLAD_API_PTR *PFNGLDRAWRANGEELEMENTSPROC)(GLenum mode, GLuint start, GLuint end, GLsizei
       count, GLenum type, const void * indices);
2047 typedef void (GLAD_API_PTR *PFNGLDRAWRANGEELEMENTSBASEVERTEXPROC)(GLenum mode, GLuint start, GLuint
       end, GLsizei count, GLenum type, const void * indices, GLint basevertex);
2048 typedef void (GLAD_API_PTR *PFNGLEDGEFLAGPROC)(GLboolean flag);
2049 typedef void (GLAD_API_PTR *PFNGLEDGEFLAGPOINTERPROC)(GLsizei stride, const void * pointer);
2050 typedef void (GLAD_API_PTR *PFNGLEDGEFLAGVPROC)(const GLboolean * flag);
2051 typedef void (GLAD_API_PTR *PFNGLENABLEPROC)(GLenum cap);
```

```
2052 typedef void (GLAD_API_PTR *PFNGLENABLECLIENTSTATEPROC)(GLenum array);
2053 typedef void (GLAD_API_PTR *PFNGLENABLEVERTEXATTRIBARRAYPROC)(GLuint index);
2054 typedef void (GLAD_API_PTR *PFNGLENABLEIPROC)(GLenum target, GLuint index);
2055 typedef void (GLAD_API_PTR *PFNGLENDPROC)(void);
2056 typedef void (GLAD_API_PTR *PFNGLENDCONDITIONALRENDERPROC)(void);
2057 typedef void (GLAD_API_PTR *PFNGLENDLISTPROC)(void);
2058 typedef void (GLAD_API_PTR *PFNGLENDQUERYPROC)(GLenum target);
2059 typedef void (GLAD_API_PTR *PFNGLENDTRANSFORMFEEDBACKPROC)(void);
2060 typedef void (GLAD_API_PTR *PFNGLEVALCOORD1DPROC)(GLdouble u);
2061 typedef void (GLAD_API_PTR *PFNGLEVALCOORD1DVPROC)(const GLdouble * u);
2062 typedef void (GLAD_API_PTR *PFNGLEVALCOORD1FPROC)(GLfloat u);
2063 typedef void (GLAD_API_PTR *PFNGLEVALCOORD1FVPROC)(const GLfloat * u);
2064 typedef void (GLAD_API_PTR *PFNGLEVALCOORD2DPROC)(GLdouble u, GLdouble v);
2065 typedef void (GLAD_API_PTR *PFNGLEVALCOORD2DVPROC)(const GLdouble * u);
2066 typedef void (GLAD_API_PTR *PFNGLEVALCOORD2FPROC)(GLfloat u, GLfloat v);
2067 typedef void (GLAD_API_PTR *PFNGLEVALCOORD2FVPROC)(const GLfloat * u);
2068 typedef void (GLAD_API_PTR *PFNGLEVALMESH1PROC)(GLenum mode, GLint i1, GLint i2);
2069 typedef void (GLAD_API_PTR *PFNGLEVALMESH2PROC)(GLenum mode, GLint i1, GLint i2, GLint j1, GLint j2);
2070 typedef void (GLAD_API_PTR *PFNGLEVALPOINT1PROC)(GLint i);
2071 typedef void (GLAD_API_PTR *PFNGLEVALPOINT2PROC)(GLint i, GLint j);
2072 typedef void (GLAD_API_PTR *PFNGLFEEDBACKBUFFERPROC)(GLsizei size, GLenum type, GLfloat * buffer);
2073 typedef GLsync (GLAD_API_PTR *PFNGLFENCESYNCPROC)(GLenum condition, GLbitfield flags);
2074 typedef void (GLAD_API_PTR *PFNGLFINISHPROC)(void);
2075 typedef void (GLAD_API_PTR *PFNGLFLUSHPROC)(void);
2076 typedef void (GLAD_API_PTR *PFNGLFLUSHMAPPEDBUFFERRANGEPROC)(GLenum target, GLintptr offset, GLsizeiptr
       length);
2077 typedef void (GLAD_API_PTR *PFNGLFOGCOORDPOINTERPROC)(GLenum type, GLsizei stride, const void *
       pointer);
2078 typedef void (GLAD_API_PTR *PFNGLFOGCOORDDPROC)(GLdouble coord);
2079 typedef void (GLAD_API_PTR *PFNGLFOGCOORDDVPROC)(const GLdouble * coord);
2080 typedef void (GLAD_API_PTR *PFNGLFOGCOORDFPROC)(GLfloat coord);
2081 typedef void (GLAD_API_PTR *PFNGLFOGCOORDFVPROC)(const GLfloat * coord);
2082 typedef void (GLAD_API_PTR *PFNGLFOGFPROC)(GLenum pname, GLfloat param);
2083 typedef void (GLAD_API_PTR *PFNGLFOGFVPROC)(GLenum pname, const GLfloat * params);
2084 typedef void (GLAD_API_PTR *PFNGLFOGIPROC)(GLenum pname, GLint param);
2085 typedef void (GLAD_API_PTR *PFNGLFOGIVPROC)(GLenum pname, const GLint * params);
2086 typedef void (GLAD_API_PTR *PFNGLFRAMEBUFFERRENDERBUFFERPROC)(GLenum target, GLenum attachment, GLenum
       renderbuffertarget, GLuint renderbuffer);
2087 typedef void (GLAD_API_PTR *PFNGLFRAMEBUFFERTEXTUREPROC)(GLenum target, GLenum attachment, GLuint
       texture, GLint level);
2088 typedef void (GLAD_API_PTR *PFNGLFRAMEBUFFERTEXTURE1DPROC)(GLenum target, GLenum attachment, GLenum
       textarget, GLuint texture, GLint level);
2089 typedef void (GLAD_API_PTR *PFNGLFRAMEBUFFERTEXTURE2DPROC)(GLenum target, GLenum attachment, GLenum
       textarget, GLuint texture, GLint level);
2090 typedef void (GLAD_API_PTR *PFNGLFRAMEBUFFERTEXTURE3DPROC)(GLenum target, GLenum attachment, GLenum
       textarget, GLuint texture, GLint level, GLint zoffset);
2091 typedef void (GLAD_API_PTR *PFNGLFRAMEBUFFERTEXTURELAYERPROC)(GLenum target, GLenum attachment, GLuint
       texture, GLint level, GLint layer);
2092 typedef void (GLAD_API_PTR *PFNGLFRONTFACEPROC)(GLenum mode);
2093 typedef void (GLAD_API_PTR *PFNGLFRUSTUMPROC)(GLdouble left, GLdouble right, GLdouble bottom, GLdouble
       top, GLdouble zNear, GLdouble zFar);
2094 typedef void (GLAD_API_PTR *PFNGLGENBUFFERSPROC)(GLsizei n, GLuint * buffers);
2095 typedef void (GLAD_API_PTR *PFNGLGENFRAMEBUFFERSPROC)(GLsizei n, GLuint * framebuffers);
2096 typedef GLuint (GLAD_API_PTR *PFNGLGENLISTSPROC)(GLsizei range);
2097 typedef void (GLAD_API_PTR *PFNGLGENQUERIESPROC)(GLsizei n, GLuint * ids);
2098 typedef void (GLAD_API_PTR *PFNGLGENRENDERBUFFERSPROC)(GLsizei n, GLuint * renderbuffers);
2099 typedef void (GLAD_API_PTR *PFNGLGENSAMPLERSPROC)(GLsizei count, GLuint * samplers);
2100 typedef void (GLAD_API_PTR *PFNGLGENTEXTURESPROC)(GLsizei n, GLuint * textures);
2101 typedef void (GLAD_API_PTR *PFNGLGENVERTEXARRAYSPROC)(GLsizei n, GLuint * arrays);
2102 typedef void (GLAD_API_PTR *PFNGLGENERATEMIPMAPPROC)(GLenum target);
2103 typedef void (GLAD_API_PTR *PFNGLGETACTIVEATTRIBPROC)(GLuint program, GLuint index, GLsizei bufSize,
       GLsizei * length, GLint * size, GLenum * type, GLchar * name);
2104 typedef void (GLAD_API_PTR *PFNGLGETACTIVEUNIFORMPROC)(GLuint program, GLuint index, GLsizei bufSize,
       GLsizei * length, GLint * size, GLenum * type, GLchar * name);
2105 typedef void (GLAD_API_PTR *PFNGLGETACTIVEUNIFORMBLOCKNAMEPROC)(GLuint program, GLuint
       uniformBlockIndex, GLsizei bufSize, GLsizei * length, GLchar * uniformBlockName);
2106 typedef void (GLAD_API_PTR *PFNGLGETACTIVEUNIFORMBLOCKIVPROC)(GLuint program, GLuint uniformBlockIndex,
       GLenum pname, GLint * params);
2107 typedef void (GLAD_API_PTR *PFNGLGETACTIVEUNIFORMNAMEPROC)(GLuint program, GLuint uniformIndex, GLsizei
       bufSize, GLsizei * length, GLchar * uniformName);
2108 typedef void (GLAD_API_PTR *PFNGLGETACTIVEUNIFORMSIVPROC)(GLuint program, GLsizei uniformCount, const
       GLuint * uniformIndices, GLenum pname, GLint * params);
2109 typedef void (GLAD_API_PTR *PFNGLGETATTACHEDSHADERSPROC)(GLuint program, GLsizei maxCount, GLsizei *
       count, GLuint * shaders);
2110 typedef GLint (GLAD_API_PTR *PFNGLGETATTRIBLOCATIONPROC)(GLuint program, const GLchar * name);
2111 typedef void (GLAD_API_PTR *PFNGLGETBOOLEANI_VPROC)(GLenum target, GLuint index, GLboolean * data);
2112 typedef void (GLAD_API_PTR *PFNGLGETBOOLEANVPROC)(GLenum pname, GLboolean * data);
2113 typedef void (GLAD_API_PTR *PFNGLGETBUFFERPARAMETERI64VPROC)(GLenum target, GLenum pname, GLint64 *
       params);
2114 typedef void (GLAD_API_PTR *PFNGLGETBUFFERPARAMETERIVPROC)(GLenum target, GLenum pname, GLint *
       params);
2115 typedef void (GLAD_API_PTR *PFNGLGETBUFFERPOINTERVPROC)(GLenum target, GLenum pname, void ** params);
2116 typedef void (GLAD_API_PTR *PFNGLGETBUFFERSUBDATAPROC)(GLenum target, GLintptr offset, GLsizeiptr size,
       void * data);
2117 typedef void (GLAD_API_PTR *PFNGLGETCLIPPLANEPROC)(GLenum plane, GLdouble * equation);
2118 typedef void (GLAD_API_PTR *PFNGLGETCOMPRESSEDTEXIMAGEPROC)(GLenum target, GLint level, void * img);
2119 typedef GLuint (GLAD_API_PTR *PFNGLGETDEBUGMESSAGELOGPROC)(GLuint count, GLsizei bufSize, GLenum *
```

```
      sources, GLenum * types, GLuint * ids, GLenum * severities, GLsizei * lengths, GLchar * messageLog);
2120 typedef void (GLAD_API_PTR *PFNGLGETDOUBLEVPROC)(GLenum pname, GLdouble * data);
2121 typedef GLenum (GLAD_API_PTR *PFNGLGETERRORPROC)(void);
2122 typedef void (GLAD_API_PTR *PFNGLGETFLOATVPROC)(GLenum pname, GLfloat * data);
2123 typedef GLint (GLAD_API_PTR *PFNGLGETFRAGDATAINDEXPROC)(GLuint program, const GLchar * name);
2124 typedef GLint (GLAD_API_PTR *PFNGLGETFRAGDATALOCATIONPROC)(GLuint program, const GLchar * name);
2125 typedef void (GLAD_API_PTR *PFNGLGETFRAMEBUFFERATTACHMENTPARAMETERIVPROC)(GLenum target, GLenum
      attachment, GLenum pname, GLint * params);
2126 typedef GLenum (GLAD_API_PTR *PFNGLGETGRAPHICSRESETSTATUSARBPROC)(void);
2127 typedef void (GLAD_API_PTR *PFNGLGETINTEGER64I_VPROC)(GLenum target, GLuint index, GLint64 * data);
2128 typedef void (GLAD_API_PTR *PFNGLGETINTEGER64VPROC)(GLenum pname, GLint64 * data);
2129 typedef void (GLAD_API_PTR *PFNGLGETINTEGERI_VPROC)(GLenum target, GLuint index, GLint * data);
2130 typedef void (GLAD_API_PTR *PFNGLGETINTEGERVPROC)(GLenum pname, GLint * data);
2131 typedef void (GLAD_API_PTR *PFNGLGETLIGHTFVPROC)(GLenum light, GLenum pname, GLfloat * params);
2132 typedef void (GLAD_API_PTR *PFNGLGETLIGHTIVPROC)(GLenum light, GLenum pname, GLint * params);
2133 typedef void (GLAD_API_PTR *PFNGLGETMAPDVPROC)(GLenum target, GLenum query, GLdouble * v);
2134 typedef void (GLAD_API_PTR *PFNGLGETMAPFVPROC)(GLenum target, GLenum query, GLfloat * v);
2135 typedef void (GLAD_API_PTR *PFNGLGETMAPIVPROC)(GLenum target, GLenum query, GLint * v);
2136 typedef void (GLAD_API_PTR *PFNGLGETMATERIALFVPROC)(GLenum face, GLenum pname, GLfloat * params);
2137 typedef void (GLAD_API_PTR *PFNGLGETMATERIALIVPROC)(GLenum face, GLenum pname, GLint * params);
2138 typedef void (GLAD_API_PTR *PFNGLGETMULTISAMPLEFVPROC)(GLenum pname, GLuint index, GLfloat * val);
2139 typedef void (GLAD_API_PTR *PFNGLGETOBJECTLABELPROC)(GLenum identifier, GLuint name, GLsizei bufSize,
      GLsizei * length, GLchar * label);
2140 typedef void (GLAD_API_PTR *PFNGLGETOBJECTPTRLABELPROC)(const void * ptr, GLsizei bufSize, GLsizei *
      length, GLchar * label);
2141 typedef void (GLAD_API_PTR *PFNGLGETPIXELMAPFVPROC)(GLenum map, GLfloat * values);
2142 typedef void (GLAD_API_PTR *PFNGLGETPIXELMAPUIVPROC)(GLenum map, GLuint * values);
2143 typedef void (GLAD_API_PTR *PFNGLGETPIXELMAPUSVPROC)(GLenum map, GLushort * values);
2144 typedef void (GLAD_API_PTR *PFNGLGETPOINTERVPROC)(GLenum pname, void ** params);
2145 typedef void (GLAD_API_PTR *PFNGLGETPOLYGONSTIPPLEPROC)(GLubyte * mask);
2146 typedef void (GLAD_API_PTR *PFNGLGETPROGRAMINFOLOGPROC)(GLuint program, GLsizei bufSize, GLsizei *
      length, GLchar * infoLog);
2147 typedef void (GLAD_API_PTR *PFNGLGETPROGRAMIVPROC)(GLuint program, GLenum pname, GLint * params);
2148 typedef void (GLAD_API_PTR *PFNGLGETQUERYOBJECTI64VPROC)(GLuint id, GLenum pname, GLint64 * params);
2149 typedef void (GLAD_API_PTR *PFNGLGETQUERYOBJECTIVPROC)(GLuint id, GLenum pname, GLint * params);
2150 typedef void (GLAD_API_PTR *PFNGLGETQUERYOBJECTUI64VPROC)(GLuint id, GLenum pname, GLuint64 * params);
2151 typedef void (GLAD_API_PTR *PFNGLGETQUERYOBJECTUIVPROC)(GLuint id, GLenum pname, GLuint * params);
2152 typedef void (GLAD_API_PTR *PFNGLGETQUERYIVPROC)(GLenum target, GLenum pname, GLint * params);
2153 typedef void (GLAD_API_PTR *PFNGLGETRENDERBUFFERPARAMETERIVPROC)(GLenum target, GLenum pname, GLint *
      params);
2154 typedef void (GLAD_API_PTR *PFNGLGETSAMPLERPARAMETERIIVPROC)(GLuint sampler, GLenum pname, GLint *
      params);
2155 typedef void (GLAD_API_PTR *PFNGLGETSAMPLERPARAMETERIUIVPROC)(GLuint sampler, GLenum pname, GLuint *
      params);
2156 typedef void (GLAD_API_PTR *PFNGLGETSAMPLERPARAMETERFVPROC)(GLuint sampler, GLenum pname, GLfloat *
      params);
2157 typedef void (GLAD_API_PTR *PFNGLGETSAMPLERPARAMETERIVPROC)(GLuint sampler, GLenum pname, GLint *
      params);
2158 typedef void (GLAD_API_PTR *PFNGLGETSHADERINFOLOGPROC)(GLuint shader, GLsizei bufSize, GLsizei *
      length, GLchar * infoLog);
2159 typedef void (GLAD_API_PTR *PFNGLGETSHADERSOURCEPROC)(GLuint shader, GLsizei bufSize, GLsizei * length,
      GLchar * source);
2160 typedef void (GLAD_API_PTR *PFNGLGETSHADERIVPROC)(GLuint shader, GLenum pname, GLint * params);
2161 typedef const GLubyte * (GLAD_API_PTR *PFNGLGETSTRINGPROC)(GLenum name);
2162 typedef const GLubyte * (GLAD_API_PTR *PFNGLGETSTRINGIPROC)(GLenum name, GLuint index);
2163 typedef void (GLAD_API_PTR *PFNGLGETSYNCIVPROC)(GLsync sync, GLenum pname, GLsizei count, GLsizei *
      length, GLint * values);
2164 typedef void (GLAD_API_PTR *PFNGLGETTEXENVFVPROC)(GLenum target, GLenum pname, GLfloat * params);
2165 typedef void (GLAD_API_PTR *PFNGLGETTEXENVIVPROC)(GLenum target, GLenum pname, GLint * params);
2166 typedef void (GLAD_API_PTR *PFNGLGETTEXGENDVPROC)(GLenum coord, GLenum pname, GLdouble * params);
2167 typedef void (GLAD_API_PTR *PFNGLGETTEXGENFVPROC)(GLenum coord, GLenum pname, GLfloat * params);
2168 typedef void (GLAD_API_PTR *PFNGLGETTEXGENIVPROC)(GLenum coord, GLenum pname, GLint * params);
2169 typedef void (GLAD_API_PTR *PFNGLGETTEXIMAGEPROC)(GLenum target, GLint level, GLenum format, GLenum
      type, void * pixels);
2170 typedef void (GLAD_API_PTR *PFNGLGETTEXLEVELPARAMETERFVPROC)(GLenum target, GLint level, GLenum pname,
      GLfloat * params);
2171 typedef void (GLAD_API_PTR *PFNGLGETTEXLEVELPARAMETERIVPROC)(GLenum target, GLint level, GLenum pname,
      GLint * params);
2172 typedef void (GLAD_API_PTR *PFNGLGETTEXPARAMETERIIVPROC)(GLenum target, GLenum pname, GLint * params);
2173 typedef void (GLAD_API_PTR *PFNGLGETTEXPARAMETERIUIVPROC)(GLenum target, GLenum pname, GLuint *
      params);
2174 typedef void (GLAD_API_PTR *PFNGLGETTEXPARAMETERFVPROC)(GLenum target, GLenum pname, GLfloat * params);
2175 typedef void (GLAD_API_PTR *PFNGLGETTEXPARAMETERIVPROC)(GLenum target, GLenum pname, GLint * params);
2176 typedef void (GLAD_API_PTR *PFNGLGETTRANSFORMFEEDBACKVARYINGPROC)(GLuint program, GLuint index, GLsizei
      bufSize, GLsizei * length, GLsizei * size, GLenum * type, GLchar * name);
2177 typedef GLuint (GLAD_API_PTR *PFNGLGETUNIFORMBLOCKINDEXPROC)(GLuint program, const GLchar *
      uniformBlockName);
2178 typedef void (GLAD_API_PTR *PFNGLGETUNIFORMINDICESPROC)(GLuint program, GLsizei uniformCount, const
      GLchar *const* uniformNames, GLuint * uniformIndices);
2179 typedef GLint (GLAD_API_PTR *PFNGLGETUNIFORMLOCATIONPROC)(GLuint program, const GLchar * name);
2180 typedef void (GLAD_API_PTR *PFNGLGETUNIFORMFVPROC)(GLuint program, GLint location, GLfloat * params);
2181 typedef void (GLAD_API_PTR *PFNGLGETUNIFORMIVPROC)(GLuint program, GLint location, GLint * params);
2182 typedef void (GLAD_API_PTR *PFNGLGETUNIFORMUIVPROC)(GLuint program, GLint location, GLuint * params);
2183 typedef void (GLAD_API_PTR *PFNGLGETVERTEXATTRIBIIVPROC)(GLuint index, GLenum pname, GLint * params);
2184 typedef void (GLAD_API_PTR *PFNGLGETVERTEXATTRIBIUIVPROC)(GLuint index, GLenum pname, GLuint * params);
2185 typedef void (GLAD_API_PTR *PFNGLGETVERTEXATTRIBPOINTERVPROC)(GLuint index, GLenum pname, void **
      pointer);
```

```
2186 typedef void (GLAD_API_PTR *PFNGLGETVERTEXATTRIBDVPROC)(GLuint index, GLenum pname, GLdouble * params);
2187 typedef void (GLAD_API_PTR *PFNGLGETVERTEXATTRIBFVPROC)(GLuint index, GLenum pname, GLfloat * params);
2188 typedef void (GLAD_API_PTR *PFNGLGETVERTEXATTRIBIVPROC)(GLuint index, GLenum pname, GLint * params);
2189 typedef void (GLAD_API_PTR *PFNGLGETNCOLORTABLEARBPROC)(GLenum target, GLenum format, GLenum type,
       GLsizei bufSize, void * table);
2190 typedef void (GLAD_API_PTR *PFNGLGETNCOMPRESSEDTEXIMAGEARBPROC)(GLenum target, GLint lod, GLsizei
       bufSize, void * img);
2191 typedef void (GLAD_API_PTR *PFNGLGETNCONVOLUTIONFILTERARBPROC)(GLenum target, GLenum format, GLenum
       type, GLsizei bufSize, void * image);
2192 typedef void (GLAD_API_PTR *PFNGLGETNHISTOGRAMARBPROC)(GLenum target, GLboolean reset, GLenum format,
       GLenum type, GLsizei bufSize, void * values);
2193 typedef void (GLAD_API_PTR *PFNGLGETNMAPDVARBPROC)(GLenum target, GLenum query, GLsizei bufSize,
       GLdouble * v);
2194 typedef void (GLAD_API_PTR *PFNGLGETNMAPFVARBPROC)(GLenum target, GLenum query, GLsizei bufSize,
       GLfloat * v);
2195 typedef void (GLAD_API_PTR *PFNGLGETNMAPIVARBPROC)(GLenum target, GLenum query, GLsizei bufSize, GLint
       * v);
2196 typedef void (GLAD_API_PTR *PFNGLGETNMINMAXARBPROC)(GLenum target, GLboolean reset, GLenum format,
       GLenum type, GLsizei bufSize, void * values);
2197 typedef void (GLAD_API_PTR *PFNGLGETNPIXELMAPFVARBPROC)(GLenum map, GLsizei bufSize, GLfloat * values);
2198 typedef void (GLAD_API_PTR *PFNGLGETNPIXELMAPUIVARBPROC)(GLenum map, GLsizei bufSize, GLuint * values);
2199 typedef void (GLAD_API_PTR *PFNGLGETNPIXELMAPUSVARBPROC)(GLenum map, GLsizei bufSize, GLushort *
       values);
2200 typedef void (GLAD_API_PTR *PFNGLGETNPOLYGONSTIPPLEARBPROC)(GLsizei bufSize, GLubyte * pattern);
2201 typedef void (GLAD_API_PTR *PFNGLGETNSEPARABLEFILTERARBPROC)(GLenum target, GLenum format, GLenum type,
       GLsizei rowBufSize, void * row, GLsizei columnBufSize, void * column, void * span);
2202 typedef void (GLAD_API_PTR *PFNGLGETNTEXIMAGEARBPROC)(GLenum target, GLint level, GLenum format, GLenum
       type, GLsizei bufSize, void * img);
2203 typedef void (GLAD_API_PTR *PFNGLGETNUNIFORMDVARBPROC)(GLuint program, GLint location, GLsizei bufSize,
       GLdouble * params);
2204 typedef void (GLAD_API_PTR *PFNGLGETNUNIFORMFVARBPROC)(GLuint program, GLint location, GLsizei bufSize,
       GLfloat * params);
2205 typedef void (GLAD_API_PTR *PFNGLGETNUNIFORMIVARBPROC)(GLuint program, GLint location, GLsizei bufSize,
       GLint * params);
2206 typedef void (GLAD_API_PTR *PFNGLGETNUNIFORMUIVARBPROC)(GLuint program, GLint location, GLsizei
       bufSize, GLuint * params);
2207 typedef void (GLAD_API_PTR *PFNGLHINTPROC)(GLenum target, GLenum mode);
2208 typedef void (GLAD_API_PTR *PFNGLINDEXMASKPROC)(GLuint mask);
2209 typedef void (GLAD_API_PTR *PFNGLINDEXPOINTERPROC)(GLenum type, GLsizei stride, const void * pointer);
2210 typedef void (GLAD_API_PTR *PFNGLINDEXDPROC)(GLdouble c);
2211 typedef void (GLAD_API_PTR *PFNGLINDEXDVPROC)(const GLdouble * c);
2212 typedef void (GLAD_API_PTR *PFNGLINDEXFPROC)(GLfloat c);
2213 typedef void (GLAD_API_PTR *PFNGLINDEXFVPROC)(const GLfloat * c);
2214 typedef void (GLAD_API_PTR *PFNGLINDEXIPROC)(GLint c);
2215 typedef void (GLAD_API_PTR *PFNGLINDEXIVPROC)(const GLint * c);
2216 typedef void (GLAD_API_PTR *PFNGLINDEXSPROC)(GLshort c);
2217 typedef void (GLAD_API_PTR *PFNGLINDEXSVPROC)(const GLshort * c);
2218 typedef void (GLAD_API_PTR *PFNGLINDEXUBPROC)(GLubyte c);
2219 typedef void (GLAD_API_PTR *PFNGLINDEXUBVPROC)(const GLubyte * c);
2220 typedef void (GLAD_API_PTR *PFNGLINITNAMESPROC)(void);
2221 typedef void (GLAD_API_PTR *PFNGLINTERLEAVEDARRAYSPROC)(GLenum format, GLsizei stride, const void *
       pointer);
2222 typedef GLboolean (GLAD_API_PTR *PFNGLISBUFFERPROC)(GLuint buffer);
2223 typedef GLboolean (GLAD_API_PTR *PFNGLISENABLEDPROC)(GLenum cap);
2224 typedef GLboolean (GLAD_API_PTR *PFNGLISENABLEDIPROC)(GLenum target, GLuint index);
2225 typedef GLboolean (GLAD_API_PTR *PFNGLISFRAMEBUFFERPROC)(GLuint framebuffer);
2226 typedef GLboolean (GLAD_API_PTR *PFNGLISLISTPROC)(GLuint list);
2227 typedef GLboolean (GLAD_API_PTR *PFNGLISPROGRAMPROC)(GLuint program);
2228 typedef GLboolean (GLAD_API_PTR *PFNGLISQUERYPROC)(GLuint id);
2229 typedef GLboolean (GLAD_API_PTR *PFNGLISRENDERBUFFERPROC)(GLuint renderbuffer);
2230 typedef GLboolean (GLAD_API_PTR *PFNGLISSAMPLERPROC)(GLuint sampler);
2231 typedef GLboolean (GLAD_API_PTR *PFNGLISSHADERPROC)(GLuint shader);
2232 typedef GLboolean (GLAD_API_PTR *PFNGLISSYNCPROC)(GLsync sync);
2233 typedef GLboolean (GLAD_API_PTR *PFNGLISTEXTUREPROC)(GLuint texture);
2234 typedef GLboolean (GLAD_API_PTR *PFNGLISVERTEXARRAYPROC)(GLuint array);
2235 typedef void (GLAD_API_PTR *PFNGLLIGHTMODELFPROC)(GLenum pname, GLfloat param);
2236 typedef void (GLAD_API_PTR *PFNGLLIGHTMODELFVPROC)(GLenum pname, const GLfloat * params);
2237 typedef void (GLAD_API_PTR *PFNGLLIGHTMODELIPROC)(GLenum pname, GLint param);
2238 typedef void (GLAD_API_PTR *PFNGLLIGHTMODELIVPROC)(GLenum pname, const GLint * params);
2239 typedef void (GLAD_API_PTR *PFNGLLIGHTFPROC)(GLenum light, GLenum pname, GLfloat param);
2240 typedef void (GLAD_API_PTR *PFNGLLIGHTFVPROC)(GLenum light, GLenum pname, const GLfloat * params);
2241 typedef void (GLAD_API_PTR *PFNGLLIGHTIPROC)(GLenum light, GLenum pname, GLint param);
2242 typedef void (GLAD_API_PTR *PFNGLLIGHTIVPROC)(GLenum light, GLenum pname, const GLint * params);
2243 typedef void (GLAD_API_PTR *PFNGLLINESTIPPLEPROC)(GLint factor, GLushort pattern);
2244 typedef void (GLAD_API_PTR *PFNGLLINEWIDTHPROC)(GLfloat width);
2245 typedef void (GLAD_API_PTR *PFNGLLINKPROGRAMPROC)(GLuint program);
2246 typedef void (GLAD_API_PTR *PFNGLLISTBASEPROC)(GLuint base);
2247 typedef void (GLAD_API_PTR *PFNGLLOADIDENTITYPROC)(void);
2248 typedef void (GLAD_API_PTR *PFNGLLOADMATRIXDPROC)(const GLdouble * m);
2249 typedef void (GLAD_API_PTR *PFNGLLOADMATRIXFPROC)(const GLfloat * m);
2250 typedef void (GLAD_API_PTR *PFNGLLOADNAMEPROC)(GLuint name);
2251 typedef void (GLAD_API_PTR *PFNGLLOADTRANSPOSEMATRIXDPROC)(const GLdouble * m);
2252 typedef void (GLAD_API_PTR *PFNGLLOADTRANSPOSEMATRIXFPROC)(const GLfloat * m);
2253 typedef void (GLAD_API_PTR *PFNGLLOGICOPPROC)(GLenum opcode);
2254 typedef void (GLAD_API_PTR *PFNGLMAP1DPROC)(GLenum target, GLdouble u1, GLdouble u2, GLint stride,
       GLint order, const GLdouble * points);
2255 typedef void (GLAD_API_PTR *PFNGLMAP1FPROC)(GLenum target, GLfloat u1, GLfloat u2, GLint stride, GLint
```

```
      order, const GLfloat * points);
2256 typedef void (GLAD_API_PTR *PFNGLMAP2DPROC)(GLenum target, GLdouble u1, GLdouble u2, GLint ustride,
      GLint uorder, GLdouble v1, GLdouble v2, GLint vstride, GLint vorder, const GLdouble * points);
2257 typedef void (GLAD_API_PTR *PFNGLMAP2FPROC)(GLenum target, GLfloat u1, GLfloat u2, GLint ustride, GLint
      uorder, GLfloat v1, GLfloat v2, GLint vstride, GLint vorder, const GLfloat * points);
2258 typedef void * (GLAD_API_PTR *PFNGLMAPBUFFERPROC)(GLenum target, GLenum access);
2259 typedef void * (GLAD_API_PTR *PFNGLMAPBUFFERRANGEPROC)(GLenum target, GLintptr offset, GLsizeiptr
      length, GLbitfield access);
2260 typedef void (GLAD_API_PTR *PFNGLMAPGRID1DPROC)(GLint un, GLdouble u1, GLdouble u2);
2261 typedef void (GLAD_API_PTR *PFNGLMAPGRID1FPROC)(GLint un, GLfloat u1, GLfloat u2);
2262 typedef void (GLAD_API_PTR *PFNGLMAPGRID2DPROC)(GLint un, GLdouble u1, GLdouble u2, GLint vn, GLdouble
      v1, GLdouble v2);
2263 typedef void (GLAD_API_PTR *PFNGLMAPGRID2FPROC)(GLint un, GLfloat u1, GLfloat u2, GLint vn, GLfloat v1,
      GLfloat v2);
2264 typedef void (GLAD_API_PTR *PFNGLMATERIALFPROC)(GLenum face, GLenum pname, GLfloat param);
2265 typedef void (GLAD_API_PTR *PFNGLMATERIALFVPROC)(GLenum face, GLenum pname, const GLfloat * params);
2266 typedef void (GLAD_API_PTR *PFNGLMATERIALIPROC)(GLenum face, GLenum pname, GLint param);
2267 typedef void (GLAD_API_PTR *PFNGLMATERIALIVPROC)(GLenum face, GLenum pname, const GLint * params);
2268 typedef void (GLAD_API_PTR *PFNGLMATRIXMODEPROC)(GLenum mode);
2269 typedef void (GLAD_API_PTR *PFNGLMULTMATRIXDPROC)(const GLdouble * m);
2270 typedef void (GLAD_API_PTR *PFNGLMULTMATRIXFPROC)(const GLfloat * m);
2271 typedef void (GLAD_API_PTR *PFNGLMULTTRANSPOSEMATRIXDPROC)(const GLdouble * m);
2272 typedef void (GLAD_API_PTR *PFNGLMULTTRANSPOSEMATRIXFPROC)(const GLfloat * m);
2273 typedef void (GLAD_API_PTR *PFNGLMULTIDRAWARRAYSPROC)(GLenum mode, const GLint * first, const GLsizei *
      count, GLsizei drawcount);
2274 typedef void (GLAD_API_PTR *PFNGLMULTIDRAWELEMENTSPROC)(GLenum mode, const GLsizei * count, GLenum
      type, const void *const* indices, GLsizei drawcount);
2275 typedef void (GLAD_API_PTR *PFNGLMULTIDRAWELEMENTSBASEVERTEXPROC)(GLenum mode, const GLsizei * count,
      GLenum type, const void *const* indices, GLsizei drawcount, const GLint * basevertex);
2276 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD1DPROC)(GLenum target, GLdouble s);
2277 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD1DVPROC)(GLenum target, const GLdouble * v);
2278 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD1FPROC)(GLenum target, GLfloat s);
2279 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD1FVPROC)(GLenum target, const GLfloat * v);
2280 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD1IPROC)(GLenum target, GLint s);
2281 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD1IVPROC)(GLenum target, const GLint * v);
2282 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD1SPROC)(GLenum target, GLshort s);
2283 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD1SVPROC)(GLenum target, const GLshort * v);
2284 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD2DPROC)(GLenum target, GLdouble s, GLdouble t);
2285 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD2DVPROC)(GLenum target, const GLdouble * v);
2286 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD2FPROC)(GLenum target, GLfloat s, GLfloat t);
2287 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD2FVPROC)(GLenum target, const GLfloat * v);
2288 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD2IPROC)(GLenum target, GLint s, GLint t);
2289 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD2IVPROC)(GLenum target, const GLint * v);
2290 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD2SPROC)(GLenum target, GLshort s, GLshort t);
2291 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD2SVPROC)(GLenum target, const GLshort * v);
2292 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD3DPROC)(GLenum target, GLdouble s, GLdouble t, GLdouble
      r);
2293 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD3DVPROC)(GLenum target, const GLdouble * v);
2294 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD3FPROC)(GLenum target, GLfloat s, GLfloat t, GLfloat r);
2295 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD3FVPROC)(GLenum target, const GLfloat * v);
2296 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD3IPROC)(GLenum target, GLint s, GLint t, GLint r);
2297 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD3IVPROC)(GLenum target, const GLint * v);
2298 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD3SPROC)(GLenum target, GLshort s, GLshort t, GLshort r);
2299 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD3SVPROC)(GLenum target, const GLshort * v);
2300 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD4DPROC)(GLenum target, GLdouble s, GLdouble t, GLdouble
      r, GLdouble q);
2301 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD4DVPROC)(GLenum target, const GLdouble * v);
2302 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD4FPROC)(GLenum target, GLfloat s, GLfloat t, GLfloat r,
      GLfloat q);
2303 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD4FVPROC)(GLenum target, const GLfloat * v);
2304 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD4IPROC)(GLenum target, GLint s, GLint t, GLint r, GLint
      q);
2305 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD4IVPROC)(GLenum target, const GLint * v);
2306 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD4SPROC)(GLenum target, GLshort s, GLshort t, GLshort r,
      GLshort q);
2307 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORD4SVPROC)(GLenum target, const GLshort * v);
2308 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORDP1UIPROC)(GLenum texture, GLenum type, GLuint coords);
2309 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORDP1UIVPROC)(GLenum texture, GLenum type, const GLuint *
      coords);
2310 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORDP2UIPROC)(GLenum texture, GLenum type, GLuint coords);
2311 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORDP2UIVPROC)(GLenum texture, GLenum type, const GLuint *
      coords);
2312 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORDP3UIPROC)(GLenum texture, GLenum type, GLuint coords);
2313 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORDP3UIVPROC)(GLenum texture, GLenum type, const GLuint *
      coords);
2314 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORDP4UIPROC)(GLenum texture, GLenum type, GLuint coords);
2315 typedef void (GLAD_API_PTR *PFNGLMULTITEXCOORDP4UIVPROC)(GLenum texture, GLenum type, const GLuint *
      coords);
2316 typedef void (GLAD_API_PTR *PFNGLNEWLISTPROC)(GLuint list, GLenum mode);
2317 typedef void (GLAD_API_PTR *PFNGLNORMAL3BPROC)(GLbyte nx, GLbyte ny, GLbyte nz);
2318 typedef void (GLAD_API_PTR *PFNGLNORMAL3BVPROC)(const GLbyte * v);
2319 typedef void (GLAD_API_PTR *PFNGLNORMAL3DPROC)(GLdouble nx, GLdouble ny, GLdouble nz);
2320 typedef void (GLAD_API_PTR *PFNGLNORMAL3DVPROC)(const GLdouble * v);
2321 typedef void (GLAD_API_PTR *PFNGLNORMAL3FPROC)(GLfloat nx, GLfloat ny, GLfloat nz);
2322 typedef void (GLAD_API_PTR *PFNGLNORMAL3FVPROC)(const GLfloat * v);
2323 typedef void (GLAD_API_PTR *PFNGLNORMAL3IPROC)(GLint nx, GLint ny, GLint nz);
2324 typedef void (GLAD_API_PTR *PFNGLNORMAL3IVPROC)(const GLint * v);
```

```
2325 typedef void (GLAD_API_PTR *PFNGLNORMAL3SPROC)(GLshort nx, GLshort ny, GLshort nz);
2326 typedef void (GLAD_API_PTR *PFNGLNORMAL3SVPROC)(const GLshort * v);
2327 typedef void (GLAD_API_PTR *PFNGLNORMALP3UIPROC)(GLenum type, GLuint coords);
2328 typedef void (GLAD_API_PTR *PFNGLNORMALP3UIVPROC)(GLenum type, const GLuint * coords);
2329 typedef void (GLAD_API_PTR *PFNGLNORMALPOINTERPROC)(GLenum type, GLsizei stride, const void * pointer);
2330 typedef void (GLAD_API_PTR *PFNGLOBJECTLABELPROC)(GLenum identifier, GLuint name, GLsizei length, const
       GLchar * label);
2331 typedef void (GLAD_API_PTR *PFNGLOBJECTPTRLABELPROC)(const void * ptr, GLsizei length, const GLchar *
       label);
2332 typedef void (GLAD_API_PTR *PFNGLORTHOPROC)(GLdouble left, GLdouble right, GLdouble bottom, GLdouble
       top, GLdouble zNear, GLdouble zFar);
2333 typedef void (GLAD_API_PTR *PFNGLPASSTHROUGHPROC)(GLfloat token);
2334 typedef void (GLAD_API_PTR *PFNGLPIXELMAPFVPROC)(GLenum map, GLsizei mapsize, const GLfloat * values);
2335 typedef void (GLAD_API_PTR *PFNGLPIXELMAPUIVPROC)(GLenum map, GLsizei mapsize, const GLuint * values);
2336 typedef void (GLAD_API_PTR *PFNGLPIXELMAPUSVPROC)(GLenum map, GLsizei mapsize, const GLushort *
       values);
2337 typedef void (GLAD_API_PTR *PFNGLPIXELSTOREFPROC)(GLenum pname, GLfloat param);
2338 typedef void (GLAD_API_PTR *PFNGLPIXELSTOREIPROC)(GLenum pname, GLint param);
2339 typedef void (GLAD_API_PTR *PFNGLPIXELTRANSFERFPROC)(GLenum pname, GLfloat param);
2340 typedef void (GLAD_API_PTR *PFNGLPIXELTRANSFERIPROC)(GLenum pname, GLint param);
2341 typedef void (GLAD_API_PTR *PFNGLPIXELZOOMPROC)(GLfloat xfactor, GLfloat yfactor);
2342 typedef void (GLAD_API_PTR *PFNGLPOINTPARAMETERFPROC)(GLenum pname, GLfloat param);
2343 typedef void (GLAD_API_PTR *PFNGLPOINTPARAMETERFVPROC)(GLenum pname, const GLfloat * params);
2344 typedef void (GLAD_API_PTR *PFNGLPOINTPARAMETERIPROC)(GLenum pname, GLint param);
2345 typedef void (GLAD_API_PTR *PFNGLPOINTPARAMETERIVPROC)(GLenum pname, const GLint * params);
2346 typedef void (GLAD_API_PTR *PFNGLPOINTSIZEPROC)(GLfloat size);
2347 typedef void (GLAD_API_PTR *PFNGLPOLYGONMODEPROC)(GLenum face, GLenum mode);
2348 typedef void (GLAD_API_PTR *PFNGLPOLYGONOFFSETPROC)(GLfloat factor, GLfloat units);
2349 typedef void (GLAD_API_PTR *PFNGLPOLYGONSTIPPLEPROC)(const GLubyte * mask);
2350 typedef void (GLAD_API_PTR *PFNGLPOPATTRIBPROC)(void);
2351 typedef void (GLAD_API_PTR *PFNGLPOPCLIENTATTRIBPROC)(void);
2352 typedef void (GLAD_API_PTR *PFNGLPOPDEBUGGROUPPROC)(void);
2353 typedef void (GLAD_API_PTR *PFNGLPOPMATRIXPROC)(void);
2354 typedef void (GLAD_API_PTR *PFNGLPOPNAMEPROC)(void);
2355 typedef void (GLAD_API_PTR *PFNGLPRIMITIVERESTARTINDEXPROC)(GLuint index);
2356 typedef void (GLAD_API_PTR *PFNGLPRIORITIZETEXTURESPROC)(GLsizei n, const GLuint * textures, const
       GLfloat * priorities);
2357 typedef void (GLAD_API_PTR *PFNGLPROVOKINGVERTEXPROC)(GLenum mode);
2358 typedef void (GLAD_API_PTR *PFNGLPUSHATTRIBPROC)(GLbitfield mask);
2359 typedef void (GLAD_API_PTR *PFNGLPUSHCLIENTATTRIBPROC)(GLbitfield mask);
2360 typedef void (GLAD_API_PTR *PFNGLPUSHDEBUGGROUPPROC)(GLenum source, GLuint id, GLsizei length, const
       GLchar * message);
2361 typedef void (GLAD_API_PTR *PFNGLPUSHMATRIXPROC)(void);
2362 typedef void (GLAD_API_PTR *PFNGLPUSHNAMEPROC)(GLuint name);
2363 typedef void (GLAD_API_PTR *PFNGLQUERYCOUNTERPROC)(GLuint id, GLenum target);
2364 typedef void (GLAD_API_PTR *PFNGLRASTERPOS2DPROC)(GLdouble x, GLdouble y);
2365 typedef void (GLAD_API_PTR *PFNGLRASTERPOS2DVPROC)(const GLdouble * v);
2366 typedef void (GLAD_API_PTR *PFNGLRASTERPOS2FPROC)(GLfloat x, GLfloat y);
2367 typedef void (GLAD_API_PTR *PFNGLRASTERPOS2FVPROC)(const GLfloat * v);
2368 typedef void (GLAD_API_PTR *PFNGLRASTERPOS2IPROC)(GLint x, GLint y);
2369 typedef void (GLAD_API_PTR *PFNGLRASTERPOS2IVPROC)(const GLint * v);
2370 typedef void (GLAD_API_PTR *PFNGLRASTERPOS2SPROC)(GLshort x, GLshort y);
2371 typedef void (GLAD_API_PTR *PFNGLRASTERPOS2SVPROC)(const GLshort * v);
2372 typedef void (GLAD_API_PTR *PFNGLRASTERPOS3DPROC)(GLdouble x, GLdouble y, GLdouble z);
2373 typedef void (GLAD_API_PTR *PFNGLRASTERPOS3DVPROC)(const GLdouble * v);
2374 typedef void (GLAD_API_PTR *PFNGLRASTERPOS3FPROC)(GLfloat x, GLfloat y, GLfloat z);
2375 typedef void (GLAD_API_PTR *PFNGLRASTERPOS3FVPROC)(const GLfloat * v);
2376 typedef void (GLAD_API_PTR *PFNGLRASTERPOS3IPROC)(GLint x, GLint y, GLint z);
2377 typedef void (GLAD_API_PTR *PFNGLRASTERPOS3IVPROC)(const GLint * v);
2378 typedef void (GLAD_API_PTR *PFNGLRASTERPOS3SPROC)(GLshort x, GLshort y, GLshort z);
2379 typedef void (GLAD_API_PTR *PFNGLRASTERPOS3SVPROC)(const GLshort * v);
2380 typedef void (GLAD_API_PTR *PFNGLRASTERPOS4DPROC)(GLdouble x, GLdouble y, GLdouble z, GLdouble w);
2381 typedef void (GLAD_API_PTR *PFNGLRASTERPOS4DVPROC)(const GLdouble * v);
2382 typedef void (GLAD_API_PTR *PFNGLRASTERPOS4FPROC)(GLfloat x, GLfloat y, GLfloat z, GLfloat w);
2383 typedef void (GLAD_API_PTR *PFNGLRASTERPOS4FVPROC)(const GLfloat * v);
2384 typedef void (GLAD_API_PTR *PFNGLRASTERPOS4IPROC)(GLint x, GLint y, GLint z, GLint w);
2385 typedef void (GLAD_API_PTR *PFNGLRASTERPOS4IVPROC)(const GLint * v);
2386 typedef void (GLAD_API_PTR *PFNGLRASTERPOS4SPROC)(GLshort x, GLshort y, GLshort z, GLshort w);
2387 typedef void (GLAD_API_PTR *PFNGLRASTERPOS4SVPROC)(const GLshort * v);
2388 typedef void (GLAD_API_PTR *PFNGLREADBUFFERPROC)(GLenum src);
2389 typedef void (GLAD_API_PTR *PFNGLREADPIXELSPROC)(GLint x, GLint y, GLsizei width, GLsizei height,
       GLenum format, GLenum type, void * pixels);
2390 typedef void (GLAD_API_PTR *PFNGLREADNPIXELSARBPROC)(GLint x, GLint y, GLsizei width, GLsizei height,
       GLenum format, GLenum type, GLsizei bufSize, void * data);
2391 typedef void (GLAD_API_PTR *PFNGLRECTDPROC)(GLdouble x1, GLdouble y1, GLdouble x2, GLdouble y2);
2392 typedef void (GLAD_API_PTR *PFNGLRECTDVPROC)(const GLdouble * v1, const GLdouble * v2);
2393 typedef void (GLAD_API_PTR *PFNGLRECTFPROC)(GLfloat x1, GLfloat y1, GLfloat x2, GLfloat y2);
2394 typedef void (GLAD_API_PTR *PFNGLRECTFVPROC)(const GLfloat * v1, const GLfloat * v2);
2395 typedef void (GLAD_API_PTR *PFNGLRECTIPROC)(GLint x1, GLint y1, GLint x2, GLint y2);
2396 typedef void (GLAD_API_PTR *PFNGLRECTIVPROC)(const GLint * v1, const GLint * v2);
2397 typedef void (GLAD_API_PTR *PFNGLRECTSPROC)(GLshort x1, GLshort y1, GLshort x2, GLshort y2);
2398 typedef void (GLAD_API_PTR *PFNGLRECTSVPROC)(const GLshort * v1, const GLshort * v2);
2399 typedef GLint (GLAD_API_PTR *PFNGLRENDERMODEPROC)(GLenum mode);
2400 typedef void (GLAD_API_PTR *PFNGLRENDERBUFFERSTORAGEPROC)(GLenum target, GLenum internalformat, GLsizei
       width, GLsizei height);
2401 typedef void (GLAD_API_PTR *PFNGLRENDERBUFFERSTORAGEMULTISAMPLEPROC)(GLenum target, GLsizei samples,
       GLenum internalformat, GLsizei width, GLsizei height);
```

```
2402 typedef void (GLAD_API_PTR *PFNGLROTATEDPROC)(GLdouble angle, GLdouble x, GLdouble y, GLdouble z);
2403 typedef void (GLAD_API_PTR *PFNGLROTATEFPROC)(GLfloat angle, GLfloat x, GLfloat y, GLfloat z);
2404 typedef void (GLAD_API_PTR *PFNGLSAMPLECOVERAGEPROC)(GLfloat value, GLboolean invert);
2405 typedef void (GLAD_API_PTR *PFNGLSAMPLECOVERAGEARBPROC)(GLfloat value, GLboolean invert);
2406 typedef void (GLAD_API_PTR *PFNGLSAMPLEMASKIPROC)(GLuint maskNumber, GLbitfield mask);
2407 typedef void (GLAD_API_PTR *PFNGLSAMPLERPARAMETERIIVPROC)(GLuint sampler, GLenum pname, const GLint *
       param);
2408 typedef void (GLAD_API_PTR *PFNGLSAMPLERPARAMETERIUIVPROC)(GLuint sampler, GLenum pname, const GLuint *
       param);
2409 typedef void (GLAD_API_PTR *PFNGLSAMPLERPARAMETERFPROC)(GLuint sampler, GLenum pname, GLfloat param);
2410 typedef void (GLAD_API_PTR *PFNGLSAMPLERPARAMETERFVPROC)(GLuint sampler, GLenum pname, const GLfloat *
       param);
2411 typedef void (GLAD_API_PTR *PFNGLSAMPLERPARAMETERIPROC)(GLuint sampler, GLenum pname, GLint param);
2412 typedef void (GLAD_API_PTR *PFNGLSAMPLERPARAMETERIVPROC)(GLuint sampler, GLenum pname, const GLint *
       param);
2413 typedef void (GLAD_API_PTR *PFNGLSCALEDPROC)(GLdouble x, GLdouble y, GLdouble z);
2414 typedef void (GLAD_API_PTR *PFNGLSCALEFPROC)(GLfloat x, GLfloat y, GLfloat z);
2415 typedef void (GLAD_API_PTR *PFNGLSCISSORPROC)(GLint x, GLint y, GLsizei width, GLsizei height);
2416 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLOR3BPROC)(GLbyte red, GLbyte green, GLbyte blue);
2417 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLOR3BVPROC)(const GLbyte * v);
2418 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLOR3DPROC)(GLdouble red, GLdouble green, GLdouble blue);
2419 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLOR3DVPROC)(const GLdouble * v);
2420 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLOR3FPROC)(GLfloat red, GLfloat green, GLfloat blue);
2421 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLOR3FVPROC)(const GLfloat * v);
2422 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLOR3IPROC)(GLint red, GLint green, GLint blue);
2423 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLOR3IVPROC)(const GLint * v);
2424 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLOR3SPROC)(GLshort red, GLshort green, GLshort blue);
2425 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLOR3SVPROC)(const GLshort * v);
2426 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLOR3UBPROC)(GLubyte red, GLubyte green, GLubyte blue);
2427 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLOR3UBVPROC)(const GLubyte * v);
2428 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLOR3UIPROC)(GLuint red, GLuint green, GLuint blue);
2429 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLOR3UIVPROC)(const GLuint * v);
2430 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLOR3USPROC)(GLushort red, GLushort green, GLushort blue);
2431 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLOR3USVPROC)(const GLushort * v);
2432 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLORP3UIPROC)(GLenum type, GLuint color);
2433 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLORP3UIVPROC)(GLenum type, const GLuint * color);
2434 typedef void (GLAD_API_PTR *PFNGLSECONDARYCOLORPOINTERPROC)(GLint size, GLenum type, GLsizei stride,
       const void * pointer);
2435 typedef void (GLAD_API_PTR *PFNGLSELECTBUFFERPROC)(GLsizei size, GLuint * buffer);
2436 typedef void (GLAD_API_PTR *PFNGLSHADEMODELPROC)(GLenum mode);
2437 typedef void (GLAD_API_PTR *PFNGLSHADERSOURCEPROC)(GLuint shader, GLsizei count, const GLchar *const*
       string, const GLint * length);
2438 typedef void (GLAD_API_PTR *PFNGLSTENCILFUNCPROC)(GLenum func, GLint ref, GLuint mask);
2439 typedef void (GLAD_API_PTR *PFNGLSTENCILFUNCSEPARATEPROC)(GLenum face, GLenum func, GLint ref, GLuint
       mask);
2440 typedef void (GLAD_API_PTR *PFNGLSTENCILMASKPROC)(GLuint mask);
2441 typedef void (GLAD_API_PTR *PFNGLSTENCILMASKSEPARATEPROC)(GLenum face, GLuint mask);
2442 typedef void (GLAD_API_PTR *PFNGLSTENCILOPPROC)(GLenum fail, GLenum zfail, GLenum zpass);
2443 typedef void (GLAD_API_PTR *PFNGLSTENCILOPSEPARATEPROC)(GLenum face, GLenum sfail, GLenum dpfail,
       GLenum dppass);
2444 typedef void (GLAD_API_PTR *PFNGLTEXBUFFERPROC)(GLenum target, GLenum internalformat, GLuint buffer);
2445 typedef void (GLAD_API_PTR *PFNGLTEXCOORD1DPROC)(GLdouble s);
2446 typedef void (GLAD_API_PTR *PFNGLTEXCOORD1DVPROC)(const GLdouble * v);
2447 typedef void (GLAD_API_PTR *PFNGLTEXCOORD1FPROC)(GLfloat s);
2448 typedef void (GLAD_API_PTR *PFNGLTEXCOORD1FVPROC)(const GLfloat * v);
2449 typedef void (GLAD_API_PTR *PFNGLTEXCOORD1IPROC)(GLint s);
2450 typedef void (GLAD_API_PTR *PFNGLTEXCOORD1IVPROC)(const GLint * v);
2451 typedef void (GLAD_API_PTR *PFNGLTEXCOORD1SPROC)(GLshort s);
2452 typedef void (GLAD_API_PTR *PFNGLTEXCOORD1SVPROC)(const GLshort * v);
2453 typedef void (GLAD_API_PTR *PFNGLTEXCOORD2DPROC)(GLdouble s, GLdouble t);
2454 typedef void (GLAD_API_PTR *PFNGLTEXCOORD2DVPROC)(const GLdouble * v);
2455 typedef void (GLAD_API_PTR *PFNGLTEXCOORD2FPROC)(GLfloat s, GLfloat t);
2456 typedef void (GLAD_API_PTR *PFNGLTEXCOORD2FVPROC)(const GLfloat * v);
2457 typedef void (GLAD_API_PTR *PFNGLTEXCOORD2IPROC)(GLint s, GLint t);
2458 typedef void (GLAD_API_PTR *PFNGLTEXCOORD2IVPROC)(const GLint * v);
2459 typedef void (GLAD_API_PTR *PFNGLTEXCOORD2SPROC)(GLshort s, GLshort t);
2460 typedef void (GLAD_API_PTR *PFNGLTEXCOORD2SVPROC)(const GLshort * v);
2461 typedef void (GLAD_API_PTR *PFNGLTEXCOORD3DPROC)(GLdouble s, GLdouble t, GLdouble r);
2462 typedef void (GLAD_API_PTR *PFNGLTEXCOORD3DVPROC)(const GLdouble * v);
2463 typedef void (GLAD_API_PTR *PFNGLTEXCOORD3FPROC)(GLfloat s, GLfloat t, GLfloat r);
2464 typedef void (GLAD_API_PTR *PFNGLTEXCOORD3FVPROC)(const GLfloat * v);
2465 typedef void (GLAD_API_PTR *PFNGLTEXCOORD3IPROC)(GLint s, GLint t, GLint r);
2466 typedef void (GLAD_API_PTR *PFNGLTEXCOORD3IVPROC)(const GLint * v);
2467 typedef void (GLAD_API_PTR *PFNGLTEXCOORD3SPROC)(GLshort s, GLshort t, GLshort r);
2468 typedef void (GLAD_API_PTR *PFNGLTEXCOORD3SVPROC)(const GLshort * v);
2469 typedef void (GLAD_API_PTR *PFNGLTEXCOORD4DPROC)(GLdouble s, GLdouble t, GLdouble r, GLdouble q);
2470 typedef void (GLAD_API_PTR *PFNGLTEXCOORD4DVPROC)(const GLdouble * v);
2471 typedef void (GLAD_API_PTR *PFNGLTEXCOORD4FPROC)(GLfloat s, GLfloat t, GLfloat r, GLfloat q);
2472 typedef void (GLAD_API_PTR *PFNGLTEXCOORD4FVPROC)(const GLfloat * v);
2473 typedef void (GLAD_API_PTR *PFNGLTEXCOORD4IPROC)(GLint s, GLint t, GLint r, GLint q);
2474 typedef void (GLAD_API_PTR *PFNGLTEXCOORD4IVPROC)(const GLint * v);
2475 typedef void (GLAD_API_PTR *PFNGLTEXCOORD4SPROC)(GLshort s, GLshort t, GLshort r, GLshort q);
2476 typedef void (GLAD_API_PTR *PFNGLTEXCOORD4SVPROC)(const GLshort * v);
2477 typedef void (GLAD_API_PTR *PFNGLTEXCOORDP1UIPROC)(GLenum type, GLuint coords);
2478 typedef void (GLAD_API_PTR *PFNGLTEXCOORDP1UIVPROC)(GLenum type, const GLuint * coords);
2479 typedef void (GLAD_API_PTR *PFNGLTEXCOORDP2UIPROC)(GLenum type, GLuint coords);
2480 typedef void (GLAD_API_PTR *PFNGLTEXCOORDP2UIVPROC)(GLenum type, const GLuint * coords);
```

```
2481 typedef void (GLAD_API_PTR *PFNGLTEXCOORDP3UIPROC)(GLenum type, GLuint coords);
2482 typedef void (GLAD_API_PTR *PFNGLTEXCOORDP3UIVPROC)(GLenum type, const GLuint * coords);
2483 typedef void (GLAD_API_PTR *PFNGLTEXCOORDP4UIPROC)(GLenum type, GLuint coords);
2484 typedef void (GLAD_API_PTR *PFNGLTEXCOORDP4UIVPROC)(GLenum type, const GLuint * coords);
2485 typedef void (GLAD_API_PTR *PFNGLTEXCOORDPOINTERPROC)(GLint size, GLenum type, GLsizei stride, const
       void * pointer);
2486 typedef void (GLAD_API_PTR *PFNGLTEXENVFPROC)(GLenum target, GLenum pname, GLfloat param);
2487 typedef void (GLAD_API_PTR *PFNGLTEXENVFVPROC)(GLenum target, GLenum pname, const GLfloat * params);
2488 typedef void (GLAD_API_PTR *PFNGLTEXENVIPROC)(GLenum target, GLenum pname, GLint param);
2489 typedef void (GLAD_API_PTR *PFNGLTEXENVIVPROC)(GLenum target, GLenum pname, const GLint * params);
2490 typedef void (GLAD_API_PTR *PFNGLTEXGENDPROC)(GLenum coord, GLenum pname, GLdouble param);
2491 typedef void (GLAD_API_PTR *PFNGLTEXGENDVPROC)(GLenum coord, GLenum pname, const GLdouble * params);
2492 typedef void (GLAD_API_PTR *PFNGLTEXGENFPROC)(GLenum coord, GLenum pname, GLfloat param);
2493 typedef void (GLAD_API_PTR *PFNGLTEXGENFVPROC)(GLenum coord, GLenum pname, const GLfloat * params);
2494 typedef void (GLAD_API_PTR *PFNGLTEXGENIPROC)(GLenum coord, GLenum pname, GLint param);
2495 typedef void (GLAD_API_PTR *PFNGLTEXGENIVPROC)(GLenum coord, GLenum pname, const GLint * params);
2496 typedef void (GLAD_API_PTR *PFNGLTEXIMAGE1DPROC)(GLenum target, GLint level, GLint internalformat,
       GLsizei width, GLint border, GLenum format, GLenum type, const void * pixels);
2497 typedef void (GLAD_API_PTR *PFNGLTEXIMAGE2DPROC)(GLenum target, GLint level, GLint internalformat,
       GLsizei width, GLsizei height, GLint border, GLenum format, GLenum type, const void * pixels);
2498 typedef void (GLAD_API_PTR *PFNGLTEXIMAGE2DMULTISAMPLEPROC)(GLenum target, GLsizei samples, GLenum
       internalformat, GLsizei width, GLsizei height, GLboolean fixedsamplelocations);
2499 typedef void (GLAD_API_PTR *PFNGLTEXIMAGE3DPROC)(GLenum target, GLint level, GLint internalformat,
       GLsizei width, GLsizei height, GLsizei depth, GLint border, GLenum format, GLenum type, const void *
       pixels);
2500 typedef void (GLAD_API_PTR *PFNGLTEXIMAGE3DMULTISAMPLEPROC)(GLenum target, GLsizei samples, GLenum
       internalformat, GLsizei width, GLsizei height, GLsizei depth, GLboolean fixedsamplelocations);
2501 typedef void (GLAD_API_PTR *PFNGLTEXPARAMETERIIVPROC)(GLenum target, GLenum pname, const GLint *
       params);
2502 typedef void (GLAD_API_PTR *PFNGLTEXPARAMETERIUIVPROC)(GLenum target, GLenum pname, const GLuint *
       params);
2503 typedef void (GLAD_API_PTR *PFNGLTEXPARAMETERFPROC)(GLenum target, GLenum pname, GLfloat param);
2504 typedef void (GLAD_API_PTR *PFNGLTEXPARAMETERFVPROC)(GLenum target, GLenum pname, const GLfloat *
       params);
2505 typedef void (GLAD_API_PTR *PFNGLTEXPARAMETERIPROC)(GLenum target, GLenum pname, GLint param);
2506 typedef void (GLAD_API_PTR *PFNGLTEXPARAMETERIVPROC)(GLenum target, GLenum pname, const GLint *
       params);
2507 typedef void (GLAD_API_PTR *PFNGLTEXSUBIMAGE1DPROC)(GLenum target, GLint level, GLint xoffset, GLsizei
       width, GLenum format, GLenum type, const void * pixels);
2508 typedef void (GLAD_API_PTR *PFNGLTEXSUBIMAGE2DPROC)(GLenum target, GLint level, GLint xoffset, GLint
       yoffset, GLsizei width, GLsizei height, GLenum format, GLenum type, const void * pixels);
2509 typedef void (GLAD_API_PTR *PFNGLTEXSUBIMAGE3DPROC)(GLenum target, GLint level, GLint xoffset, GLint
       yoffset, GLint zoffset, GLsizei width, GLsizei height, GLsizei depth, GLenum format, GLenum type,
       const void * pixels);
2510 typedef void (GLAD_API_PTR *PFNGLTRANSFORMFEEDBACKVARYINGSPROC)(GLuint program, GLsizei count, const
       GLchar *const* varyings, GLenum bufferMode);
2511 typedef void (GLAD_API_PTR *PFNGLTRANSLATEDPROC)(GLdouble x, GLdouble y, GLdouble z);
2512 typedef void (GLAD_API_PTR *PFNGLTRANSLATEFPROC)(GLfloat x, GLfloat y, GLfloat z);
2513 typedef void (GLAD_API_PTR *PFNGLUNIFORM1FPROC)(GLint location, GLfloat v0);
2514 typedef void (GLAD_API_PTR *PFNGLUNIFORM1FVPROC)(GLint location, GLsizei count, const GLfloat * value);
2515 typedef void (GLAD_API_PTR *PFNGLUNIFORM1IPROC)(GLint location, GLint v0);
2516 typedef void (GLAD_API_PTR *PFNGLUNIFORM1IVPROC)(GLint location, GLsizei count, const GLint * value);
2517 typedef void (GLAD_API_PTR *PFNGLUNIFORM1UIPROC)(GLint location, GLuint v0);
2518 typedef void (GLAD_API_PTR *PFNGLUNIFORM1UIVPROC)(GLint location, GLsizei count, const GLuint * value);
2519 typedef void (GLAD_API_PTR *PFNGLUNIFORM2FPROC)(GLint location, GLfloat v0, GLfloat v1);
2520 typedef void (GLAD_API_PTR *PFNGLUNIFORM2FVPROC)(GLint location, GLsizei count, const GLfloat * value);
2521 typedef void (GLAD_API_PTR *PFNGLUNIFORM2IPROC)(GLint location, GLint v0, GLint v1);
2522 typedef void (GLAD_API_PTR *PFNGLUNIFORM2IVPROC)(GLint location, GLsizei count, const GLint * value);
2523 typedef void (GLAD_API_PTR *PFNGLUNIFORM2UIPROC)(GLint location, GLuint v0, GLuint v1);
2524 typedef void (GLAD_API_PTR *PFNGLUNIFORM2UIVPROC)(GLint location, GLsizei count, const GLuint * value);
2525 typedef void (GLAD_API_PTR *PFNGLUNIFORM3FPROC)(GLint location, GLfloat v0, GLfloat v1, GLfloat v2);
2526 typedef void (GLAD_API_PTR *PFNGLUNIFORM3FVPROC)(GLint location, GLsizei count, const GLfloat * value);
2527 typedef void (GLAD_API_PTR *PFNGLUNIFORM3IPROC)(GLint location, GLint v0, GLint v1, GLint v2);
2528 typedef void (GLAD_API_PTR *PFNGLUNIFORM3IVPROC)(GLint location, GLsizei count, const GLint * value);
2529 typedef void (GLAD_API_PTR *PFNGLUNIFORM3UIPROC)(GLint location, GLuint v0, GLuint v1, GLuint v2);
2530 typedef void (GLAD_API_PTR *PFNGLUNIFORM3UIVPROC)(GLint location, GLsizei count, const GLuint * value);
2531 typedef void (GLAD_API_PTR *PFNGLUNIFORM4FPROC)(GLint location, GLfloat v0, GLfloat v1, GLfloat v2,
       GLfloat v3);
2532 typedef void (GLAD_API_PTR *PFNGLUNIFORM4FVPROC)(GLint location, GLsizei count, const GLfloat * value);
2533 typedef void (GLAD_API_PTR *PFNGLUNIFORM4IPROC)(GLint location, GLint v0, GLint v1, GLint v2, GLint
       v3);
2534 typedef void (GLAD_API_PTR *PFNGLUNIFORM4IVPROC)(GLint location, GLsizei count, const GLint * value);
2535 typedef void (GLAD_API_PTR *PFNGLUNIFORM4UIPROC)(GLint location, GLuint v0, GLuint v1, GLuint v2,
       GLuint v3);
2536 typedef void (GLAD_API_PTR *PFNGLUNIFORM4UIVPROC)(GLint location, GLsizei count, const GLuint * value);
2537 typedef void (GLAD_API_PTR *PFNGLUNIFORMBLOCKBINDINGPROC)(GLuint program, GLuint uniformBlockIndex,
       GLuint uniformBlockBinding);
2538 typedef void (GLAD_API_PTR *PFNGLUNIFORMMATRIX2FVPROC)(GLint location, GLsizei count, GLboolean
       transpose, const GLfloat * value);
2539 typedef void (GLAD_API_PTR *PFNGLUNIFORMMATRIX2X3FVPROC)(GLint location, GLsizei count, GLboolean
       transpose, const GLfloat * value);
2540 typedef void (GLAD_API_PTR *PFNGLUNIFORMMATRIX2X4FVPROC)(GLint location, GLsizei count, GLboolean
       transpose, const GLfloat * value);
2541 typedef void (GLAD_API_PTR *PFNGLUNIFORMMATRIX3FVPROC)(GLint location, GLsizei count, GLboolean
       transpose, const GLfloat * value);
2542 typedef void (GLAD_API_PTR *PFNGLUNIFORMMATRIX3X2FVPROC)(GLint location, GLsizei count, GLboolean
       transpose, const GLfloat * value);
```

```
2543 typedef void (GLAD_API_PTR *PFNGLUNIFORMMATRIX3X4FVPROC)(GLint location, GLsizei count, GLboolean
       transpose, const GLfloat * value);
2544 typedef void (GLAD_API_PTR *PFNGLUNIFORMMATRIX4FVPROC)(GLint location, GLsizei count, GLboolean
       transpose, const GLfloat * value);
2545 typedef void (GLAD_API_PTR *PFNGLUNIFORMMATRIX4X2FVPROC)(GLint location, GLsizei count, GLboolean
       transpose, const GLfloat * value);
2546 typedef void (GLAD_API_PTR *PFNGLUNIFORMMATRIX4X3FVPROC)(GLint location, GLsizei count, GLboolean
       transpose, const GLfloat * value);
2547 typedef GLboolean (GLAD_API_PTR *PFNGLUNMAPBUFFERPROC)(GLenum target);
2548 typedef void (GLAD_API_PTR *PFNGLUSEPROGRAMPROC)(GLuint program);
2549 typedef void (GLAD_API_PTR *PFNGLVALIDATEPROGRAMPROC)(GLuint program);
2550 typedef void (GLAD_API_PTR *PFNGLVERTEX2DPROC)(GLdouble x, GLdouble y);
2551 typedef void (GLAD_API_PTR *PFNGLVERTEX2DVPROC)(const GLdouble * v);
2552 typedef void (GLAD_API_PTR *PFNGLVERTEX2FPROC)(GLfloat x, GLfloat y);
2553 typedef void (GLAD_API_PTR *PFNGLVERTEX2FVPROC)(const GLfloat * v);
2554 typedef void (GLAD_API_PTR *PFNGLVERTEX2IPROC)(GLint x, GLint y);
2555 typedef void (GLAD_API_PTR *PFNGLVERTEX2IVPROC)(const GLint * v);
2556 typedef void (GLAD_API_PTR *PFNGLVERTEX2SPROC)(GLshort x, GLshort y);
2557 typedef void (GLAD_API_PTR *PFNGLVERTEX2SVPROC)(const GLshort * v);
2558 typedef void (GLAD_API_PTR *PFNGLVERTEX3DPROC)(GLdouble x, GLdouble y, GLdouble z);
2559 typedef void (GLAD_API_PTR *PFNGLVERTEX3DVPROC)(const GLdouble * v);
2560 typedef void (GLAD_API_PTR *PFNGLVERTEX3FPROC)(GLfloat x, GLfloat y, GLfloat z);
2561 typedef void (GLAD_API_PTR *PFNGLVERTEX3FVPROC)(const GLfloat * v);
2562 typedef void (GLAD_API_PTR *PFNGLVERTEX3IPROC)(GLint x, GLint y, GLint z);
2563 typedef void (GLAD_API_PTR *PFNGLVERTEX3IVPROC)(const GLint * v);
2564 typedef void (GLAD_API_PTR *PFNGLVERTEX3SPROC)(GLshort x, GLshort y, GLshort z);
2565 typedef void (GLAD_API_PTR *PFNGLVERTEX3SVPROC)(const GLshort * v);
2566 typedef void (GLAD_API_PTR *PFNGLVERTEX4DPROC)(GLdouble x, GLdouble y, GLdouble z, GLdouble w);
2567 typedef void (GLAD_API_PTR *PFNGLVERTEX4DVPROC)(const GLdouble * v);
2568 typedef void (GLAD_API_PTR *PFNGLVERTEX4FPROC)(GLfloat x, GLfloat y, GLfloat z, GLfloat w);
2569 typedef void (GLAD_API_PTR *PFNGLVERTEX4FVPROC)(const GLfloat * v);
2570 typedef void (GLAD_API_PTR *PFNGLVERTEX4IPROC)(GLint x, GLint y, GLint z, GLint w);
2571 typedef void (GLAD_API_PTR *PFNGLVERTEX4IVPROC)(const GLint * v);
2572 typedef void (GLAD_API_PTR *PFNGLVERTEX4SPROC)(GLshort x, GLshort y, GLshort z, GLshort w);
2573 typedef void (GLAD_API_PTR *PFNGLVERTEX4SVPROC)(const GLshort * v);
2574 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB1DPROC)(GLuint index, GLdouble x);
2575 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB1DVPROC)(GLuint index, const GLdouble * v);
2576 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB1FPROC)(GLuint index, GLfloat x);
2577 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB1FVPROC)(GLuint index, const GLfloat * v);
2578 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB1SPROC)(GLuint index, GLshort x);
2579 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB1SVPROC)(GLuint index, const GLshort * v);
2580 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB2DPROC)(GLuint index, GLdouble x, GLdouble y);
2581 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB2DVPROC)(GLuint index, const GLdouble * v);
2582 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB2FPROC)(GLuint index, GLfloat x, GLfloat y);
2583 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB2FVPROC)(GLuint index, const GLfloat * v);
2584 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB2SPROC)(GLuint index, GLshort x, GLshort y);
2585 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB2SVPROC)(GLuint index, const GLshort * v);
2586 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB3DPROC)(GLuint index, GLdouble x, GLdouble y, GLdouble z);
2587 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB3DVPROC)(GLuint index, const GLdouble * v);
2588 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB3FPROC)(GLuint index, GLfloat x, GLfloat y, GLfloat z);
2589 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB3FVPROC)(GLuint index, const GLfloat * v);
2590 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB3SPROC)(GLuint index, GLshort x, GLshort y, GLshort z);
2591 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB3SVPROC)(GLuint index, const GLshort * v);
2592 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4NBVPROC)(GLuint index, const GLbyte * v);
2593 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4NIVPROC)(GLuint index, const GLint * v);
2594 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4NSVPROC)(GLuint index, const GLshort * v);
2595 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4NUBPROC)(GLuint index, GLubyte x, GLubyte y, GLubyte z,
       GLubyte w);
2596 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4NUBVPROC)(GLuint index, const GLubyte * v);
2597 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4NUIVPROC)(GLuint index, const GLuint * v);
2598 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4NUSVPROC)(GLuint index, const GLushort * v);
2599 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4BVPROC)(GLuint index, const GLbyte * v);
2600 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4DPROC)(GLuint index, GLdouble x, GLdouble y, GLdouble z,
       GLdouble w);
2601 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4DVPROC)(GLuint index, const GLdouble * v);
2602 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4FPROC)(GLuint index, GLfloat x, GLfloat y, GLfloat z,
       GLfloat w);
2603 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4FVPROC)(GLuint index, const GLfloat * v);
2604 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4IVPROC)(GLuint index, const GLint * v);
2605 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4SPROC)(GLuint index, GLshort x, GLshort y, GLshort z,
       GLshort w);
2606 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4SVPROC)(GLuint index, const GLshort * v);
2607 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4UBVPROC)(GLuint index, const GLubyte * v);
2608 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4UIVPROC)(GLuint index, const GLuint * v);
2609 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4USVPROC)(GLuint index, const GLushort * v);
2610 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBDIVISORPROC)(GLuint index, GLuint divisor);
2611 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI1IPROC)(GLuint index, GLint x);
2612 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI1IVPROC)(GLuint index, const GLint * v);
2613 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI1UIPROC)(GLuint index, GLuint x);
2614 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI1UIVPROC)(GLuint index, const GLuint * v);
2615 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI2IPROC)(GLuint index, GLint x, GLint y);
2616 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI2IVPROC)(GLuint index, const GLint * v);
2617 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI2UIPROC)(GLuint index, GLuint x, GLuint y);
2618 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI2UIVPROC)(GLuint index, const GLuint * v);
2619 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI3IPROC)(GLuint index, GLint x, GLint y, GLint z);
2620 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI3IVPROC)(GLuint index, const GLint * v);
2621 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI3UIPROC)(GLuint index, GLuint x, GLuint y, GLuint z);
```

```
2622 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI3UIVPROC)(GLuint index, const GLuint * v);
2623 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI4BVPROC)(GLuint index, const GLbyte * v);
2624 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI4IPROC)(GLuint index, GLint x, GLint y, GLint z, GLint
       w);
2625 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI4IVPROC)(GLuint index, const GLint * v);
2626 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI4SVPROC)(GLuint index, const GLshort * v);
2627 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI4UBVPROC)(GLuint index, const GLubyte * v);
2628 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI4UIPROC)(GLuint index, GLuint x, GLuint y, GLuint z,
       GLuint w);
2629 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI4UIVPROC)(GLuint index, const GLuint * v);
2630 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBI4USVPROC)(GLuint index, const GLushort * v);
2631 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBIPOINTERPROC)(GLuint index, GLint size, GLenum type,
       GLsizei stride, const void * pointer);
2632 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBP1UIPROC)(GLuint index, GLenum type, GLboolean normalized,
       GLuint value);
2633 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBP1UIVPROC)(GLuint index, GLenum type, GLboolean
       normalized, const GLuint * value);
2634 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBP2UIPROC)(GLuint index, GLenum type, GLboolean normalized,
       GLuint value);
2635 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBP2UIVPROC)(GLuint index, GLenum type, GLboolean
       normalized, const GLuint * value);
2636 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBP3UIPROC)(GLuint index, GLenum type, GLboolean normalized,
       GLuint value);
2637 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBP3UIVPROC)(GLuint index, GLenum type, GLboolean
       normalized, const GLuint * value);
2638 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBP4UIPROC)(GLuint index, GLenum type, GLboolean normalized,
       GLuint value);
2639 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBP4UIVPROC)(GLuint index, GLenum type, GLboolean
       normalized, const GLuint * value);
2640 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBPOINTERPROC)(GLuint index, GLint size, GLenum type,
       GLboolean normalized, GLsizei stride, const void * pointer);
2641 typedef void (GLAD_API_PTR *PFNGLVERTEXP2UIPROC)(GLenum type, GLuint value);
2642 typedef void (GLAD_API_PTR *PFNGLVERTEXP2UIVPROC)(GLenum type, const GLuint * value);
2643 typedef void (GLAD_API_PTR *PFNGLVERTEXP3UIPROC)(GLenum type, GLuint value);
2644 typedef void (GLAD_API_PTR *PFNGLVERTEXP3UIVPROC)(GLenum type, const GLuint * value);
2645 typedef void (GLAD_API_PTR *PFNGLVERTEXP4UIPROC)(GLenum type, GLuint value);
2646 typedef void (GLAD_API_PTR *PFNGLVERTEXP4UIVPROC)(GLenum type, const GLuint * value);
2647 typedef void (GLAD_API_PTR *PFNGLVERTEXPOINTERPROC)(GLint size, GLenum type, GLsizei stride, const void
       * pointer);
2648 typedef void (GLAD_API_PTR *PFNGLVIEWPORTPROC)(GLint x, GLint y, GLsizei width, GLsizei height);
2649 typedef void (GLAD_API_PTR *PFNGLWAITSYNCPROC)(GLsync sync, GLbitfield flags, GLuint64 timeout);
2650 typedef void (GLAD_API_PTR *PFNGLWINDOWPOS2DPROC)(GLdouble x, GLdouble y);
2651 typedef void (GLAD_API_PTR *PFNGLWINDOWPOS2DVPROC)(const GLdouble * v);
2652 typedef void (GLAD_API_PTR *PFNGLWINDOWPOS2FPROC)(GLfloat x, GLfloat y);
2653 typedef void (GLAD_API_PTR *PFNGLWINDOWPOS2FVPROC)(const GLfloat * v);
2654 typedef void (GLAD_API_PTR *PFNGLWINDOWPOS2IPROC)(GLint x, GLint y);
2655 typedef void (GLAD_API_PTR *PFNGLWINDOWPOS2IVPROC)(const GLint * v);
2656 typedef void (GLAD_API_PTR *PFNGLWINDOWPOS2SPROC)(GLshort x, GLshort y);
2657 typedef void (GLAD_API_PTR *PFNGLWINDOWPOS2SVPROC)(const GLshort * v);
2658 typedef void (GLAD_API_PTR *PFNGLWINDOWPOS3DPROC)(GLdouble x, GLdouble y, GLdouble z);
2659 typedef void (GLAD_API_PTR *PFNGLWINDOWPOS3DVPROC)(const GLdouble * v);
2660 typedef void (GLAD_API_PTR *PFNGLWINDOWPOS3FPROC)(GLfloat x, GLfloat y, GLfloat z);
2661 typedef void (GLAD_API_PTR *PFNGLWINDOWPOS3FVPROC)(const GLfloat * v);
2662 typedef void (GLAD_API_PTR *PFNGLWINDOWPOS3IPROC)(GLint x, GLint y, GLint z);
2663 typedef void (GLAD_API_PTR *PFNGLWINDOWPOS3IVPROC)(const GLint * v);
2664 typedef void (GLAD_API_PTR *PFNGLWINDOWPOS3SPROC)(GLshort x, GLshort y, GLshort z);
2665 typedef void (GLAD_API_PTR *PFNGLWINDOWPOS3SVPROC)(const GLshort * v);
2666
2667 GLAD_API_CALL PFNGLACCUMPROC glad_glAccum;
2668 #define glAccum glad_glAccum
2669 GLAD_API_CALL PFNGLACTIVETEXTUREPROC glad_glActiveTexture;
2670 #define glActiveTexture glad_glActiveTexture
2671 GLAD_API_CALL PFNGLALPHAFUNCPROC glad_glAlphaFunc;
2672 #define glAlphaFunc glad_glAlphaFunc
2673 GLAD_API_CALL PFNGLARETEXTURESRESIDENTPROC glad_glAreTexturesResident;
2674 #define glAreTexturesResident glad_glAreTexturesResident
2675 GLAD_API_CALL PFNGLARRAYELEMENTPROC glad_glArrayElement;
2676 #define glArrayElement glad_glArrayElement
2677 GLAD_API_CALL PFNGLATTACHSHADERPROC glad_glAttachShader;
2678 #define glAttachShader glad_glAttachShader
2679 GLAD_API_CALL PFNGLBEGINPROC glad_glBegin;
2680 #define glBegin glad_glBegin
2681 GLAD_API_CALL PFNGLBEGINCONDITIONALRENDERPROC glad_glBeginConditionalRender;
2682 #define glBeginConditionalRender glad_glBeginConditionalRender
2683 GLAD_API_CALL PFNGLBEGINQUERYPROC glad_glBeginQuery;
2684 #define glBeginQuery glad_glBeginQuery
2685 GLAD_API_CALL PFNGLBEGINTRANSFORMFEEDBACKPROC glad_glBeginTransformFeedback;
2686 #define glBeginTransformFeedback glad_glBeginTransformFeedback
2687 GLAD_API_CALL PFNGLBINDATTRIBLOCATIONPROC glad_glBindAttribLocation;
2688 #define glBindAttribLocation glad_glBindAttribLocation
2689 GLAD_API_CALL PFNGLBINDBUFFERPROC glad_glBindBuffer;
2690 #define glBindBuffer glad_glBindBuffer
2691 GLAD_API_CALL PFNGLBINDBUFFERBASEPROC glad_glBindBufferBase;
2692 #define glBindBufferBase glad_glBindBufferBase
2693 GLAD_API_CALL PFNGLBINDBUFFERRANGEPROC glad_glBindBufferRange;
2694 #define glBindBufferRange glad_glBindBufferRange
2695 GLAD_API_CALL PFNGLBINDFRAGDATALOCATIONPROC glad_glBindFragDataLocation;
```

```
2696 #define glBindFragDataLocation glad_glBindFragDataLocation
2697 GLAD_API_CALL PFNGLBINDFRAGDATALOCATIONINDEXEDPROC glad_glBindFragDataLocationIndexed;
2698 #define glBindFragDataLocationIndexed glad_glBindFragDataLocationIndexed
2699 GLAD_API_CALL PFNGLBINDFRAMEBUFFERPROC glad_glBindFramebuffer;
2700 #define glBindFramebuffer glad_glBindFramebuffer
2701 GLAD_API_CALL PFNGLBINDRENDERBUFFERPROC glad_glBindRenderbuffer;
2702 #define glBindRenderbuffer glad_glBindRenderbuffer
2703 GLAD_API_CALL PFNGLBINDSAMPLERPROC glad_glBindSampler;
2704 #define glBindSampler glad_glBindSampler
2705 GLAD_API_CALL PFNGLBINDTEXTUREPROC glad_glBindTexture;
2706 #define glBindTexture glad_glBindTexture
2707 GLAD_API_CALL PFNGLBINDVERTEXARRAYPROC glad_glBindVertexArray;
2708 #define glBindVertexArray glad_glBindVertexArray
2709 GLAD_API_CALL PFNGLBITMAPPROC glad_glBitmap;
2710 #define glBitmap glad_glBitmap
2711 GLAD_API_CALL PFNGLBLENDCOLORPROC glad_glBlendColor;
2712 #define glBlendColor glad_glBlendColor
2713 GLAD_API_CALL PFNGLBLENDEQUATIONPROC glad_glBlendEquation;
2714 #define glBlendEquation glad_glBlendEquation
2715 GLAD_API_CALL PFNGLBLENDEQUATIONSEPARATEPROC glad_glBlendEquationSeparate;
2716 #define glBlendEquationSeparate glad_glBlendEquationSeparate
2717 GLAD_API_CALL PFNGLBLENDFUNCPROC glad_glBlendFunc;
2718 #define glBlendFunc glad_glBlendFunc
2719 GLAD_API_CALL PFNGLBLENDFUNCSEPARATEPROC glad_glBlendFuncSeparate;
2720 #define glBlendFuncSeparate glad_glBlendFuncSeparate
2721 GLAD_API_CALL PFNGLBLITFRAMEBUFFERPROC glad_glBlitFramebuffer;
2722 #define glBlitFramebuffer glad_glBlitFramebuffer
2723 GLAD_API_CALL PFNGLBUFFERDATAPROC glad_glBufferData;
2724 #define glBufferData glad_glBufferData
2725 GLAD_API_CALL PFNGLBUFFERSUBDATAPROC glad_glBufferSubData;
2726 #define glBufferSubData glad_glBufferSubData
2727 GLAD_API_CALL PFNGLCALLLISTPROC glad_glCallList;
2728 #define glCallList glad_glCallList
2729 GLAD_API_CALL PFNGLCALLLISTSPROC glad_glCallLists;
2730 #define glCallLists glad_glCallLists
2731 GLAD_API_CALL PFNGLCHECKFRAMEBUFFERSTATUSPROC glad_glCheckFramebufferStatus;
2732 #define glCheckFramebufferStatus glad_glCheckFramebufferStatus
2733 GLAD_API_CALL PFNGLCLAMPCOLORPROC glad_glClampColor;
2734 #define glClampColor glad_glClampColor
2735 GLAD_API_CALL PFNGLCLEARPROC glad_glClear;
2736 #define glClear glad_glClear
2737 GLAD_API_CALL PFNGLCLEARACCUMPROC glad_glClearAccum;
2738 #define glClearAccum glad_glClearAccum
2739 GLAD_API_CALL PFNGLCLEARBUFFERFIPROC glad_glClearBufferfi;
2740 #define glClearBufferfi glad_glClearBufferfi
2741 GLAD_API_CALL PFNGLCLEARBUFFERFVPROC glad_glClearBufferfv;
2742 #define glClearBufferfv glad_glClearBufferfv
2743 GLAD_API_CALL PFNGLCLEARBUFFERIVPROC glad_glClearBufferiv;
2744 #define glClearBufferiv glad_glClearBufferiv
2745 GLAD_API_CALL PFNGLCLEARBUFFERUIVPROC glad_glClearBufferuiv;
2746 #define glClearBufferuiv glad_glClearBufferuiv
2747 GLAD_API_CALL PFNGLCLEARCOLORPROC glad_glClearColor;
2748 #define glClearColor glad_glClearColor
2749 GLAD_API_CALL PFNGLCLEARDEPTHPROC glad_glClearDepth;
2750 #define glClearDepth glad_glClearDepth
2751 GLAD_API_CALL PFNGLCLEARINDEXPROC glad_glClearIndex;
2752 #define glClearIndex glad_glClearIndex
2753 GLAD_API_CALL PFNGLCLEARSTENCILPROC glad_glClearStencil;
2754 #define glClearStencil glad_glClearStencil
2755 GLAD_API_CALL PFNGLCLIENTACTIVETEXTUREPROC glad_glClientActiveTexture;
2756 #define glClientActiveTexture glad_glClientActiveTexture
2757 GLAD_API_CALL PFNGLCLIENTWAITSYNCPROC glad_glClientWaitSync;
2758 #define glClientWaitSync glad_glClientWaitSync
2759 GLAD_API_CALL PFNGLCLIPPLANEPROC glad_glClipPlane;
2760 #define glClipPlane glad_glClipPlane
2761 GLAD_API_CALL PFNGLCOLOR3BPROC glad_glColor3b;
2762 #define glColor3b glad_glColor3b
2763 GLAD_API_CALL PFNGLCOLOR3BVPROC glad_glColor3bv;
2764 #define glColor3bv glad_glColor3bv
2765 GLAD_API_CALL PFNGLCOLOR3DPROC glad_glColor3d;
2766 #define glColor3d glad_glColor3d
2767 GLAD_API_CALL PFNGLCOLOR3DVPROC glad_glColor3dv;
2768 #define glColor3dv glad_glColor3dv
2769 GLAD_API_CALL PFNGLCOLOR3FPROC glad_glColor3f;
2770 #define glColor3f glad_glColor3f
2771 GLAD_API_CALL PFNGLCOLOR3FVPROC glad_glColor3fv;
2772 #define glColor3fv glad_glColor3fv
2773 GLAD_API_CALL PFNGLCOLOR3IPROC glad_glColor3i;
2774 #define glColor3i glad_glColor3i
2775 GLAD_API_CALL PFNGLCOLOR3IVPROC glad_glColor3iv;
2776 #define glColor3iv glad_glColor3iv
2777 GLAD_API_CALL PFNGLCOLOR3SPROC glad_glColor3s;
2778 #define glColor3s glad_glColor3s
2779 GLAD_API_CALL PFNGLCOLOR3SVPROC glad_glColor3sv;
2780 #define glColor3sv glad_glColor3sv
2781 GLAD_API_CALL PFNGLCOLOR3UBPROC glad_glColor3ub;
2782 #define glColor3ub glad_glColor3ub
```

```
2783 GLAD_API_CALL PFNGLCOLOR3UBVPROC glad_glColor3ubv;
2784 #define glColor3ubv glad_glColor3ubv
2785 GLAD_API_CALL PFNGLCOLOR3UIPROC glad_glColor3ui;
2786 #define glColor3ui glad_glColor3ui
2787 GLAD_API_CALL PFNGLCOLOR3UIVPROC glad_glColor3uiv;
2788 #define glColor3uiv glad_glColor3uiv
2789 GLAD_API_CALL PFNGLCOLOR3USPROC glad_glColor3us;
2790 #define glColor3us glad_glColor3us
2791 GLAD_API_CALL PFNGLCOLOR3USVPROC glad_glColor3usv;
2792 #define glColor3usv glad_glColor3usv
2793 GLAD_API_CALL PFNGLCOLOR4BPROC glad_glColor4b;
2794 #define glColor4b glad_glColor4b
2795 GLAD_API_CALL PFNGLCOLOR4BVPROC glad_glColor4bv;
2796 #define glColor4bv glad_glColor4bv
2797 GLAD_API_CALL PFNGLCOLOR4DPROC glad_glColor4d;
2798 #define glColor4d glad_glColor4d
2799 GLAD_API_CALL PFNGLCOLOR4DVPROC glad_glColor4dv;
2800 #define glColor4dv glad_glColor4dv
2801 GLAD_API_CALL PFNGLCOLOR4FPROC glad_glColor4f;
2802 #define glColor4f glad_glColor4f
2803 GLAD_API_CALL PFNGLCOLOR4FVPROC glad_glColor4fv;
2804 #define glColor4fv glad_glColor4fv
2805 GLAD_API_CALL PFNGLCOLOR4IPROC glad_glColor4i;
2806 #define glColor4i glad_glColor4i
2807 GLAD_API_CALL PFNGLCOLOR4IVPROC glad_glColor4iv;
2808 #define glColor4iv glad_glColor4iv
2809 GLAD_API_CALL PFNGLCOLOR4SPROC glad_glColor4s;
2810 #define glColor4s glad_glColor4s
2811 GLAD_API_CALL PFNGLCOLOR4SVPROC glad_glColor4sv;
2812 #define glColor4sv glad_glColor4sv
2813 GLAD_API_CALL PFNGLCOLOR4UBPROC glad_glColor4ub;
2814 #define glColor4ub glad_glColor4ub
2815 GLAD_API_CALL PFNGLCOLOR4UBVPROC glad_glColor4ubv;
2816 #define glColor4ubv glad_glColor4ubv
2817 GLAD_API_CALL PFNGLCOLOR4UIPROC glad_glColor4ui;
2818 #define glColor4ui glad_glColor4ui
2819 GLAD_API_CALL PFNGLCOLOR4UIVPROC glad_glColor4uiv;
2820 #define glColor4uiv glad_glColor4uiv
2821 GLAD_API_CALL PFNGLCOLOR4USPROC glad_glColor4us;
2822 #define glColor4us glad_glColor4us
2823 GLAD_API_CALL PFNGLCOLOR4USVPROC glad_glColor4usv;
2824 #define glColor4usv glad_glColor4usv
2825 GLAD_API_CALL PFNGLCOLORMASKPROC glad_glColorMask;
2826 #define glColorMask glad_glColorMask
2827 GLAD_API_CALL PFNGLCOLORMASKIPROC glad_glColorMaski;
2828 #define glColorMaski glad_glColorMaski
2829 GLAD_API_CALL PFNGLCOLORMATERIALPROC glad_glColorMaterial;
2830 #define glColorMaterial glad_glColorMaterial
2831 GLAD_API_CALL PFNGLCOLORP3UIPROC glad_glColorP3ui;
2832 #define glColorP3ui glad_glColorP3ui
2833 GLAD_API_CALL PFNGLCOLORP3UIVPROC glad_glColorP3uiv;
2834 #define glColorP3uiv glad_glColorP3uiv
2835 GLAD_API_CALL PFNGLCOLORP4UIPROC glad_glColorP4ui;
2836 #define glColorP4ui glad_glColorP4ui
2837 GLAD_API_CALL PFNGLCOLORP4UIVPROC glad_glColorP4uiv;
2838 #define glColorP4uiv glad_glColorP4uiv
2839 GLAD_API_CALL PFNGLCOLORPOINTERPROC glad_glColorPointer;
2840 #define glColorPointer glad_glColorPointer
2841 GLAD_API_CALL PFNGLCOMPILESHADERPROC glad_glCompileShader;
2842 #define glCompileShader glad_glCompileShader
2843 GLAD_API_CALL PFNGLCOMPRESSEDTEXIMAGE1DPROC glad_glCompressedTexImage1D;
2844 #define glCompressedTexImage1D glad_glCompressedTexImage1D
2845 GLAD_API_CALL PFNGLCOMPRESSEDTEXIMAGE2DPROC glad_glCompressedTexImage2D;
2846 #define glCompressedTexImage2D glad_glCompressedTexImage2D
2847 GLAD_API_CALL PFNGLCOMPRESSEDTEXIMAGE3DPROC glad_glCompressedTexImage3D;
2848 #define glCompressedTexImage3D glad_glCompressedTexImage3D
2849 GLAD_API_CALL PFNGLCOMPRESSEDTEXSUBIMAGE1DPROC glad_glCompressedTexSubImage1D;
2850 #define glCompressedTexSubImage1D glad_glCompressedTexSubImage1D
2851 GLAD_API_CALL PFNGLCOMPRESSEDTEXSUBIMAGE2DPROC glad_glCompressedTexSubImage2D;
2852 #define glCompressedTexSubImage2D glad_glCompressedTexSubImage2D
2853 GLAD_API_CALL PFNGLCOMPRESSEDTEXSUBIMAGE3DPROC glad_glCompressedTexSubImage3D;
2854 #define glCompressedTexSubImage3D glad_glCompressedTexSubImage3D
2855 GLAD_API_CALL PFNGLCOPYBUFFERSUBDATAPROC glad_glCopyBufferSubData;
2856 #define glCopyBufferSubData glad_glCopyBufferSubData
2857 GLAD_API_CALL PFNGLCOPYPIXELSPROC glad_glCopyPixels;
2858 #define glCopyPixels glad_glCopyPixels
2859 GLAD_API_CALL PFNGLCOPYTEXIMAGE1DPROC glad_glCopyTexImage1D;
2860 #define glCopyTexImage1D glad_glCopyTexImage1D
2861 GLAD_API_CALL PFNGLCOPYTEXIMAGE2DPROC glad_glCopyTexImage2D;
2862 #define glCopyTexImage2D glad_glCopyTexImage2D
2863 GLAD_API_CALL PFNGLCOPYTEXSUBIMAGE1DPROC glad_glCopyTexSubImage1D;
2864 #define glCopyTexSubImage1D glad_glCopyTexSubImage1D
2865 GLAD_API_CALL PFNGLCOPYTEXSUBIMAGE2DPROC glad_glCopyTexSubImage2D;
2866 #define glCopyTexSubImage2D glad_glCopyTexSubImage2D
2867 GLAD_API_CALL PFNGLCOPYTEXSUBIMAGE3DPROC glad_glCopyTexSubImage3D;
2868 #define glCopyTexSubImage3D glad_glCopyTexSubImage3D
2869 GLAD_API_CALL PFNGLCREATEPROGRAMPROC glad_glCreateProgram;
```

```
2870 #define glCreateProgram glad_glCreateProgram
2871 GLAD_API_CALL PFNGLCREATESHADERPROC glad_glCreateShader;
2872 #define glCreateShader glad_glCreateShader
2873 GLAD_API_CALL PFNGLCULLFACEPROC glad_glCullFace;
2874 #define glCullFace glad_glCullFace
2875 GLAD_API_CALL PFNGLDEBUGMESSAGECALLBACKPROC glad_glDebugMessageCallback;
2876 #define glDebugMessageCallback glad_glDebugMessageCallback
2877 GLAD_API_CALL PFNGLDEBUGMESSAGECONTROLPROC glad_glDebugMessageControl;
2878 #define glDebugMessageControl glad_glDebugMessageControl
2879 GLAD_API_CALL PFNGLDEBUGMESSAGEINSERTPROC glad_glDebugMessageInsert;
2880 #define glDebugMessageInsert glad_glDebugMessageInsert
2881 GLAD_API_CALL PFNGLDELETEBUFFERSPROC glad_glDeleteBuffers;
2882 #define glDeleteBuffers glad_glDeleteBuffers
2883 GLAD_API_CALL PFNGLDELETEFRAMEBUFFERSPROC glad_glDeleteFramebuffers;
2884 #define glDeleteFramebuffers glad_glDeleteFramebuffers
2885 GLAD_API_CALL PFNGLDELETELISTSPROC glad_glDeleteLists;
2886 #define glDeleteLists glad_glDeleteLists
2887 GLAD_API_CALL PFNGLDELETEPROGRAMPROC glad_glDeleteProgram;
2888 #define glDeleteProgram glad_glDeleteProgram
2889 GLAD_API_CALL PFNGLDELETEQUERIESPROC glad_glDeleteQueries;
2890 #define glDeleteQueries glad_glDeleteQueries
2891 GLAD_API_CALL PFNGLDELETERENDERBUFFERSPROC glad_glDeleteRenderbuffers;
2892 #define glDeleteRenderbuffers glad_glDeleteRenderbuffers
2893 GLAD_API_CALL PFNGLDELETESAMPLERSPROC glad_glDeleteSamplers;
2894 #define glDeleteSamplers glad_glDeleteSamplers
2895 GLAD_API_CALL PFNGLDELETESHADERPROC glad_glDeleteShader;
2896 #define glDeleteShader glad_glDeleteShader
2897 GLAD_API_CALL PFNGLDELETESYNCPROC glad_glDeleteSync;
2898 #define glDeleteSync glad_glDeleteSync
2899 GLAD_API_CALL PFNGLDELETETEXTURESPROC glad_glDeleteTextures;
2900 #define glDeleteTextures glad_glDeleteTextures
2901 GLAD_API_CALL PFNGLDELETEVERTEXARRAYSPROC glad_glDeleteVertexArrays;
2902 #define glDeleteVertexArrays glad_glDeleteVertexArrays
2903 GLAD_API_CALL PFNGLDEPTHFUNCPROC glad_glDepthFunc;
2904 #define glDepthFunc glad_glDepthFunc
2905 GLAD_API_CALL PFNGLDEPTHMASKPROC glad_glDepthMask;
2906 #define glDepthMask glad_glDepthMask
2907 GLAD_API_CALL PFNGLDEPTHRANGEPROC glad_glDepthRange;
2908 #define glDepthRange glad_glDepthRange
2909 GLAD_API_CALL PFNGLDETACHSHADERPROC glad_glDetachShader;
2910 #define glDetachShader glad_glDetachShader
2911 GLAD_API_CALL PFNGLDISABLEPROC glad_glDisable;
2912 #define glDisable glad_glDisable
2913 GLAD_API_CALL PFNGLDISABLECLIENTSTATEPROC glad_glDisableClientState;
2914 #define glDisableClientState glad_glDisableClientState
2915 GLAD_API_CALL PFNGLDISABLEVERTEXATTRIBARRAYPROC glad_glDisableVertexAttribArray;
2916 #define glDisableVertexAttribArray glad_glDisableVertexAttribArray
2917 GLAD_API_CALL PFNGLDISABLEIPROC glad_glDisablei;
2918 #define glDisablei glad_glDisablei
2919 GLAD_API_CALL PFNGLDRAWARRAYSPROC glad_glDrawArrays;
2920 #define glDrawArrays glad_glDrawArrays
2921 GLAD_API_CALL PFNGLDRAWARRAYSINSTANCEDPROC glad_glDrawArraysInstanced;
2922 #define glDrawArraysInstanced glad_glDrawArraysInstanced
2923 GLAD_API_CALL PFNGLDRAWBUFFERPROC glad_glDrawBuffer;
2924 #define glDrawBuffer glad_glDrawBuffer
2925 GLAD_API_CALL PFNGLDRAWBUFFERSPROC glad_glDrawBuffers;
2926 #define glDrawBuffers glad_glDrawBuffers
2927 GLAD_API_CALL PFNGLDRAWELEMENTSPROC glad_glDrawElements;
2928 #define glDrawElements glad_glDrawElements
2929 GLAD_API_CALL PFNGLDRAWELEMENTSBASEVERTEXPROC glad_glDrawElementsBaseVertex;
2930 #define glDrawElementsBaseVertex glad_glDrawElementsBaseVertex
2931 GLAD_API_CALL PFNGLDRAWELEMENTSINSTANCEDPROC glad_glDrawElementsInstanced;
2932 #define glDrawElementsInstanced glad_glDrawElementsInstanced
2933 GLAD_API_CALL PFNGLDRAWELEMENTSINSTANCEDBASEVERTEXPROC glad_glDrawElementsInstancedBaseVertex;
2934 #define glDrawElementsInstancedBaseVertex glad_glDrawElementsInstancedBaseVertex
2935 GLAD_API_CALL PFNGLDRAWPIXELSPROC glad_glDrawPixels;
2936 #define glDrawPixels glad_glDrawPixels
2937 GLAD_API_CALL PFNGLDRAWRANGEELEMENTSPROC glad_glDrawRangeElements;
2938 #define glDrawRangeElements glad_glDrawRangeElements
2939 GLAD_API_CALL PFNGLDRAWRANGEELEMENTSBASEVERTEXPROC glad_glDrawRangeElementsBaseVertex;
2940 #define glDrawRangeElementsBaseVertex glad_glDrawRangeElementsBaseVertex
2941 GLAD_API_CALL PFNGLEDGEFLAGPROC glad_glEdgeFlag;
2942 #define glEdgeFlag glad_glEdgeFlag
2943 GLAD_API_CALL PFNGLEDGEFLAGPOINTERPROC glad_glEdgeFlagPointer;
2944 #define glEdgeFlagPointer glad_glEdgeFlagPointer
2945 GLAD_API_CALL PFNGLEDGEFLAGVPROC glad_glEdgeFlagv;
2946 #define glEdgeFlagv glad_glEdgeFlagv
2947 GLAD_API_CALL PFNGLENABLEPROC glad_glEnable;
2948 #define glEnable glad_glEnable
2949 GLAD_API_CALL PFNGLENABLECLIENTSTATEPROC glad_glEnableClientState;
2950 #define glEnableClientState glad_glEnableClientState
2951 GLAD_API_CALL PFNGLENABLEVERTEXATTRIBARRAYPROC glad_glEnableVertexAttribArray;
2952 #define glEnableVertexAttribArray glad_glEnableVertexAttribArray
2953 GLAD_API_CALL PFNGLENABLEIPROC glad_glEnablei;
2954 #define glEnablei glad_glEnablei
2955 GLAD_API_CALL PFNGLENDPROC glad_glEnd;
2956 #define glEnd glad_glEnd
```

```
2957 GLAD_API_CALL PFNGLENDCONDITIONALRENDERPROC glad_glEndConditionalRender;
2958 #define glEndConditionalRender glad_glEndConditionalRender
2959 GLAD_API_CALL PFNGLENDLISTPROC glad_glEndList;
2960 #define glEndList glad_glEndList
2961 GLAD_API_CALL PFNGLENDQUERYPROC glad_glEndQuery;
2962 #define glEndQuery glad_glEndQuery
2963 GLAD_API_CALL PFNGLENDTRANSFORMFEEDBACKPROC glad_glEndTransformFeedback;
2964 #define glEndTransformFeedback glad_glEndTransformFeedback
2965 GLAD_API_CALL PFNGLEVALCOORD1DPROC glad_glEvalCoord1d;
2966 #define glEvalCoord1d glad_glEvalCoord1d
2967 GLAD_API_CALL PFNGLEVALCOORD1DVPROC glad_glEvalCoord1dv;
2968 #define glEvalCoord1dv glad_glEvalCoord1dv
2969 GLAD_API_CALL PFNGLEVALCOORD1FPROC glad_glEvalCoord1f;
2970 #define glEvalCoord1f glad_glEvalCoord1f
2971 GLAD_API_CALL PFNGLEVALCOORD1FVPROC glad_glEvalCoord1fv;
2972 #define glEvalCoord1fv glad_glEvalCoord1fv
2973 GLAD_API_CALL PFNGLEVALCOORD2DPROC glad_glEvalCoord2d;
2974 #define glEvalCoord2d glad_glEvalCoord2d
2975 GLAD_API_CALL PFNGLEVALCOORD2DVPROC glad_glEvalCoord2dv;
2976 #define glEvalCoord2dv glad_glEvalCoord2dv
2977 GLAD_API_CALL PFNGLEVALCOORD2FPROC glad_glEvalCoord2f;
2978 #define glEvalCoord2f glad_glEvalCoord2f
2979 GLAD_API_CALL PFNGLEVALCOORD2FVPROC glad_glEvalCoord2fv;
2980 #define glEvalCoord2fv glad_glEvalCoord2fv
2981 GLAD_API_CALL PFNGLEVALMESH1PROC glad_glEvalMesh1;
2982 #define glEvalMesh1 glad_glEvalMesh1
2983 GLAD_API_CALL PFNGLEVALMESH2PROC glad_glEvalMesh2;
2984 #define glEvalMesh2 glad_glEvalMesh2
2985 GLAD_API_CALL PFNGLEVALPOINT1PROC glad_glEvalPoint1;
2986 #define glEvalPoint1 glad_glEvalPoint1
2987 GLAD_API_CALL PFNGLEVALPOINT2PROC glad_glEvalPoint2;
2988 #define glEvalPoint2 glad_glEvalPoint2
2989 GLAD_API_CALL PFNGLFEEDBACKBUFFERPROC glad_glFeedbackBuffer;
2990 #define glFeedbackBuffer glad_glFeedbackBuffer
2991 GLAD_API_CALL PFNGLFENCESYNCPROC glad_glFenceSync;
2992 #define glFenceSync glad_glFenceSync
2993 GLAD_API_CALL PFNGLFINISHPROC glad_glFinish;
2994 #define glFinish glad_glFinish
2995 GLAD_API_CALL PFNGLFLUSHPROC glad_glFlush;
2996 #define glFlush glad_glFlush
2997 GLAD_API_CALL PFNGLFLUSHMAPPEDBUFFERRANGEPROC glad_glFlushMappedBufferRange;
2998 #define glFlushMappedBufferRange glad_glFlushMappedBufferRange
2999 GLAD_API_CALL PFNGLFOGCOORDPOINTERPROC glad_glFogCoordPointer;
3000 #define glFogCoordPointer glad_glFogCoordPointer
3001 GLAD_API_CALL PFNGLFOGCOORDDPROC glad_glFogCoordd;
3002 #define glFogCoordd glad_glFogCoordd
3003 GLAD_API_CALL PFNGLFOGCOORDDVPROC glad_glFogCoorddv;
3004 #define glFogCoorddv glad_glFogCoorddv
3005 GLAD_API_CALL PFNGLFOGCOORDFPROC glad_glFogCoordf;
3006 #define glFogCoordf glad_glFogCoordf
3007 GLAD_API_CALL PFNGLFOGCOORDFVPROC glad_glFogCoordfv;
3008 #define glFogCoordfv glad_glFogCoordfv
3009 GLAD_API_CALL PFNGLFOGFPROC glad_glFogf;
3010 #define glFogf glad_glFogf
3011 GLAD_API_CALL PFNGLFOGFVPROC glad_glFogfv;
3012 #define glFogfv glad_glFogfv
3013 GLAD_API_CALL PFNGLFOGIPROC glad_glFogi;
3014 #define glFogi glad_glFogi
3015 GLAD_API_CALL PFNGLFOGIVPROC glad_glFogiv;
3016 #define glFogiv glad_glFogiv
3017 GLAD_API_CALL PFNGLFRAMEBUFFERRENDERBUFFERPROC glad_glFramebufferRenderbuffer;
3018 #define glFramebufferRenderbuffer glad_glFramebufferRenderbuffer
3019 GLAD_API_CALL PFNGLFRAMEBUFFERTEXTUREPROC glad_glFramebufferTexture;
3020 #define glFramebufferTexture glad_glFramebufferTexture
3021 GLAD_API_CALL PFNGLFRAMEBUFFERTEXTURE1DPROC glad_glFramebufferTexture1D;
3022 #define glFramebufferTexture1D glad_glFramebufferTexture1D
3023 GLAD_API_CALL PFNGLFRAMEBUFFERTEXTURE2DPROC glad_glFramebufferTexture2D;
3024 #define glFramebufferTexture2D glad_glFramebufferTexture2D
3025 GLAD_API_CALL PFNGLFRAMEBUFFERTEXTURE3DPROC glad_glFramebufferTexture3D;
3026 #define glFramebufferTexture3D glad_glFramebufferTexture3D
3027 GLAD_API_CALL PFNGLFRAMEBUFFERTEXTURELAYERPROC glad_glFramebufferTextureLayer;
3028 #define glFramebufferTextureLayer glad_glFramebufferTextureLayer
3029 GLAD_API_CALL PFNGLFRONTFACEPROC glad_glFrontFace;
3030 #define glFrontFace glad_glFrontFace
3031 GLAD_API_CALL PFNGLFRUSTUMPROC glad_glFrustum;
3032 #define glFrustum glad_glFrustum
3033 GLAD_API_CALL PFNGLGENBUFFERSPROC glad_glGenBuffers;
3034 #define glGenBuffers glad_glGenBuffers
3035 GLAD_API_CALL PFNGLGENFRAMEBUFFERSPROC glad_glGenFramebuffers;
3036 #define glGenFramebuffers glad_glGenFramebuffers
3037 GLAD_API_CALL PFNGLGENLISTSPROC glad_glGenLists;
3038 #define glGenLists glad_glGenLists
3039 GLAD_API_CALL PFNGLGENQUERIESPROC glad_glGenQueries;
3040 #define glGenQueries glad_glGenQueries
3041 GLAD_API_CALL PFNGLGENRENDERBUFFERSPROC glad_glGenRenderbuffers;
3042 #define glGenRenderbuffers glad_glGenRenderbuffers
3043 GLAD_API_CALL PFNGLGENSAMPLERSPROC glad_glGenSamplers;
```

```
3044 #define glGenSamplers glad_glGenSamplers
3045 GLAD_API_CALL PFNGLGENTEXTURESPROC glad_glGenTextures;
3046 #define glGenTextures glad_glGenTextures
3047 GLAD_API_CALL PFNGLGENVERTEXARRAYSPROC glad_glGenVertexArrays;
3048 #define glGenVertexArrays glad_glGenVertexArrays
3049 GLAD_API_CALL PFNGLGENERATEMIPMAPPROC glad_glGenerateMipmap;
3050 #define glGenerateMipmap glad_glGenerateMipmap
3051 GLAD_API_CALL PFNGLGETACTIVEATTRIBPROC glad_glGetActiveAttrib;
3052 #define glGetActiveAttrib glad_glGetActiveAttrib
3053 GLAD_API_CALL PFNGLGETACTIVEUNIFORMPROC glad_glGetActiveUniform;
3054 #define glGetActiveUniform glad_glGetActiveUniform
3055 GLAD_API_CALL PFNGLGETACTIVEUNIFORMBLOCKNAMEPROC glad_glGetActiveUniformBlockName;
3056 #define glGetActiveUniformBlockName glad_glGetActiveUniformBlockName
3057 GLAD_API_CALL PFNGLGETACTIVEUNIFORMBLOCKIVPROC glad_glGetActiveUniformBlockiv;
3058 #define glGetActiveUniformBlockiv glad_glGetActiveUniformBlockiv
3059 GLAD_API_CALL PFNGLGETACTIVEUNIFORMNAMEPROC glad_glGetActiveUniformName;
3060 #define glGetActiveUniformName glad_glGetActiveUniformName
3061 GLAD_API_CALL PFNGLGETACTIVEUNIFORMSIVPROC glad_glGetActiveUniformsiv;
3062 #define glGetActiveUniformsiv glad_glGetActiveUniformsiv
3063 GLAD_API_CALL PFNGLGETATTACHEDSHADERSPROC glad_glGetAttachedShaders;
3064 #define glGetAttachedShaders glad_glGetAttachedShaders
3065 GLAD_API_CALL PFNGLGETATTRIBLOCATIONPROC glad_glGetAttribLocation;
3066 #define glGetAttribLocation glad_glGetAttribLocation
3067 GLAD_API_CALL PFNGLGETBOOLEANI_VPROC glad_glGetBooleani_v;
3068 #define glGetBooleani_v glad_glGetBooleani_v
3069 GLAD_API_CALL PFNGLGETBOOLEANVPROC glad_glGetBooleanv;
3070 #define glGetBooleanv glad_glGetBooleanv
3071 GLAD_API_CALL PFNGLGETBUFFERPARAMETERI64VPROC glad_glGetBufferParameteri64v;
3072 #define glGetBufferParameteri64v glad_glGetBufferParameteri64v
3073 GLAD_API_CALL PFNGLGETBUFFERPARAMETERIVPROC glad_glGetBufferParameteriv;
3074 #define glGetBufferParameteriv glad_glGetBufferParameteriv
3075 GLAD_API_CALL PFNGLGETBUFFERPOINTERVPROC glad_glGetBufferPointerv;
3076 #define glGetBufferPointerv glad_glGetBufferPointerv
3077 GLAD_API_CALL PFNGLGETBUFFERSUBDATAPROC glad_glGetBufferSubData;
3078 #define glGetBufferSubData glad_glGetBufferSubData
3079 GLAD_API_CALL PFNGLGETCLIPPLANEPROC glad_glGetClipPlane;
3080 #define glGetClipPlane glad_glGetClipPlane
3081 GLAD_API_CALL PFNGLGETCOMPRESSEDTEXIMAGEPROC glad_glGetCompressedTexImage;
3082 #define glGetCompressedTexImage glad_glGetCompressedTexImage
3083 GLAD_API_CALL PFNGLGETDEBUGMESSAGELOGPROC glad_glGetDebugMessageLog;
3084 #define glGetDebugMessageLog glad_glGetDebugMessageLog
3085 GLAD_API_CALL PFNGLGETDOUBLEVPROC glad_glGetDoublev;
3086 #define glGetDoublev glad_glGetDoublev
3087 GLAD_API_CALL PFNGLGETERRORPROC glad_glGetError;
3088 #define glGetError glad_glGetError
3089 GLAD_API_CALL PFNGLGETFLOATVPROC glad_glGetFloatv;
3090 #define glGetFloatv glad_glGetFloatv
3091 GLAD_API_CALL PFNGLGETFRAGDATAINDEXPROC glad_glGetFragDataIndex;
3092 #define glGetFragDataIndex glad_glGetFragDataIndex
3093 GLAD_API_CALL PFNGLGETFRAGDATALOCATIONPROC glad_glGetFragDataLocation;
3094 #define glGetFragDataLocation glad_glGetFragDataLocation
3095 GLAD_API_CALL PFNGLGETFRAMEBUFFERATTACHMENTPARAMETERIVPROC glad_glGetFramebufferAttachmentParameteriv;
3096 #define glGetFramebufferAttachmentParameteriv glad_glGetFramebufferAttachmentParameteriv
3097 GLAD_API_CALL PFNGLGETGRAPHICSRESETSTATUSARBPROC glad_glGetGraphicsResetStatusARB;
3098 #define glGetGraphicsResetStatusARB glad_glGetGraphicsResetStatusARB
3099 GLAD_API_CALL PFNGLGETINTEGER64I_VPROC glad_glGetInteger64i_v;
3100 #define glGetInteger64i_v glad_glGetInteger64i_v
3101 GLAD_API_CALL PFNGLGETINTEGER64VPROC glad_glGetInteger64v;
3102 #define glGetInteger64v glad_glGetInteger64v
3103 GLAD_API_CALL PFNGLGETINTEGERI_VPROC glad_glGetIntegeri_v;
3104 #define glGetIntegeri_v glad_glGetIntegeri_v
3105 GLAD_API_CALL PFNGLGETINTEGERVPROC glad_glGetIntegerv;
3106 #define glGetIntegerv glad_glGetIntegerv
3107 GLAD_API_CALL PFNGLGETLIGHTFVPROC glad_glGetLightfv;
3108 #define glGetLightfv glad_glGetLightfv
3109 GLAD_API_CALL PFNGLGETLIGHTIVPROC glad_glGetLightiv;
3110 #define glGetLightiv glad_glGetLightiv
3111 GLAD_API_CALL PFNGLGETMAPDVPROC glad_glGetMapdv;
3112 #define glGetMapdv glad_glGetMapdv
3113 GLAD_API_CALL PFNGLGETMAPFVPROC glad_glGetMapfv;
3114 #define glGetMapfv glad_glGetMapfv
3115 GLAD_API_CALL PFNGLGETMAPIVPROC glad_glGetMapiv;
3116 #define glGetMapiv glad_glGetMapiv
3117 GLAD_API_CALL PFNGLGETMATERIALFVPROC glad_glGetMaterialfv;
3118 #define glGetMaterialfv glad_glGetMaterialfv
3119 GLAD_API_CALL PFNGLGETMATERIALIVPROC glad_glGetMaterialiv;
3120 #define glGetMaterialiv glad_glGetMaterialiv
3121 GLAD_API_CALL PFNGLGETMULTISAMPLEFVPROC glad_glGetMultisamplefv;
3122 #define glGetMultisamplefv glad_glGetMultisamplefv
3123 GLAD_API_CALL PFNGLGETOBJECTLABELPROC glad_glGetObjectLabel;
3124 #define glGetObjectLabel glad_glGetObjectLabel
3125 GLAD_API_CALL PFNGLGETOBJECTPTRLABELPROC glad_glGetObjectPtrLabel;
3126 #define glGetObjectPtrLabel glad_glGetObjectPtrLabel
3127 GLAD_API_CALL PFNGLGETPIXELMAPFVPROC glad_glGetPixelMapfv;
3128 #define glGetPixelMapfv glad_glGetPixelMapfv
3129 GLAD_API_CALL PFNGLGETPIXELMAPUIVPROC glad_glGetPixelMapuiv;
3130 #define glGetPixelMapuiv glad_glGetPixelMapuiv
```

```
3131 GLAD_API_CALL PFNGLGETPIXELMAPUSVPROC glad_glGetPixelMapusv;
3132 #define glGetPixelMapusv glad_glGetPixelMapusv
3133 GLAD_API_CALL PFNGLGETPOINTERVPROC glad_glGetPointerv;
3134 #define glGetPointerv glad_glGetPointerv
3135 GLAD_API_CALL PFNGLGETPOLYGONSTIPPLEPROC glad_glGetPolygonStipple;
3136 #define glGetPolygonStipple glad_glGetPolygonStipple
3137 GLAD_API_CALL PFNGLGETPROGRAMINFOLOGPROC glad_glGetProgramInfoLog;
3138 #define glGetProgramInfoLog glad_glGetProgramInfoLog
3139 GLAD_API_CALL PFNGLGETPROGRAMIVPROC glad_glGetProgramiv;
3140 #define glGetProgramiv glad_glGetProgramiv
3141 GLAD_API_CALL PFNGLGETQUERYOBJECTI64VPROC glad_glGetQueryObjecti64v;
3142 #define glGetQueryObjecti64v glad_glGetQueryObjecti64v
3143 GLAD_API_CALL PFNGLGETQUERYOBJECTIVPROC glad_glGetQueryObjectiv;
3144 #define glGetQueryObjectiv glad_glGetQueryObjectiv
3145 GLAD_API_CALL PFNGLGETQUERYOBJECTUI64VPROC glad_glGetQueryObjectui64v;
3146 #define glGetQueryObjectui64v glad_glGetQueryObjectui64v
3147 GLAD_API_CALL PFNGLGETQUERYOBJECTUIVPROC glad_glGetQueryObjectuiv;
3148 #define glGetQueryObjectuiv glad_glGetQueryObjectuiv
3149 GLAD_API_CALL PFNGLGETQUERYIVPROC glad_glGetQueryiv;
3150 #define glGetQueryiv glad_glGetQueryiv
3151 GLAD_API_CALL PFNGLGETRENDERBUFFERPARAMETERIVPROC glad_glGetRenderbufferParameteriv;
3152 #define glGetRenderbufferParameteriv glad_glGetRenderbufferParameteriv
3153 GLAD_API_CALL PFNGLGETSAMPLERPARAMETERIIVPROC glad_glGetSamplerParameterIiv;
3154 #define glGetSamplerParameterIiv glad_glGetSamplerParameterIiv
3155 GLAD_API_CALL PFNGLGETSAMPLERPARAMETERIUIVPROC glad_glGetSamplerParameterIuiv;
3156 #define glGetSamplerParameterIuiv glad_glGetSamplerParameterIuiv
3157 GLAD_API_CALL PFNGLGETSAMPLERPARAMETERFVPROC glad_glGetSamplerParameterfv;
3158 #define glGetSamplerParameterfv glad_glGetSamplerParameterfv
3159 GLAD_API_CALL PFNGLGETSAMPLERPARAMETERIVPROC glad_glGetSamplerParameteriv;
3160 #define glGetSamplerParameteriv glad_glGetSamplerParameteriv
3161 GLAD_API_CALL PFNGLGETSHADERINFOLOGPROC glad_glGetShaderInfoLog;
3162 #define glGetShaderInfoLog glad_glGetShaderInfoLog
3163 GLAD_API_CALL PFNGLGETSHADERSOURCEPROC glad_glGetShaderSource;
3164 #define glGetShaderSource glad_glGetShaderSource
3165 GLAD_API_CALL PFNGLGETSHADERIVPROC glad_glGetShaderiv;
3166 #define glGetShaderiv glad_glGetShaderiv
3167 GLAD_API_CALL PFNGLGETSTRINGPROC glad_glGetString;
3168 #define glGetString glad_glGetString
3169 GLAD_API_CALL PFNGLGETSTRINGIPROC glad_glGetStringi;
3170 #define glGetStringi glad_glGetStringi
3171 GLAD_API_CALL PFNGLGETSYNCIVPROC glad_glGetSynciv;
3172 #define glGetSynciv glad_glGetSynciv
3173 GLAD_API_CALL PFNGLGETTEXENVFVPROC glad_glGetTexEnvfv;
3174 #define glGetTexEnvfv glad_glGetTexEnvfv
3175 GLAD_API_CALL PFNGLGETTEXENVIVPROC glad_glGetTexEnviv;
3176 #define glGetTexEnviv glad_glGetTexEnviv
3177 GLAD_API_CALL PFNGLGETTEXGENDVPROC glad_glGetTexGendv;
3178 #define glGetTexGendv glad_glGetTexGendv
3179 GLAD_API_CALL PFNGLGETTEXGENFVPROC glad_glGetTexGenfv;
3180 #define glGetTexGenfv glad_glGetTexGenfv
3181 GLAD_API_CALL PFNGLGETTEXGENIVPROC glad_glGetTexGeniv;
3182 #define glGetTexGeniv glad_glGetTexGeniv
3183 GLAD_API_CALL PFNGLGETTEXIMAGEPROC glad_glGetTexImage;
3184 #define glGetTexImage glad_glGetTexImage
3185 GLAD_API_CALL PFNGLGETTEXLEVELPARAMETERFVPROC glad_glGetTexLevelParameterfv;
3186 #define glGetTexLevelParameterfv glad_glGetTexLevelParameterfv
3187 GLAD_API_CALL PFNGLGETTEXLEVELPARAMETERIVPROC glad_glGetTexLevelParameteriv;
3188 #define glGetTexLevelParameteriv glad_glGetTexLevelParameteriv
3189 GLAD_API_CALL PFNGLGETTEXPARAMETERIIVPROC glad_glGetTexParameterIiv;
3190 #define glGetTexParameterIiv glad_glGetTexParameterIiv
3191 GLAD_API_CALL PFNGLGETTEXPARAMETERIUIVPROC glad_glGetTexParameterIuiv;
3192 #define glGetTexParameterIuiv glad_glGetTexParameterIuiv
3193 GLAD_API_CALL PFNGLGETTEXPARAMETERFVPROC glad_glGetTexParameterfv;
3194 #define glGetTexParameterfv glad_glGetTexParameterfv
3195 GLAD_API_CALL PFNGLGETTEXPARAMETERIVPROC glad_glGetTexParameteriv;
3196 #define glGetTexParameteriv glad_glGetTexParameteriv
3197 GLAD_API_CALL PFNGLGETTRANSFORMFEEDBACKVARYINGPROC glad_glGetTransformFeedbackVarying;
3198 #define glGetTransformFeedbackVarying glad_glGetTransformFeedbackVarying
3199 GLAD_API_CALL PFNGLGETUNIFORMBLOCKINDEXPROC glad_glGetUniformBlockIndex;
3200 #define glGetUniformBlockIndex glad_glGetUniformBlockIndex
3201 GLAD_API_CALL PFNGLGETUNIFORMINDICESPROC glad_glGetUniformIndices;
3202 #define glGetUniformIndices glad_glGetUniformIndices
3203 GLAD_API_CALL PFNGLGETUNIFORMLOCATIONPROC glad_glGetUniformLocation;
3204 #define glGetUniformLocation glad_glGetUniformLocation
3205 GLAD_API_CALL PFNGLGETUNIFORMFVPROC glad_glGetUniformfv;
3206 #define glGetUniformfv glad_glGetUniformfv
3207 GLAD_API_CALL PFNGLGETUNIFORMIVPROC glad_glGetUniformiv;
3208 #define glGetUniformiv glad_glGetUniformiv
3209 GLAD_API_CALL PFNGLGETUNIFORMUIVPROC glad_glGetUniformuiv;
3210 #define glGetUniformuiv glad_glGetUniformuiv
3211 GLAD_API_CALL PFNGLGETVERTEXATTRIBIIVPROC glad_glGetVertexAttribIiv;
3212 #define glGetVertexAttribIiv glad_glGetVertexAttribIiv
3213 GLAD_API_CALL PFNGLGETVERTEXATTRIBIUIVPROC glad_glGetVertexAttribIuiv;
3214 #define glGetVertexAttribIuiv glad_glGetVertexAttribIuiv
3215 GLAD_API_CALL PFNGLGETVERTEXATTRIBPOINTERVPROC glad_glGetVertexAttribPointerv;
3216 #define glGetVertexAttribPointerv glad_glGetVertexAttribPointerv
3217 GLAD_API_CALL PFNGLGETVERTEXATTRIBDVPROC glad_glGetVertexAttribdv;
```

```
3218 #define glGetVertexAttribdv glad_glGetVertexAttribdv
3219 GLAD_API_CALL PFNGLGETVERTEXATTRIBFVPROC glad_glGetVertexAttribfv;
3220 #define glGetVertexAttribfv glad_glGetVertexAttribfv
3221 GLAD_API_CALL PFNGLGETVERTEXATTRIBIVPROC glad_glGetVertexAttribiv;
3222 #define glGetVertexAttribiv glad_glGetVertexAttribiv
3223 GLAD_API_CALL PFNGLGETNCOLORTABLEARBPROC glad_glGetnColorTableARB;
3224 #define glGetnColorTableARB glad_glGetnColorTableARB
3225 GLAD_API_CALL PFNGLGETNCOMPRESSEDTEXIMAGEARBPROC glad_glGetnCompressedTexImageARB;
3226 #define glGetnCompressedTexImageARB glad_glGetnCompressedTexImageARB
3227 GLAD_API_CALL PFNGLGETNCONVOLUTIONFILTERARBPROC glad_glGetnConvolutionFilterARB;
3228 #define glGetnConvolutionFilterARB glad_glGetnConvolutionFilterARB
3229 GLAD_API_CALL PFNGLGETNHISTOGRAMARBPROC glad_glGetnHistogramARB;
3230 #define glGetnHistogramARB glad_glGetnHistogramARB
3231 GLAD_API_CALL PFNGLGETNMAPDVARBPROC glad_glGetnMapdvARB;
3232 #define glGetnMapdvARB glad_glGetnMapdvARB
3233 GLAD_API_CALL PFNGLGETNMAPFVARBPROC glad_glGetnMapfvARB;
3234 #define glGetnMapfvARB glad_glGetnMapfvARB
3235 GLAD_API_CALL PFNGLGETNMAPIVARBPROC glad_glGetnMapivARB;
3236 #define glGetnMapivARB glad_glGetnMapivARB
3237 GLAD_API_CALL PFNGLGETNMINMAXARBPROC glad_glGetnMinmaxARB;
3238 #define glGetnMinmaxARB glad_glGetnMinmaxARB
3239 GLAD_API_CALL PFNGLGETNPIXELMAPFVARBPROC glad_glGetnPixelMapfvARB;
3240 #define glGetnPixelMapfvARB glad_glGetnPixelMapfvARB
3241 GLAD_API_CALL PFNGLGETNPIXELMAPUIVARBPROC glad_glGetnPixelMapuivARB;
3242 #define glGetnPixelMapuivARB glad_glGetnPixelMapuivARB
3243 GLAD_API_CALL PFNGLGETNPIXELMAPUSVARBPROC glad_glGetnPixelMapusvARB;
3244 #define glGetnPixelMapusvARB glad_glGetnPixelMapusvARB
3245 GLAD_API_CALL PFNGLGETNPOLYGONSTIPPLEARBPROC glad_glGetnPolygonStippleARB;
3246 #define glGetnPolygonStippleARB glad_glGetnPolygonStippleARB
3247 GLAD_API_CALL PFNGLGETNSEPARABLEFILTERARBPROC glad_glGetnSeparableFilterARB;
3248 #define glGetnSeparableFilterARB glad_glGetnSeparableFilterARB
3249 GLAD_API_CALL PFNGLGETNTEXIMAGEARBPROC glad_glGetnTexImageARB;
3250 #define glGetnTexImageARB glad_glGetnTexImageARB
3251 GLAD_API_CALL PFNGLGETNUNIFORMDVARBPROC glad_glGetnUniformdvARB;
3252 #define glGetnUniformdvARB glad_glGetnUniformdvARB
3253 GLAD_API_CALL PFNGLGETNUNIFORMFVARBPROC glad_glGetnUniformfvARB;
3254 #define glGetnUniformfvARB glad_glGetnUniformfvARB
3255 GLAD_API_CALL PFNGLGETNUNIFORMIVARBPROC glad_glGetnUniformivARB;
3256 #define glGetnUniformivARB glad_glGetnUniformivARB
3257 GLAD_API_CALL PFNGLGETNUNIFORMUIVARBPROC glad_glGetnUniformuivARB;
3258 #define glGetnUniformuivARB glad_glGetnUniformuivARB
3259 GLAD_API_CALL PFNGLHINTPROC glad_glHint;
3260 #define glHint glad_glHint
3261 GLAD_API_CALL PFNGLINDEXMASKPROC glad_glIndexMask;
3262 #define glIndexMask glad_glIndexMask
3263 GLAD_API_CALL PFNGLINDEXPOINTERPROC glad_glIndexPointer;
3264 #define glIndexPointer glad_glIndexPointer
3265 GLAD_API_CALL PFNGLINDEXDPROC glad_glIndexd;
3266 #define glIndexd glad_glIndexd
3267 GLAD_API_CALL PFNGLINDEXDVPROC glad_glIndexdv;
3268 #define glIndexdv glad_glIndexdv
3269 GLAD_API_CALL PFNGLINDEXFPROC glad_glIndexf;
3270 #define glIndexf glad_glIndexf
3271 GLAD_API_CALL PFNGLINDEXFVPROC glad_glIndexfv;
3272 #define glIndexfv glad_glIndexfv
3273 GLAD_API_CALL PFNGLINDEXIPROC glad_glIndexi;
3274 #define glIndexi glad_glIndexi
3275 GLAD_API_CALL PFNGLINDEXIVPROC glad_glIndexiv;
3276 #define glIndexiv glad_glIndexiv
3277 GLAD_API_CALL PFNGLINDEXSPROC glad_glIndexs;
3278 #define glIndexs glad_glIndexs
3279 GLAD_API_CALL PFNGLINDEXSVPROC glad_glIndexsv;
3280 #define glIndexsv glad_glIndexsv
3281 GLAD_API_CALL PFNGLINDEXUBPROC glad_glIndexub;
3282 #define glIndexub glad_glIndexub
3283 GLAD_API_CALL PFNGLINDEXUBVPROC glad_glIndexubv;
3284 #define glIndexubv glad_glIndexubv
3285 GLAD_API_CALL PFNGLINITNAMESPROC glad_glInitNames;
3286 #define glInitNames glad_glInitNames
3287 GLAD_API_CALL PFNGLINTERLEAVEDARRAYSPROC glad_glInterleavedArrays;
3288 #define glInterleavedArrays glad_glInterleavedArrays
3289 GLAD_API_CALL PFNGLISBUFFERPROC glad_glIsBuffer;
3290 #define glIsBuffer glad_glIsBuffer
3291 GLAD_API_CALL PFNGLISENABLEDPROC glad_glIsEnabled;
3292 #define glIsEnabled glad_glIsEnabled
3293 GLAD_API_CALL PFNGLISENABLEDIPROC glad_glIsEnabledi;
3294 #define glIsEnabledi glad_glIsEnabledi
3295 GLAD_API_CALL PFNGLISFRAMEBUFFERPROC glad_glIsFramebuffer;
3296 #define glIsFramebuffer glad_glIsFramebuffer
3297 GLAD_API_CALL PFNGLISLISTPROC glad_glIsList;
3298 #define glIsList glad_glIsList
3299 GLAD_API_CALL PFNGLISPROGRAMPROC glad_glIsProgram;
3300 #define glIsProgram glad_glIsProgram
3301 GLAD_API_CALL PFNGLISQUERYPROC glad_glIsQuery;
3302 #define glIsQuery glad_glIsQuery
3303 GLAD_API_CALL PFNGLISRENDERBUFFERPROC glad_glIsRenderbuffer;
3304 #define glIsRenderbuffer glad_glIsRenderbuffer
```

```
3305 GLAD_API_CALL PFNGLISSAMPLERPROC glad_glIsSampler;
3306 #define glIsSampler glad_glIsSampler
3307 GLAD_API_CALL PFNGLISSHADERPROC glad_glIsShader;
3308 #define glIsShader glad_glIsShader
3309 GLAD_API_CALL PFNGLISSYNCPROC glad_glIsSync;
3310 #define glIsSync glad_glIsSync
3311 GLAD_API_CALL PFNGLISTEXTUREPROC glad_glIsTexture;
3312 #define glIsTexture glad_glIsTexture
3313 GLAD_API_CALL PFNGLISVERTEXARRAYPROC glad_glIsVertexArray;
3314 #define glIsVertexArray glad_glIsVertexArray
3315 GLAD_API_CALL PFNGLLIGHTMODELFPROC glad_glLightModelf;
3316 #define glLightModelf glad_glLightModelf
3317 GLAD_API_CALL PFNGLLIGHTMODELFVPROC glad_glLightModelfv;
3318 #define glLightModelfv glad_glLightModelfv
3319 GLAD_API_CALL PFNGLLIGHTMODELIPROC glad_glLightModeli;
3320 #define glLightModeli glad_glLightModeli
3321 GLAD_API_CALL PFNGLLIGHTMODELIVPROC glad_glLightModeliv;
3322 #define glLightModeliv glad_glLightModeliv
3323 GLAD_API_CALL PFNGLLIGHTFPROC glad_glLightf;
3324 #define glLightf glad_glLightf
3325 GLAD_API_CALL PFNGLLIGHTFVPROC glad_glLightfv;
3326 #define glLightfv glad_glLightfv
3327 GLAD_API_CALL PFNGLLIGHTIPROC glad_glLighti;
3328 #define glLighti glad_glLighti
3329 GLAD_API_CALL PFNGLLIGHTIVPROC glad_glLightiv;
3330 #define glLightiv glad_glLightiv
3331 GLAD_API_CALL PFNGLLINESTIPPLEPROC glad_glLineStipple;
3332 #define glLineStipple glad_glLineStipple
3333 GLAD_API_CALL PFNGLLINEWIDTHPROC glad_glLineWidth;
3334 #define glLineWidth glad_glLineWidth
3335 GLAD_API_CALL PFNGLLINKPROGRAMPROC glad_glLinkProgram;
3336 #define glLinkProgram glad_glLinkProgram
3337 GLAD_API_CALL PFNGLLISTBASEPROC glad_glListBase;
3338 #define glListBase glad_glListBase
3339 GLAD_API_CALL PFNGLLOADIDENTITYPROC glad_glLoadIdentity;
3340 #define glLoadIdentity glad_glLoadIdentity
3341 GLAD_API_CALL PFNGLLOADMATRIXDPROC glad_glLoadMatrixd;
3342 #define glLoadMatrixd glad_glLoadMatrixd
3343 GLAD_API_CALL PFNGLLOADMATRIXFPROC glad_glLoadMatrixf;
3344 #define glLoadMatrixf glad_glLoadMatrixf
3345 GLAD_API_CALL PFNGLLOADNAMEPROC glad_glLoadName;
3346 #define glLoadName glad_glLoadName
3347 GLAD_API_CALL PFNGLLOADTRANSPOSEMATRIXDPROC glad_glLoadTransposeMatrixd;
3348 #define glLoadTransposeMatrixd glad_glLoadTransposeMatrixd
3349 GLAD_API_CALL PFNGLLOADTRANSPOSEMATRIXFPROC glad_glLoadTransposeMatrixf;
3350 #define glLoadTransposeMatrixf glad_glLoadTransposeMatrixf
3351 GLAD_API_CALL PFNGLLOGICOPPROC glad_glLogicOp;
3352 #define glLogicOp glad_glLogicOp
3353 GLAD_API_CALL PFNGLMAP1DPROC glad_glMap1d;
3354 #define glMap1d glad_glMap1d
3355 GLAD_API_CALL PFNGLMAP1FPROC glad_glMap1f;
3356 #define glMap1f glad_glMap1f
3357 GLAD_API_CALL PFNGLMAP2DPROC glad_glMap2d;
3358 #define glMap2d glad_glMap2d
3359 GLAD_API_CALL PFNGLMAP2FPROC glad_glMap2f;
3360 #define glMap2f glad_glMap2f
3361 GLAD_API_CALL PFNGLMAPBUFFERPROC glad_glMapBuffer;
3362 #define glMapBuffer glad_glMapBuffer
3363 GLAD_API_CALL PFNGLMAPBUFFERRANGEPROC glad_glMapBufferRange;
3364 #define glMapBufferRange glad_glMapBufferRange
3365 GLAD_API_CALL PFNGLMAPGRID1DPROC glad_glMapGrid1d;
3366 #define glMapGrid1d glad_glMapGrid1d
3367 GLAD_API_CALL PFNGLMAPGRID1FPROC glad_glMapGrid1f;
3368 #define glMapGrid1f glad_glMapGrid1f
3369 GLAD_API_CALL PFNGLMAPGRID2DPROC glad_glMapGrid2d;
3370 #define glMapGrid2d glad_glMapGrid2d
3371 GLAD_API_CALL PFNGLMAPGRID2FPROC glad_glMapGrid2f;
3372 #define glMapGrid2f glad_glMapGrid2f
3373 GLAD_API_CALL PFNGLMATERIALFPROC glad_glMaterialf;
3374 #define glMaterialf glad_glMaterialf
3375 GLAD_API_CALL PFNGLMATERIALFVPROC glad_glMaterialfv;
3376 #define glMaterialfv glad_glMaterialfv
3377 GLAD_API_CALL PFNGLMATERIALIPROC glad_glMateriali;
3378 #define glMateriali glad_glMateriali
3379 GLAD_API_CALL PFNGLMATERIALIVPROC glad_glMaterialiv;
3380 #define glMaterialiv glad_glMaterialiv
3381 GLAD_API_CALL PFNGLMATRIXMODEPROC glad_glMatrixMode;
3382 #define glMatrixMode glad_glMatrixMode
3383 GLAD_API_CALL PFNGLMULTMATRIXDPROC glad_glMultMatrixd;
3384 #define glMultMatrixd glad_glMultMatrixd
3385 GLAD_API_CALL PFNGLMULTMATRIXFPROC glad_glMultMatrixf;
3386 #define glMultMatrixf glad_glMultMatrixf
3387 GLAD_API_CALL PFNGLMULTTRANSPOSEMATRIXDPROC glad_glMultTransposeMatrixd;
3388 #define glMultTransposeMatrixd glad_glMultTransposeMatrixd
3389 GLAD_API_CALL PFNGLMULTTRANSPOSEMATRIXFPROC glad_glMultTransposeMatrixf;
3390 #define glMultTransposeMatrixf glad_glMultTransposeMatrixf
3391 GLAD_API_CALL PFNGLMULTIDRAWARRAYSPROC glad_glMultiDrawArrays;
```

```
3392 #define glMultiDrawArrays glad_glMultiDrawArrays
3393 GLAD_API_CALL PFNGLMULTIDRAWELEMENTSPROC glad_glMultiDrawElements;
3394 #define glMultiDrawElements glad_glMultiDrawElements
3395 GLAD_API_CALL PFNGLMULTIDRAWELEMENTSBASEVERTEXPROC glad_glMultiDrawElementsBaseVertex;
3396 #define glMultiDrawElementsBaseVertex glad_glMultiDrawElementsBaseVertex
3397 GLAD_API_CALL PFNGLMULTITEXCOORD1DPROC glad_glMultiTexCoord1d;
3398 #define glMultiTexCoord1d glad_glMultiTexCoord1d
3399 GLAD_API_CALL PFNGLMULTITEXCOORD1DVPROC glad_glMultiTexCoord1dv;
3400 #define glMultiTexCoord1dv glad_glMultiTexCoord1dv
3401 GLAD_API_CALL PFNGLMULTITEXCOORD1FPROC glad_glMultiTexCoord1f;
3402 #define glMultiTexCoord1f glad_glMultiTexCoord1f
3403 GLAD_API_CALL PFNGLMULTITEXCOORD1FVPROC glad_glMultiTexCoord1fv;
3404 #define glMultiTexCoord1fv glad_glMultiTexCoord1fv
3405 GLAD_API_CALL PFNGLMULTITEXCOORD1IPROC glad_glMultiTexCoord1i;
3406 #define glMultiTexCoord1i glad_glMultiTexCoord1i
3407 GLAD_API_CALL PFNGLMULTITEXCOORD1IVPROC glad_glMultiTexCoord1iv;
3408 #define glMultiTexCoord1iv glad_glMultiTexCoord1iv
3409 GLAD_API_CALL PFNGLMULTITEXCOORD1SPROC glad_glMultiTexCoord1s;
3410 #define glMultiTexCoord1s glad_glMultiTexCoord1s
3411 GLAD_API_CALL PFNGLMULTITEXCOORD1SVPROC glad_glMultiTexCoord1sv;
3412 #define glMultiTexCoord1sv glad_glMultiTexCoord1sv
3413 GLAD_API_CALL PFNGLMULTITEXCOORD2DPROC glad_glMultiTexCoord2d;
3414 #define glMultiTexCoord2d glad_glMultiTexCoord2d
3415 GLAD_API_CALL PFNGLMULTITEXCOORD2DVPROC glad_glMultiTexCoord2dv;
3416 #define glMultiTexCoord2dv glad_glMultiTexCoord2dv
3417 GLAD_API_CALL PFNGLMULTITEXCOORD2FPROC glad_glMultiTexCoord2f;
3418 #define glMultiTexCoord2f glad_glMultiTexCoord2f
3419 GLAD_API_CALL PFNGLMULTITEXCOORD2FVPROC glad_glMultiTexCoord2fv;
3420 #define glMultiTexCoord2fv glad_glMultiTexCoord2fv
3421 GLAD_API_CALL PFNGLMULTITEXCOORD2IPROC glad_glMultiTexCoord2i;
3422 #define glMultiTexCoord2i glad_glMultiTexCoord2i
3423 GLAD_API_CALL PFNGLMULTITEXCOORD2IVPROC glad_glMultiTexCoord2iv;
3424 #define glMultiTexCoord2iv glad_glMultiTexCoord2iv
3425 GLAD_API_CALL PFNGLMULTITEXCOORD2SPROC glad_glMultiTexCoord2s;
3426 #define glMultiTexCoord2s glad_glMultiTexCoord2s
3427 GLAD_API_CALL PFNGLMULTITEXCOORD2SVPROC glad_glMultiTexCoord2sv;
3428 #define glMultiTexCoord2sv glad_glMultiTexCoord2sv
3429 GLAD_API_CALL PFNGLMULTITEXCOORD3DPROC glad_glMultiTexCoord3d;
3430 #define glMultiTexCoord3d glad_glMultiTexCoord3d
3431 GLAD_API_CALL PFNGLMULTITEXCOORD3DVPROC glad_glMultiTexCoord3dv;
3432 #define glMultiTexCoord3dv glad_glMultiTexCoord3dv
3433 GLAD_API_CALL PFNGLMULTITEXCOORD3FPROC glad_glMultiTexCoord3f;
3434 #define glMultiTexCoord3f glad_glMultiTexCoord3f
3435 GLAD_API_CALL PFNGLMULTITEXCOORD3FVPROC glad_glMultiTexCoord3fv;
3436 #define glMultiTexCoord3fv glad_glMultiTexCoord3fv
3437 GLAD_API_CALL PFNGLMULTITEXCOORD3IPROC glad_glMultiTexCoord3i;
3438 #define glMultiTexCoord3i glad_glMultiTexCoord3i
3439 GLAD_API_CALL PFNGLMULTITEXCOORD3IVPROC glad_glMultiTexCoord3iv;
3440 #define glMultiTexCoord3iv glad_glMultiTexCoord3iv
3441 GLAD_API_CALL PFNGLMULTITEXCOORD3SPROC glad_glMultiTexCoord3s;
3442 #define glMultiTexCoord3s glad_glMultiTexCoord3s
3443 GLAD_API_CALL PFNGLMULTITEXCOORD3SVPROC glad_glMultiTexCoord3sv;
3444 #define glMultiTexCoord3sv glad_glMultiTexCoord3sv
3445 GLAD_API_CALL PFNGLMULTITEXCOORD4DPROC glad_glMultiTexCoord4d;
3446 #define glMultiTexCoord4d glad_glMultiTexCoord4d
3447 GLAD_API_CALL PFNGLMULTITEXCOORD4DVPROC glad_glMultiTexCoord4dv;
3448 #define glMultiTexCoord4dv glad_glMultiTexCoord4dv
3449 GLAD_API_CALL PFNGLMULTITEXCOORD4FPROC glad_glMultiTexCoord4f;
3450 #define glMultiTexCoord4f glad_glMultiTexCoord4f
3451 GLAD_API_CALL PFNGLMULTITEXCOORD4FVPROC glad_glMultiTexCoord4fv;
3452 #define glMultiTexCoord4fv glad_glMultiTexCoord4fv
3453 GLAD_API_CALL PFNGLMULTITEXCOORD4IPROC glad_glMultiTexCoord4i;
3454 #define glMultiTexCoord4i glad_glMultiTexCoord4i
3455 GLAD_API_CALL PFNGLMULTITEXCOORD4IVPROC glad_glMultiTexCoord4iv;
3456 #define glMultiTexCoord4iv glad_glMultiTexCoord4iv
3457 GLAD_API_CALL PFNGLMULTITEXCOORD4SPROC glad_glMultiTexCoord4s;
3458 #define glMultiTexCoord4s glad_glMultiTexCoord4s
3459 GLAD_API_CALL PFNGLMULTITEXCOORD4SVPROC glad_glMultiTexCoord4sv;
3460 #define glMultiTexCoord4sv glad_glMultiTexCoord4sv
3461 GLAD_API_CALL PFNGLMULTITEXCOORDP1UIPROC glad_glMultiTexCoordP1ui;
3462 #define glMultiTexCoordP1ui glad_glMultiTexCoordP1ui
3463 GLAD_API_CALL PFNGLMULTITEXCOORDP1UIVPROC glad_glMultiTexCoordP1uiv;
3464 #define glMultiTexCoordP1uiv glad_glMultiTexCoordP1uiv
3465 GLAD_API_CALL PFNGLMULTITEXCOORDP2UIPROC glad_glMultiTexCoordP2ui;
3466 #define glMultiTexCoordP2ui glad_glMultiTexCoordP2ui
3467 GLAD_API_CALL PFNGLMULTITEXCOORDP2UIVPROC glad_glMultiTexCoordP2uiv;
3468 #define glMultiTexCoordP2uiv glad_glMultiTexCoordP2uiv
3469 GLAD_API_CALL PFNGLMULTITEXCOORDP3UIPROC glad_glMultiTexCoordP3ui;
3470 #define glMultiTexCoordP3ui glad_glMultiTexCoordP3ui
3471 GLAD_API_CALL PFNGLMULTITEXCOORDP3UIVPROC glad_glMultiTexCoordP3uiv;
3472 #define glMultiTexCoordP3uiv glad_glMultiTexCoordP3uiv
3473 GLAD_API_CALL PFNGLMULTITEXCOORDP4UIPROC glad_glMultiTexCoordP4ui;
3474 #define glMultiTexCoordP4ui glad_glMultiTexCoordP4ui
3475 GLAD_API_CALL PFNGLMULTITEXCOORDP4UIVPROC glad_glMultiTexCoordP4uiv;
3476 #define glMultiTexCoordP4uiv glad_glMultiTexCoordP4uiv
3477 GLAD_API_CALL PFNGLNEWLISTPROC glad_glNewList;
3478 #define glNewList glad_glNewList
```

```
3479 GLAD_API_CALL PFNGLNORMAL3BPROC glad_glNormal3b;
3480 #define glNormal3b glad_glNormal3b
3481 GLAD_API_CALL PFNGLNORMAL3BVPROC glad_glNormal3bv;
3482 #define glNormal3bv glad_glNormal3bv
3483 GLAD_API_CALL PFNGLNORMAL3DPROC glad_glNormal3d;
3484 #define glNormal3d glad_glNormal3d
3485 GLAD_API_CALL PFNGLNORMAL3DVPROC glad_glNormal3dv;
3486 #define glNormal3dv glad_glNormal3dv
3487 GLAD_API_CALL PFNGLNORMAL3FPROC glad_glNormal3f;
3488 #define glNormal3f glad_glNormal3f
3489 GLAD_API_CALL PFNGLNORMAL3FVPROC glad_glNormal3fv;
3490 #define glNormal3fv glad_glNormal3fv
3491 GLAD_API_CALL PFNGLNORMAL3IPROC glad_glNormal3i;
3492 #define glNormal3i glad_glNormal3i
3493 GLAD_API_CALL PFNGLNORMAL3IVPROC glad_glNormal3iv;
3494 #define glNormal3iv glad_glNormal3iv
3495 GLAD_API_CALL PFNGLNORMAL3SPROC glad_glNormal3s;
3496 #define glNormal3s glad_glNormal3s
3497 GLAD_API_CALL PFNGLNORMAL3SVPROC glad_glNormal3sv;
3498 #define glNormal3sv glad_glNormal3sv
3499 GLAD_API_CALL PFNGLNORMALP3UIPROC glad_glNormalP3ui;
3500 #define glNormalP3ui glad_glNormalP3ui
3501 GLAD_API_CALL PFNGLNORMALP3UIVPROC glad_glNormalP3uiv;
3502 #define glNormalP3uiv glad_glNormalP3uiv
3503 GLAD_API_CALL PFNGLNORMALPOINTERPROC glad_glNormalPointer;
3504 #define glNormalPointer glad_glNormalPointer
3505 GLAD_API_CALL PFNGLOBJECTLABELPROC glad_glObjectLabel;
3506 #define glObjectLabel glad_glObjectLabel
3507 GLAD_API_CALL PFNGLOBJECTPTRLABELPROC glad_glObjectPtrLabel;
3508 #define glObjectPtrLabel glad_glObjectPtrLabel
3509 GLAD_API_CALL PFNGLORTHOPROC glad_glOrtho;
3510 #define glOrtho glad_glOrtho
3511 GLAD_API_CALL PFNGLPASSTHROUGHPROC glad_glPassThrough;
3512 #define glPassThrough glad_glPassThrough
3513 GLAD_API_CALL PFNGLPIXELMAPFVPROC glad_glPixelMapfv;
3514 #define glPixelMapfv glad_glPixelMapfv
3515 GLAD_API_CALL PFNGLPIXELMAPUIVPROC glad_glPixelMapuiv;
3516 #define glPixelMapuiv glad_glPixelMapuiv
3517 GLAD_API_CALL PFNGLPIXELMAPUSVPROC glad_glPixelMapusv;
3518 #define glPixelMapusv glad_glPixelMapusv
3519 GLAD_API_CALL PFNGLPIXELSTOREFPROC glad_glPixelStoref;
3520 #define glPixelStoref glad_glPixelStoref
3521 GLAD_API_CALL PFNGLPIXELSTOREIPROC glad_glPixelStorei;
3522 #define glPixelStorei glad_glPixelStorei
3523 GLAD_API_CALL PFNGLPIXELTRANSFERFPROC glad_glPixelTransferf;
3524 #define glPixelTransferf glad_glPixelTransferf
3525 GLAD_API_CALL PFNGLPIXELTRANSFERIPROC glad_glPixelTransferi;
3526 #define glPixelTransferi glad_glPixelTransferi
3527 GLAD_API_CALL PFNGLPIXELZOOMPROC glad_glPixelZoom;
3528 #define glPixelZoom glad_glPixelZoom
3529 GLAD_API_CALL PFNGLPOINTPARAMETERFPROC glad_glPointParameterf;
3530 #define glPointParameterf glad_glPointParameterf
3531 GLAD_API_CALL PFNGLPOINTPARAMETERFVPROC glad_glPointParameterfv;
3532 #define glPointParameterfv glad_glPointParameterfv
3533 GLAD_API_CALL PFNGLPOINTPARAMETERIPROC glad_glPointParameteri;
3534 #define glPointParameteri glad_glPointParameteri
3535 GLAD_API_CALL PFNGLPOINTPARAMETERIVPROC glad_glPointParameteriv;
3536 #define glPointParameteriv glad_glPointParameteriv
3537 GLAD_API_CALL PFNGLPOINTSIZEPROC glad_glPointSize;
3538 #define glPointSize glad_glPointSize
3539 GLAD_API_CALL PFNGLPOLYGONMODEPROC glad_glPolygonMode;
3540 #define glPolygonMode glad_glPolygonMode
3541 GLAD_API_CALL PFNGLPOLYGONOFFSETPROC glad_glPolygonOffset;
3542 #define glPolygonOffset glad_glPolygonOffset
3543 GLAD_API_CALL PFNGLPOLYGONSTIPPLEPROC glad_glPolygonStipple;
3544 #define glPolygonStipple glad_glPolygonStipple
3545 GLAD_API_CALL PFNGLPOPATTRIBPROC glad_glPopAttrib;
3546 #define glPopAttrib glad_glPopAttrib
3547 GLAD_API_CALL PFNGLPOPCLIENTATTRIBPROC glad_glPopClientAttrib;
3548 #define glPopClientAttrib glad_glPopClientAttrib
3549 GLAD_API_CALL PFNGLPOPDEBUGGROUPPROC glad_glPopDebugGroup;
3550 #define glPopDebugGroup glad_glPopDebugGroup
3551 GLAD_API_CALL PFNGLPOPMATRIXPROC glad_glPopMatrix;
3552 #define glPopMatrix glad_glPopMatrix
3553 GLAD_API_CALL PFNGLPOPNAMEPROC glad_glPopName;
3554 #define glPopName glad_glPopName
3555 GLAD_API_CALL PFNGLPRIMITIVERESTARTINDEXPROC glad_glPrimitiveRestartIndex;
3556 #define glPrimitiveRestartIndex glad_glPrimitiveRestartIndex
3557 GLAD_API_CALL PFNGLPRIORITIZETEXTURESPROC glad_glPrioritizeTextures;
3558 #define glPrioritizeTextures glad_glPrioritizeTextures
3559 GLAD_API_CALL PFNGLPROVOKINGVERTEXPROC glad_glProvokingVertex;
3560 #define glProvokingVertex glad_glProvokingVertex
3561 GLAD_API_CALL PFNGLPUSHATTRIBPROC glad_glPushAttrib;
3562 #define glPushAttrib glad_glPushAttrib
3563 GLAD_API_CALL PFNGLPUSHCLIENTATTRIBPROC glad_glPushClientAttrib;
3564 #define glPushClientAttrib glad_glPushClientAttrib
3565 GLAD_API_CALL PFNGLPUSHDEBUGGROUPPROC glad_glPushDebugGroup;
```

```
3566 #define glPushDebugGroup glad_glPushDebugGroup
3567 GLAD_API_CALL PFNGLPUSHMATRIXPROC glad_glPushMatrix;
3568 #define glPushMatrix glad_glPushMatrix
3569 GLAD_API_CALL PFNGLPUSHNAMEPROC glad_glPushName;
3570 #define glPushName glad_glPushName
3571 GLAD_API_CALL PFNGLQUERYCOUNTERPROC glad_glQueryCounter;
3572 #define glQueryCounter glad_glQueryCounter
3573 GLAD_API_CALL PFNGLRASTERPOS2DPROC glad_glRasterPos2d;
3574 #define glRasterPos2d glad_glRasterPos2d
3575 GLAD_API_CALL PFNGLRASTERPOS2DVPROC glad_glRasterPos2dv;
3576 #define glRasterPos2dv glad_glRasterPos2dv
3577 GLAD_API_CALL PFNGLRASTERPOS2FPROC glad_glRasterPos2f;
3578 #define glRasterPos2f glad_glRasterPos2f
3579 GLAD_API_CALL PFNGLRASTERPOS2FVPROC glad_glRasterPos2fv;
3580 #define glRasterPos2fv glad_glRasterPos2fv
3581 GLAD_API_CALL PFNGLRASTERPOS2IPROC glad_glRasterPos2i;
3582 #define glRasterPos2i glad_glRasterPos2i
3583 GLAD_API_CALL PFNGLRASTERPOS2IVPROC glad_glRasterPos2iv;
3584 #define glRasterPos2iv glad_glRasterPos2iv
3585 GLAD_API_CALL PFNGLRASTERPOS2SPROC glad_glRasterPos2s;
3586 #define glRasterPos2s glad_glRasterPos2s
3587 GLAD_API_CALL PFNGLRASTERPOS2SVPROC glad_glRasterPos2sv;
3588 #define glRasterPos2sv glad_glRasterPos2sv
3589 GLAD_API_CALL PFNGLRASTERPOS3DPROC glad_glRasterPos3d;
3590 #define glRasterPos3d glad_glRasterPos3d
3591 GLAD_API_CALL PFNGLRASTERPOS3DVPROC glad_glRasterPos3dv;
3592 #define glRasterPos3dv glad_glRasterPos3dv
3593 GLAD_API_CALL PFNGLRASTERPOS3FPROC glad_glRasterPos3f;
3594 #define glRasterPos3f glad_glRasterPos3f
3595 GLAD_API_CALL PFNGLRASTERPOS3FVPROC glad_glRasterPos3fv;
3596 #define glRasterPos3fv glad_glRasterPos3fv
3597 GLAD_API_CALL PFNGLRASTERPOS3IPROC glad_glRasterPos3i;
3598 #define glRasterPos3i glad_glRasterPos3i
3599 GLAD_API_CALL PFNGLRASTERPOS3IVPROC glad_glRasterPos3iv;
3600 #define glRasterPos3iv glad_glRasterPos3iv
3601 GLAD_API_CALL PFNGLRASTERPOS3SPROC glad_glRasterPos3s;
3602 #define glRasterPos3s glad_glRasterPos3s
3603 GLAD_API_CALL PFNGLRASTERPOS3SVPROC glad_glRasterPos3sv;
3604 #define glRasterPos3sv glad_glRasterPos3sv
3605 GLAD_API_CALL PFNGLRASTERPOS4DPROC glad_glRasterPos4d;
3606 #define glRasterPos4d glad_glRasterPos4d
3607 GLAD_API_CALL PFNGLRASTERPOS4DVPROC glad_glRasterPos4dv;
3608 #define glRasterPos4dv glad_glRasterPos4dv
3609 GLAD_API_CALL PFNGLRASTERPOS4FPROC glad_glRasterPos4f;
3610 #define glRasterPos4f glad_glRasterPos4f
3611 GLAD_API_CALL PFNGLRASTERPOS4FVPROC glad_glRasterPos4fv;
3612 #define glRasterPos4fv glad_glRasterPos4fv
3613 GLAD_API_CALL PFNGLRASTERPOS4IPROC glad_glRasterPos4i;
3614 #define glRasterPos4i glad_glRasterPos4i
3615 GLAD_API_CALL PFNGLRASTERPOS4IVPROC glad_glRasterPos4iv;
3616 #define glRasterPos4iv glad_glRasterPos4iv
3617 GLAD_API_CALL PFNGLRASTERPOS4SPROC glad_glRasterPos4s;
3618 #define glRasterPos4s glad_glRasterPos4s
3619 GLAD_API_CALL PFNGLRASTERPOS4SVPROC glad_glRasterPos4sv;
3620 #define glRasterPos4sv glad_glRasterPos4sv
3621 GLAD_API_CALL PFNGLREADBUFFERPROC glad_glReadBuffer;
3622 #define glReadBuffer glad_glReadBuffer
3623 GLAD_API_CALL PFNGLREADPIXELSPROC glad_glReadPixels;
3624 #define glReadPixels glad_glReadPixels
3625 GLAD_API_CALL PFNGLREADNPIXELSARBPROC glad_glReadnPixelsARB;
3626 #define glReadnPixelsARB glad_glReadnPixelsARB
3627 GLAD_API_CALL PFNGLRECTDPROC glad_glRectd;
3628 #define glRectd glad_glRectd
3629 GLAD_API_CALL PFNGLRECTDVPROC glad_glRectdv;
3630 #define glRectdv glad_glRectdv
3631 GLAD_API_CALL PFNGLRECTFPROC glad_glRectf;
3632 #define glRectf glad_glRectf
3633 GLAD_API_CALL PFNGLRECTFVPROC glad_glRectfv;
3634 #define glRectfv glad_glRectfv
3635 GLAD_API_CALL PFNGLRECTIPROC glad_glRecti;
3636 #define glRecti glad_glRecti
3637 GLAD_API_CALL PFNGLRECTIVPROC glad_glRectiv;
3638 #define glRectiv glad_glRectiv
3639 GLAD_API_CALL PFNGLRECTSPROC glad_glRects;
3640 #define glRects glad_glRects
3641 GLAD_API_CALL PFNGLRECTSVPROC glad_glRectsv;
3642 #define glRectsv glad_glRectsv
3643 GLAD_API_CALL PFNGLRENDERMODEPROC glad_glRenderMode;
3644 #define glRenderMode glad_glRenderMode
3645 GLAD_API_CALL PFNGLRENDERBUFFERSTORAGEPROC glad_glRenderbufferStorage;
3646 #define glRenderbufferStorage glad_glRenderbufferStorage
3647 GLAD_API_CALL PFNGLRENDERBUFFERSTORAGEMULTISAMPLEPROC glad_glRenderbufferStorageMultisample;
3648 #define glRenderbufferStorageMultisample glad_glRenderbufferStorageMultisample
3649 GLAD_API_CALL PFNGLROTATEDPROC glad_glRotated;
3650 #define glRotated glad_glRotated
3651 GLAD_API_CALL PFNGLROTATEFPROC glad_glRotatef;
3652 #define glRotatef glad_glRotatef
```

```
3653 GLAD_API_CALL PFNGLSAMPLECOVERAGEPROC glad_glSampleCoverage;
3654 #define glSampleCoverage glad_glSampleCoverage
3655 GLAD_API_CALL PFNGLSAMPLECOVERAGEARBPROC glad_glSampleCoverageARB;
3656 #define glSampleCoverageARB glad_glSampleCoverageARB
3657 GLAD_API_CALL PFNGLSAMPLEMASKIPROC glad_glSampleMaski;
3658 #define glSampleMaski glad_glSampleMaski
3659 GLAD_API_CALL PFNGLSAMPLERPARAMETERIIVPROC glad_glSamplerParameterIiv;
3660 #define glSamplerParameterIiv glad_glSamplerParameterIiv
3661 GLAD_API_CALL PFNGLSAMPLERPARAMETERIUIVPROC glad_glSamplerParameterIuiv;
3662 #define glSamplerParameterIuiv glad_glSamplerParameterIuiv
3663 GLAD_API_CALL PFNGLSAMPLERPARAMETERFPROC glad_glSamplerParameterf;
3664 #define glSamplerParameterf glad_glSamplerParameterf
3665 GLAD_API_CALL PFNGLSAMPLERPARAMETERFVPROC glad_glSamplerParameterfv;
3666 #define glSamplerParameterfv glad_glSamplerParameterfv
3667 GLAD_API_CALL PFNGLSAMPLERPARAMETERIPROC glad_glSamplerParameteri;
3668 #define glSamplerParameteri glad_glSamplerParameteri
3669 GLAD_API_CALL PFNGLSAMPLERPARAMETERIVPROC glad_glSamplerParameteriv;
3670 #define glSamplerParameteriv glad_glSamplerParameteriv
3671 GLAD_API_CALL PFNGLSCALEDPROC glad_glScaled;
3672 #define glScaled glad_glScaled
3673 GLAD_API_CALL PFNGLSCALEFPROC glad_glScalef;
3674 #define glScalef glad_glScalef
3675 GLAD_API_CALL PFNGLSCISSORPROC glad_glScissor;
3676 #define glScissor glad_glScissor
3677 GLAD_API_CALL PFNGLSECONDARYCOLOR3BPROC glad_glSecondaryColor3b;
3678 #define glSecondaryColor3b glad_glSecondaryColor3b
3679 GLAD_API_CALL PFNGLSECONDARYCOLOR3BVPROC glad_glSecondaryColor3bv;
3680 #define glSecondaryColor3bv glad_glSecondaryColor3bv
3681 GLAD_API_CALL PFNGLSECONDARYCOLOR3DPROC glad_glSecondaryColor3d;
3682 #define glSecondaryColor3d glad_glSecondaryColor3d
3683 GLAD_API_CALL PFNGLSECONDARYCOLOR3DVPROC glad_glSecondaryColor3dv;
3684 #define glSecondaryColor3dv glad_glSecondaryColor3dv
3685 GLAD_API_CALL PFNGLSECONDARYCOLOR3FPROC glad_glSecondaryColor3f;
3686 #define glSecondaryColor3f glad_glSecondaryColor3f
3687 GLAD_API_CALL PFNGLSECONDARYCOLOR3FVPROC glad_glSecondaryColor3fv;
3688 #define glSecondaryColor3fv glad_glSecondaryColor3fv
3689 GLAD_API_CALL PFNGLSECONDARYCOLOR3IPROC glad_glSecondaryColor3i;
3690 #define glSecondaryColor3i glad_glSecondaryColor3i
3691 GLAD_API_CALL PFNGLSECONDARYCOLOR3IVPROC glad_glSecondaryColor3iv;
3692 #define glSecondaryColor3iv glad_glSecondaryColor3iv
3693 GLAD_API_CALL PFNGLSECONDARYCOLOR3SPROC glad_glSecondaryColor3s;
3694 #define glSecondaryColor3s glad_glSecondaryColor3s
3695 GLAD_API_CALL PFNGLSECONDARYCOLOR3SVPROC glad_glSecondaryColor3sv;
3696 #define glSecondaryColor3sv glad_glSecondaryColor3sv
3697 GLAD_API_CALL PFNGLSECONDARYCOLOR3UBPROC glad_glSecondaryColor3ub;
3698 #define glSecondaryColor3ub glad_glSecondaryColor3ub
3699 GLAD_API_CALL PFNGLSECONDARYCOLOR3UBVPROC glad_glSecondaryColor3ubv;
3700 #define glSecondaryColor3ubv glad_glSecondaryColor3ubv
3701 GLAD_API_CALL PFNGLSECONDARYCOLOR3UIPROC glad_glSecondaryColor3ui;
3702 #define glSecondaryColor3ui glad_glSecondaryColor3ui
3703 GLAD_API_CALL PFNGLSECONDARYCOLOR3UIVPROC glad_glSecondaryColor3uiv;
3704 #define glSecondaryColor3uiv glad_glSecondaryColor3uiv
3705 GLAD_API_CALL PFNGLSECONDARYCOLOR3USPROC glad_glSecondaryColor3us;
3706 #define glSecondaryColor3us glad_glSecondaryColor3us
3707 GLAD_API_CALL PFNGLSECONDARYCOLOR3USVPROC glad_glSecondaryColor3usv;
3708 #define glSecondaryColor3usv glad_glSecondaryColor3usv
3709 GLAD_API_CALL PFNGLSECONDARYCOLORP3UIPROC glad_glSecondaryColorP3ui;
3710 #define glSecondaryColorP3ui glad_glSecondaryColorP3ui
3711 GLAD_API_CALL PFNGLSECONDARYCOLORP3UIVPROC glad_glSecondaryColorP3uiv;
3712 #define glSecondaryColorP3uiv glad_glSecondaryColorP3uiv
3713 GLAD_API_CALL PFNGLSECONDARYCOLORPOINTERPROC glad_glSecondaryColorPointer;
3714 #define glSecondaryColorPointer glad_glSecondaryColorPointer
3715 GLAD_API_CALL PFNGLSELECTBUFFERPROC glad_glSelectBuffer;
3716 #define glSelectBuffer glad_glSelectBuffer
3717 GLAD_API_CALL PFNGLSHADEMODELPROC glad_glShadeModel;
3718 #define glShadeModel glad_glShadeModel
3719 GLAD_API_CALL PFNGLSHADERSOURCEPROC glad_glShaderSource;
3720 #define glShaderSource glad_glShaderSource
3721 GLAD_API_CALL PFNGLSTENCILFUNCPROC glad_glStencilFunc;
3722 #define glStencilFunc glad_glStencilFunc
3723 GLAD_API_CALL PFNGLSTENCILFUNCSEPARATEPROC glad_glStencilFuncSeparate;
3724 #define glStencilFuncSeparate glad_glStencilFuncSeparate
3725 GLAD_API_CALL PFNGLSTENCILMASKPROC glad_glStencilMask;
3726 #define glStencilMask glad_glStencilMask
3727 GLAD_API_CALL PFNGLSTENCILMASKSEPARATEPROC glad_glStencilMaskSeparate;
3728 #define glStencilMaskSeparate glad_glStencilMaskSeparate
3729 GLAD_API_CALL PFNGLSTENCILOPPROC glad_glStencilOp;
3730 #define glStencilOp glad_glStencilOp
3731 GLAD_API_CALL PFNGLSTENCILOPSEPARATEPROC glad_glStencilOpSeparate;
3732 #define glStencilOpSeparate glad_glStencilOpSeparate
3733 GLAD_API_CALL PFNGLTEXBUFFERPROC glad_glTexBuffer;
3734 #define glTexBuffer glad_glTexBuffer
3735 GLAD_API_CALL PFNGLTEXCOORD1DPROC glad_glTexCoord1d;
3736 #define glTexCoord1d glad_glTexCoord1d
3737 GLAD_API_CALL PFNGLTEXCOORD1DVPROC glad_glTexCoord1dv;
3738 #define glTexCoord1dv glad_glTexCoord1dv
3739 GLAD_API_CALL PFNGLTEXCOORD1FPROC glad_glTexCoord1f;
```

```
3740 #define glTexCoord1f glad_glTexCoord1f
3741 GLAD_API_CALL PFNGLTEXCOORD1FVPROC glad_glTexCoord1fv;
3742 #define glTexCoord1fv glad_glTexCoord1fv
3743 GLAD_API_CALL PFNGLTEXCOORD1IPROC glad_glTexCoord1i;
3744 #define glTexCoord1i glad_glTexCoord1i
3745 GLAD_API_CALL PFNGLTEXCOORD1IVPROC glad_glTexCoord1iv;
3746 #define glTexCoord1iv glad_glTexCoord1iv
3747 GLAD_API_CALL PFNGLTEXCOORD1SPROC glad_glTexCoord1s;
3748 #define glTexCoord1s glad_glTexCoord1s
3749 GLAD_API_CALL PFNGLTEXCOORD1SVPROC glad_glTexCoord1sv;
3750 #define glTexCoord1sv glad_glTexCoord1sv
3751 GLAD_API_CALL PFNGLTEXCOORD2DPROC glad_glTexCoord2d;
3752 #define glTexCoord2d glad_glTexCoord2d
3753 GLAD_API_CALL PFNGLTEXCOORD2DVPROC glad_glTexCoord2dv;
3754 #define glTexCoord2dv glad_glTexCoord2dv
3755 GLAD_API_CALL PFNGLTEXCOORD2FPROC glad_glTexCoord2f;
3756 #define glTexCoord2f glad_glTexCoord2f
3757 GLAD_API_CALL PFNGLTEXCOORD2FVPROC glad_glTexCoord2fv;
3758 #define glTexCoord2fv glad_glTexCoord2fv
3759 GLAD_API_CALL PFNGLTEXCOORD2IPROC glad_glTexCoord2i;
3760 #define glTexCoord2i glad_glTexCoord2i
3761 GLAD_API_CALL PFNGLTEXCOORD2IVPROC glad_glTexCoord2iv;
3762 #define glTexCoord2iv glad_glTexCoord2iv
3763 GLAD_API_CALL PFNGLTEXCOORD2SPROC glad_glTexCoord2s;
3764 #define glTexCoord2s glad_glTexCoord2s
3765 GLAD_API_CALL PFNGLTEXCOORD2SVPROC glad_glTexCoord2sv;
3766 #define glTexCoord2sv glad_glTexCoord2sv
3767 GLAD_API_CALL PFNGLTEXCOORD3DPROC glad_glTexCoord3d;
3768 #define glTexCoord3d glad_glTexCoord3d
3769 GLAD_API_CALL PFNGLTEXCOORD3DVPROC glad_glTexCoord3dv;
3770 #define glTexCoord3dv glad_glTexCoord3dv
3771 GLAD_API_CALL PFNGLTEXCOORD3FPROC glad_glTexCoord3f;
3772 #define glTexCoord3f glad_glTexCoord3f
3773 GLAD_API_CALL PFNGLTEXCOORD3FVPROC glad_glTexCoord3fv;
3774 #define glTexCoord3fv glad_glTexCoord3fv
3775 GLAD_API_CALL PFNGLTEXCOORD3IPROC glad_glTexCoord3i;
3776 #define glTexCoord3i glad_glTexCoord3i
3777 GLAD_API_CALL PFNGLTEXCOORD3IVPROC glad_glTexCoord3iv;
3778 #define glTexCoord3iv glad_glTexCoord3iv
3779 GLAD_API_CALL PFNGLTEXCOORD3SPROC glad_glTexCoord3s;
3780 #define glTexCoord3s glad_glTexCoord3s
3781 GLAD_API_CALL PFNGLTEXCOORD3SVPROC glad_glTexCoord3sv;
3782 #define glTexCoord3sv glad_glTexCoord3sv
3783 GLAD_API_CALL PFNGLTEXCOORD4DPROC glad_glTexCoord4d;
3784 #define glTexCoord4d glad_glTexCoord4d
3785 GLAD_API_CALL PFNGLTEXCOORD4DVPROC glad_glTexCoord4dv;
3786 #define glTexCoord4dv glad_glTexCoord4dv
3787 GLAD_API_CALL PFNGLTEXCOORD4FPROC glad_glTexCoord4f;
3788 #define glTexCoord4f glad_glTexCoord4f
3789 GLAD_API_CALL PFNGLTEXCOORD4FVPROC glad_glTexCoord4fv;
3790 #define glTexCoord4fv glad_glTexCoord4fv
3791 GLAD_API_CALL PFNGLTEXCOORD4IPROC glad_glTexCoord4i;
3792 #define glTexCoord4i glad_glTexCoord4i
3793 GLAD_API_CALL PFNGLTEXCOORD4IVPROC glad_glTexCoord4iv;
3794 #define glTexCoord4iv glad_glTexCoord4iv
3795 GLAD_API_CALL PFNGLTEXCOORD4SPROC glad_glTexCoord4s;
3796 #define glTexCoord4s glad_glTexCoord4s
3797 GLAD_API_CALL PFNGLTEXCOORD4SVPROC glad_glTexCoord4sv;
3798 #define glTexCoord4sv glad_glTexCoord4sv
3799 GLAD_API_CALL PFNGLTEXCOORDP1UIPROC glad_glTexCoordP1ui;
3800 #define glTexCoordP1ui glad_glTexCoordP1ui
3801 GLAD_API_CALL PFNGLTEXCOORDP1UIVPROC glad_glTexCoordP1uiv;
3802 #define glTexCoordP1uiv glad_glTexCoordP1uiv
3803 GLAD_API_CALL PFNGLTEXCOORDP2UIPROC glad_glTexCoordP2ui;
3804 #define glTexCoordP2ui glad_glTexCoordP2ui
3805 GLAD_API_CALL PFNGLTEXCOORDP2UIVPROC glad_glTexCoordP2uiv;
3806 #define glTexCoordP2uiv glad_glTexCoordP2uiv
3807 GLAD_API_CALL PFNGLTEXCOORDP3UIPROC glad_glTexCoordP3ui;
3808 #define glTexCoordP3ui glad_glTexCoordP3ui
3809 GLAD_API_CALL PFNGLTEXCOORDP3UIVPROC glad_glTexCoordP3uiv;
3810 #define glTexCoordP3uiv glad_glTexCoordP3uiv
3811 GLAD_API_CALL PFNGLTEXCOORDP4UIPROC glad_glTexCoordP4ui;
3812 #define glTexCoordP4ui glad_glTexCoordP4ui
3813 GLAD_API_CALL PFNGLTEXCOORDP4UIVPROC glad_glTexCoordP4uiv;
3814 #define glTexCoordP4uiv glad_glTexCoordP4uiv
3815 GLAD_API_CALL PFNGLTEXCOORDPOINTERPROC glad_glTexCoordPointer;
3816 #define glTexCoordPointer glad_glTexCoordPointer
3817 GLAD_API_CALL PFNGLTEXENVFPROC glad_glTexEnvf;
3818 #define glTexEnvf glad_glTexEnvf
3819 GLAD_API_CALL PFNGLTEXENVFVPROC glad_glTexEnvfv;
3820 #define glTexEnvfv glad_glTexEnvfv
3821 GLAD_API_CALL PFNGLTEXENVIPROC glad_glTexEnvi;
3822 #define glTexEnvi glad_glTexEnvi
3823 GLAD_API_CALL PFNGLTEXENVIVPROC glad_glTexEnviv;
3824 #define glTexEnviv glad_glTexEnviv
3825 GLAD_API_CALL PFNGLTEXGENDPROC glad_glTexGend;
3826 #define glTexGend glad_glTexGend
```

```
3827 GLAD_API_CALL PFNGLTEXGENDVPROC glad_glTexGendv;
3828 #define glTexGendv glad_glTexGendv
3829 GLAD_API_CALL PFNGLTEXGENFPROC glad_glTexGenf;
3830 #define glTexGenf glad_glTexGenf
3831 GLAD_API_CALL PFNGLTEXGENFVPROC glad_glTexGenfv;
3832 #define glTexGenfv glad_glTexGenfv
3833 GLAD_API_CALL PFNGLTEXGENIPROC glad_glTexGeni;
3834 #define glTexGeni glad_glTexGeni
3835 GLAD_API_CALL PFNGLTEXGENIVPROC glad_glTexGeniv;
3836 #define glTexGeniv glad_glTexGeniv
3837 GLAD_API_CALL PFNGLTEXIMAGE1DPROC glad_glTexImage1D;
3838 #define glTexImage1D glad_glTexImage1D
3839 GLAD_API_CALL PFNGLTEXIMAGE2DPROC glad_glTexImage2D;
3840 #define glTexImage2D glad_glTexImage2D
3841 GLAD_API_CALL PFNGLTEXIMAGE2DMULTISAMPLEPROC glad_glTexImage2DMultisample;
3842 #define glTexImage2DMultisample glad_glTexImage2DMultisample
3843 GLAD_API_CALL PFNGLTEXIMAGE3DPROC glad_glTexImage3D;
3844 #define glTexImage3D glad_glTexImage3D
3845 GLAD_API_CALL PFNGLTEXIMAGE3DMULTISAMPLEPROC glad_glTexImage3DMultisample;
3846 #define glTexImage3DMultisample glad_glTexImage3DMultisample
3847 GLAD_API_CALL PFNGLTEXPARAMETERIIVPROC glad_glTexParameterIiv;
3848 #define glTexParameterIiv glad_glTexParameterIiv
3849 GLAD_API_CALL PFNGLTEXPARAMETERIUIVPROC glad_glTexParameterIuiv;
3850 #define glTexParameterIuiv glad_glTexParameterIuiv
3851 GLAD_API_CALL PFNGLTEXPARAMETERFPROC glad_glTexParameterf;
3852 #define glTexParameterf glad_glTexParameterf
3853 GLAD_API_CALL PFNGLTEXPARAMETERFVPROC glad_glTexParameterfv;
3854 #define glTexParameterfv glad_glTexParameterfv
3855 GLAD_API_CALL PFNGLTEXPARAMETERIPROC glad_glTexParameteri;
3856 #define glTexParameteri glad_glTexParameteri
3857 GLAD_API_CALL PFNGLTEXPARAMETERIVPROC glad_glTexParameteriv;
3858 #define glTexParameteriv glad_glTexParameteriv
3859 GLAD_API_CALL PFNGLTEXSUBIMAGE1DPROC glad_glTexSubImage1D;
3860 #define glTexSubImage1D glad_glTexSubImage1D
3861 GLAD_API_CALL PFNGLTEXSUBIMAGE2DPROC glad_glTexSubImage2D;
3862 #define glTexSubImage2D glad_glTexSubImage2D
3863 GLAD_API_CALL PFNGLTEXSUBIMAGE3DPROC glad_glTexSubImage3D;
3864 #define glTexSubImage3D glad_glTexSubImage3D
3865 GLAD_API_CALL PFNGLTRANSFORMFEEDBACKVARYINGSPROC glad_glTransformFeedbackVaryings;
3866 #define glTransformFeedbackVaryings glad_glTransformFeedbackVaryings
3867 GLAD_API_CALL PFNGLTRANSLATEDPROC glad_glTranslated;
3868 #define glTranslated glad_glTranslated
3869 GLAD_API_CALL PFNGLTRANSLATEFPROC glad_glTranslatef;
3870 #define glTranslatef glad_glTranslatef
3871 GLAD_API_CALL PFNGLUNIFORM1FPROC glad_glUniform1f;
3872 #define glUniform1f glad_glUniform1f
3873 GLAD_API_CALL PFNGLUNIFORM1FVPROC glad_glUniform1fv;
3874 #define glUniform1fv glad_glUniform1fv
3875 GLAD_API_CALL PFNGLUNIFORM1IPROC glad_glUniform1i;
3876 #define glUniform1i glad_glUniform1i
3877 GLAD_API_CALL PFNGLUNIFORM1IVPROC glad_glUniform1iv;
3878 #define glUniform1iv glad_glUniform1iv
3879 GLAD_API_CALL PFNGLUNIFORM1UIPROC glad_glUniform1ui;
3880 #define glUniform1ui glad_glUniform1ui
3881 GLAD_API_CALL PFNGLUNIFORM1UIVPROC glad_glUniform1uiv;
3882 #define glUniform1uiv glad_glUniform1uiv
3883 GLAD_API_CALL PFNGLUNIFORM2FPROC glad_glUniform2f;
3884 #define glUniform2f glad_glUniform2f
3885 GLAD_API_CALL PFNGLUNIFORM2FVPROC glad_glUniform2fv;
3886 #define glUniform2fv glad_glUniform2fv
3887 GLAD_API_CALL PFNGLUNIFORM2IPROC glad_glUniform2i;
3888 #define glUniform2i glad_glUniform2i
3889 GLAD_API_CALL PFNGLUNIFORM2IVPROC glad_glUniform2iv;
3890 #define glUniform2iv glad_glUniform2iv
3891 GLAD_API_CALL PFNGLUNIFORM2UIPROC glad_glUniform2ui;
3892 #define glUniform2ui glad_glUniform2ui
3893 GLAD_API_CALL PFNGLUNIFORM2UIVPROC glad_glUniform2uiv;
3894 #define glUniform2uiv glad_glUniform2uiv
3895 GLAD_API_CALL PFNGLUNIFORM3FPROC glad_glUniform3f;
3896 #define glUniform3f glad_glUniform3f
3897 GLAD_API_CALL PFNGLUNIFORM3FVPROC glad_glUniform3fv;
3898 #define glUniform3fv glad_glUniform3fv
3899 GLAD_API_CALL PFNGLUNIFORM3IPROC glad_glUniform3i;
3900 #define glUniform3i glad_glUniform3i
3901 GLAD_API_CALL PFNGLUNIFORM3IVPROC glad_glUniform3iv;
3902 #define glUniform3iv glad_glUniform3iv
3903 GLAD_API_CALL PFNGLUNIFORM3UIPROC glad_glUniform3ui;
3904 #define glUniform3ui glad_glUniform3ui
3905 GLAD_API_CALL PFNGLUNIFORM3UIVPROC glad_glUniform3uiv;
3906 #define glUniform3uiv glad_glUniform3uiv
3907 GLAD_API_CALL PFNGLUNIFORM4FPROC glad_glUniform4f;
3908 #define glUniform4f glad_glUniform4f
3909 GLAD_API_CALL PFNGLUNIFORM4FVPROC glad_glUniform4fv;
3910 #define glUniform4fv glad_glUniform4fv
3911 GLAD_API_CALL PFNGLUNIFORM4IPROC glad_glUniform4i;
3912 #define glUniform4i glad_glUniform4i
3913 GLAD_API_CALL PFNGLUNIFORM4IVPROC glad_glUniform4iv;
```

```
3914 #define glUniform4iv glad_glUniform4iv
3915 GLAD_API_CALL PFNGLUNIFORM4UIPROC glad_glUniform4ui;
3916 #define glUniform4ui glad_glUniform4ui
3917 GLAD_API_CALL PFNGLUNIFORM4UIVPROC glad_glUniform4uiv;
3918 #define glUniform4uiv glad_glUniform4uiv
3919 GLAD_API_CALL PFNGLUNIFORMBLOCKBINDINGPROC glad_glUniformBlockBinding;
3920 #define glUniformBlockBinding glad_glUniformBlockBinding
3921 GLAD_API_CALL PFNGLUNIFORMMATRIX2FVPROC glad_glUniformMatrix2fv;
3922 #define glUniformMatrix2fv glad_glUniformMatrix2fv
3923 GLAD_API_CALL PFNGLUNIFORMMATRIX2X3FVPROC glad_glUniformMatrix2x3fv;
3924 #define glUniformMatrix2x3fv glad_glUniformMatrix2x3fv
3925 GLAD_API_CALL PFNGLUNIFORMMATRIX2X4FVPROC glad_glUniformMatrix2x4fv;
3926 #define glUniformMatrix2x4fv glad_glUniformMatrix2x4fv
3927 GLAD_API_CALL PFNGLUNIFORMMATRIX3FVPROC glad_glUniformMatrix3fv;
3928 #define glUniformMatrix3fv glad_glUniformMatrix3fv
3929 GLAD_API_CALL PFNGLUNIFORMMATRIX3X2FVPROC glad_glUniformMatrix3x2fv;
3930 #define glUniformMatrix3x2fv glad_glUniformMatrix3x2fv
3931 GLAD_API_CALL PFNGLUNIFORMMATRIX3X4FVPROC glad_glUniformMatrix3x4fv;
3932 #define glUniformMatrix3x4fv glad_glUniformMatrix3x4fv
3933 GLAD_API_CALL PFNGLUNIFORMMATRIX4FVPROC glad_glUniformMatrix4fv;
3934 #define glUniformMatrix4fv glad_glUniformMatrix4fv
3935 GLAD_API_CALL PFNGLUNIFORMMATRIX4X2FVPROC glad_glUniformMatrix4x2fv;
3936 #define glUniformMatrix4x2fv glad_glUniformMatrix4x2fv
3937 GLAD_API_CALL PFNGLUNIFORMMATRIX4X3FVPROC glad_glUniformMatrix4x3fv;
3938 #define glUniformMatrix4x3fv glad_glUniformMatrix4x3fv
3939 GLAD_API_CALL PFNGLUNMAPBUFFERPROC glad_glUnmapBuffer;
3940 #define glUnmapBuffer glad_glUnmapBuffer
3941 GLAD_API_CALL PFNGLUSEPROGRAMPROC glad_glUseProgram;
3942 #define glUseProgram glad_glUseProgram
3943 GLAD_API_CALL PFNGLVALIDATEPROGRAMPROC glad_glValidateProgram;
3944 #define glValidateProgram glad_glValidateProgram
3945 GLAD_API_CALL PFNGLVERTEX2DPROC glad_glVertex2d;
3946 #define glVertex2d glad_glVertex2d
3947 GLAD_API_CALL PFNGLVERTEX2DVPROC glad_glVertex2dv;
3948 #define glVertex2dv glad_glVertex2dv
3949 GLAD_API_CALL PFNGLVERTEX2FPROC glad_glVertex2f;
3950 #define glVertex2f glad_glVertex2f
3951 GLAD_API_CALL PFNGLVERTEX2FVPROC glad_glVertex2fv;
3952 #define glVertex2fv glad_glVertex2fv
3953 GLAD_API_CALL PFNGLVERTEX2IPROC glad_glVertex2i;
3954 #define glVertex2i glad_glVertex2i
3955 GLAD_API_CALL PFNGLVERTEX2IVPROC glad_glVertex2iv;
3956 #define glVertex2iv glad_glVertex2iv
3957 GLAD_API_CALL PFNGLVERTEX2SPROC glad_glVertex2s;
3958 #define glVertex2s glad_glVertex2s
3959 GLAD_API_CALL PFNGLVERTEX2SVPROC glad_glVertex2sv;
3960 #define glVertex2sv glad_glVertex2sv
3961 GLAD_API_CALL PFNGLVERTEX3DPROC glad_glVertex3d;
3962 #define glVertex3d glad_glVertex3d
3963 GLAD_API_CALL PFNGLVERTEX3DVPROC glad_glVertex3dv;
3964 #define glVertex3dv glad_glVertex3dv
3965 GLAD_API_CALL PFNGLVERTEX3FPROC glad_glVertex3f;
3966 #define glVertex3f glad_glVertex3f
3967 GLAD_API_CALL PFNGLVERTEX3FVPROC glad_glVertex3fv;
3968 #define glVertex3fv glad_glVertex3fv
3969 GLAD_API_CALL PFNGLVERTEX3IPROC glad_glVertex3i;
3970 #define glVertex3i glad_glVertex3i
3971 GLAD_API_CALL PFNGLVERTEX3IVPROC glad_glVertex3iv;
3972 #define glVertex3iv glad_glVertex3iv
3973 GLAD_API_CALL PFNGLVERTEX3SPROC glad_glVertex3s;
3974 #define glVertex3s glad_glVertex3s
3975 GLAD_API_CALL PFNGLVERTEX3SVPROC glad_glVertex3sv;
3976 #define glVertex3sv glad_glVertex3sv
3977 GLAD_API_CALL PFNGLVERTEX4DPROC glad_glVertex4d;
3978 #define glVertex4d glad_glVertex4d
3979 GLAD_API_CALL PFNGLVERTEX4DVPROC glad_glVertex4dv;
3980 #define glVertex4dv glad_glVertex4dv
3981 GLAD_API_CALL PFNGLVERTEX4FPROC glad_glVertex4f;
3982 #define glVertex4f glad_glVertex4f
3983 GLAD_API_CALL PFNGLVERTEX4FVPROC glad_glVertex4fv;
3984 #define glVertex4fv glad_glVertex4fv
3985 GLAD_API_CALL PFNGLVERTEX4IPROC glad_glVertex4i;
3986 #define glVertex4i glad_glVertex4i
3987 GLAD_API_CALL PFNGLVERTEX4IVPROC glad_glVertex4iv;
3988 #define glVertex4iv glad_glVertex4iv
3989 GLAD_API_CALL PFNGLVERTEX4SPROC glad_glVertex4s;
3990 #define glVertex4s glad_glVertex4s
3991 GLAD_API_CALL PFNGLVERTEX4SVPROC glad_glVertex4sv;
3992 #define glVertex4sv glad_glVertex4sv
3993 GLAD_API_CALL PFNGLVERTEXATTRIB1DPROC glad_glVertexAttrib1d;
3994 #define glVertexAttrib1d glad_glVertexAttrib1d
3995 GLAD_API_CALL PFNGLVERTEXATTRIB1DVPROC glad_glVertexAttrib1dv;
3996 #define glVertexAttrib1dv glad_glVertexAttrib1dv
3997 GLAD_API_CALL PFNGLVERTEXATTRIB1FPROC glad_glVertexAttrib1f;
3998 #define glVertexAttrib1f glad_glVertexAttrib1f
3999 GLAD_API_CALL PFNGLVERTEXATTRIB1FVPROC glad_glVertexAttrib1fv;
4000 #define glVertexAttrib1fv glad_glVertexAttrib1fv
```

```
4001 GLAD_API_CALL PFNGLVERTEXATTRIB1SPROC glad_glVertexAttrib1s;
4002 #define glVertexAttrib1s glad_glVertexAttrib1s
4003 GLAD_API_CALL PFNGLVERTEXATTRIB1SVPROC glad_glVertexAttrib1sv;
4004 #define glVertexAttrib1sv glad_glVertexAttrib1sv
4005 GLAD_API_CALL PFNGLVERTEXATTRIB2DPROC glad_glVertexAttrib2d;
4006 #define glVertexAttrib2d glad_glVertexAttrib2d
4007 GLAD_API_CALL PFNGLVERTEXATTRIB2DVPROC glad_glVertexAttrib2dv;
4008 #define glVertexAttrib2dv glad_glVertexAttrib2dv
4009 GLAD_API_CALL PFNGLVERTEXATTRIB2FPROC glad_glVertexAttrib2f;
4010 #define glVertexAttrib2f glad_glVertexAttrib2f
4011 GLAD_API_CALL PFNGLVERTEXATTRIB2FVPROC glad_glVertexAttrib2fv;
4012 #define glVertexAttrib2fv glad_glVertexAttrib2fv
4013 GLAD_API_CALL PFNGLVERTEXATTRIB2SPROC glad_glVertexAttrib2s;
4014 #define glVertexAttrib2s glad_glVertexAttrib2s
4015 GLAD_API_CALL PFNGLVERTEXATTRIB2SVPROC glad_glVertexAttrib2sv;
4016 #define glVertexAttrib2sv glad_glVertexAttrib2sv
4017 GLAD_API_CALL PFNGLVERTEXATTRIB3DPROC glad_glVertexAttrib3d;
4018 #define glVertexAttrib3d glad_glVertexAttrib3d
4019 GLAD_API_CALL PFNGLVERTEXATTRIB3DVPROC glad_glVertexAttrib3dv;
4020 #define glVertexAttrib3dv glad_glVertexAttrib3dv
4021 GLAD_API_CALL PFNGLVERTEXATTRIB3FPROC glad_glVertexAttrib3f;
4022 #define glVertexAttrib3f glad_glVertexAttrib3f
4023 GLAD_API_CALL PFNGLVERTEXATTRIB3FVPROC glad_glVertexAttrib3fv;
4024 #define glVertexAttrib3fv glad_glVertexAttrib3fv
4025 GLAD_API_CALL PFNGLVERTEXATTRIB3SPROC glad_glVertexAttrib3s;
4026 #define glVertexAttrib3s glad_glVertexAttrib3s
4027 GLAD_API_CALL PFNGLVERTEXATTRIB3SVPROC glad_glVertexAttrib3sv;
4028 #define glVertexAttrib3sv glad_glVertexAttrib3sv
4029 GLAD_API_CALL PFNGLVERTEXATTRIB4NBVPROC glad_glVertexAttrib4Nbv;
4030 #define glVertexAttrib4Nbv glad_glVertexAttrib4Nbv
4031 GLAD_API_CALL PFNGLVERTEXATTRIB4NIVPROC glad_glVertexAttrib4Niv;
4032 #define glVertexAttrib4Niv glad_glVertexAttrib4Niv
4033 GLAD_API_CALL PFNGLVERTEXATTRIB4NSVPROC glad_glVertexAttrib4Nsv;
4034 #define glVertexAttrib4Nsv glad_glVertexAttrib4Nsv
4035 GLAD_API_CALL PFNGLVERTEXATTRIB4NUBPROC glad_glVertexAttrib4Nub;
4036 #define glVertexAttrib4Nub glad_glVertexAttrib4Nub
4037 GLAD_API_CALL PFNGLVERTEXATTRIB4NUBVPROC glad_glVertexAttrib4Nubv;
4038 #define glVertexAttrib4Nubv glad_glVertexAttrib4Nubv
4039 GLAD_API_CALL PFNGLVERTEXATTRIB4NUIVPROC glad_glVertexAttrib4Nuiv;
4040 #define glVertexAttrib4Nuiv glad_glVertexAttrib4Nuiv
4041 GLAD_API_CALL PFNGLVERTEXATTRIB4NUSVPROC glad_glVertexAttrib4Nusv;
4042 #define glVertexAttrib4Nusv glad_glVertexAttrib4Nusv
4043 GLAD_API_CALL PFNGLVERTEXATTRIB4BVPROC glad_glVertexAttrib4bv;
4044 #define glVertexAttrib4bv glad_glVertexAttrib4bv
4045 GLAD_API_CALL PFNGLVERTEXATTRIB4DPROC glad_glVertexAttrib4d;
4046 #define glVertexAttrib4d glad_glVertexAttrib4d
4047 GLAD_API_CALL PFNGLVERTEXATTRIB4DVPROC glad_glVertexAttrib4dv;
4048 #define glVertexAttrib4dv glad_glVertexAttrib4dv
4049 GLAD_API_CALL PFNGLVERTEXATTRIB4FPROC glad_glVertexAttrib4f;
4050 #define glVertexAttrib4f glad_glVertexAttrib4f
4051 GLAD_API_CALL PFNGLVERTEXATTRIB4FVPROC glad_glVertexAttrib4fv;
4052 #define glVertexAttrib4fv glad_glVertexAttrib4fv
4053 GLAD_API_CALL PFNGLVERTEXATTRIB4IVPROC glad_glVertexAttrib4iv;
4054 #define glVertexAttrib4iv glad_glVertexAttrib4iv
4055 GLAD_API_CALL PFNGLVERTEXATTRIB4SPROC glad_glVertexAttrib4s;
4056 #define glVertexAttrib4s glad_glVertexAttrib4s
4057 GLAD_API_CALL PFNGLVERTEXATTRIB4SVPROC glad_glVertexAttrib4sv;
4058 #define glVertexAttrib4sv glad_glVertexAttrib4sv
4059 GLAD_API_CALL PFNGLVERTEXATTRIB4UBVPROC glad_glVertexAttrib4ubv;
4060 #define glVertexAttrib4ubv glad_glVertexAttrib4ubv
4061 GLAD_API_CALL PFNGLVERTEXATTRIB4UIVPROC glad_glVertexAttrib4uiv;
4062 #define glVertexAttrib4uiv glad_glVertexAttrib4uiv
4063 GLAD_API_CALL PFNGLVERTEXATTRIB4USVPROC glad_glVertexAttrib4usv;
4064 #define glVertexAttrib4usv glad_glVertexAttrib4usv
4065 GLAD_API_CALL PFNGLVERTEXATTRIBDIVISORPROC glad_glVertexAttribDivisor;
4066 #define glVertexAttribDivisor glad_glVertexAttribDivisor
4067 GLAD_API_CALL PFNGLVERTEXATTRIBI1IPROC glad_glVertexAttribI1i;
4068 #define glVertexAttribI1i glad_glVertexAttribI1i
4069 GLAD_API_CALL PFNGLVERTEXATTRIBI1IVPROC glad_glVertexAttribI1iv;
4070 #define glVertexAttribI1iv glad_glVertexAttribI1iv
4071 GLAD_API_CALL PFNGLVERTEXATTRIBI1UIPROC glad_glVertexAttribI1ui;
4072 #define glVertexAttribI1ui glad_glVertexAttribI1ui
4073 GLAD_API_CALL PFNGLVERTEXATTRIBI1UIVPROC glad_glVertexAttribI1uiv;
4074 #define glVertexAttribI1uiv glad_glVertexAttribI1uiv
4075 GLAD_API_CALL PFNGLVERTEXATTRIBI2IPROC glad_glVertexAttribI2i;
4076 #define glVertexAttribI2i glad_glVertexAttribI2i
4077 GLAD_API_CALL PFNGLVERTEXATTRIBI2IVPROC glad_glVertexAttribI2iv;
4078 #define glVertexAttribI2iv glad_glVertexAttribI2iv
4079 GLAD_API_CALL PFNGLVERTEXATTRIBI2UIPROC glad_glVertexAttribI2ui;
4080 #define glVertexAttribI2ui glad_glVertexAttribI2ui
4081 GLAD_API_CALL PFNGLVERTEXATTRIBI2UIVPROC glad_glVertexAttribI2uiv;
4082 #define glVertexAttribI2uiv glad_glVertexAttribI2uiv
4083 GLAD_API_CALL PFNGLVERTEXATTRIBI3IPROC glad_glVertexAttribI3i;
4084 #define glVertexAttribI3i glad_glVertexAttribI3i
4085 GLAD_API_CALL PFNGLVERTEXATTRIBI3IVPROC glad_glVertexAttribI3iv;
4086 #define glVertexAttribI3iv glad_glVertexAttribI3iv
4087 GLAD_API_CALL PFNGLVERTEXATTRIBI3UIPROC glad_glVertexAttribI3ui;
```

```
4088 #define glVertexAttribI3ui glad_glVertexAttribI3ui
4089 GLAD_API_CALL PFNGLVERTEXATTRIBI3UIVPROC glad_glVertexAttribI3uiv;
4090 #define glVertexAttribI3uiv glad_glVertexAttribI3uiv
4091 GLAD_API_CALL PFNGLVERTEXATTRIBI4BVPROC glad_glVertexAttribI4bv;
4092 #define glVertexAttribI4bv glad_glVertexAttribI4bv
4093 GLAD_API_CALL PFNGLVERTEXATTRIBI4IPROC glad_glVertexAttribI4i;
4094 #define glVertexAttribI4i glad_glVertexAttribI4i
4095 GLAD_API_CALL PFNGLVERTEXATTRIBI4IVPROC glad_glVertexAttribI4iv;
4096 #define glVertexAttribI4iv glad_glVertexAttribI4iv
4097 GLAD_API_CALL PFNGLVERTEXATTRIBI4SVPROC glad_glVertexAttribI4sv;
4098 #define glVertexAttribI4sv glad_glVertexAttribI4sv
4099 GLAD_API_CALL PFNGLVERTEXATTRIBI4UBVPROC glad_glVertexAttribI4ubv;
4100 #define glVertexAttribI4ubv glad_glVertexAttribI4ubv
4101 GLAD_API_CALL PFNGLVERTEXATTRIBI4UIPROC glad_glVertexAttribI4ui;
4102 #define glVertexAttribI4ui glad_glVertexAttribI4ui
4103 GLAD_API_CALL PFNGLVERTEXATTRIBI4UIVPROC glad_glVertexAttribI4uiv;
4104 #define glVertexAttribI4uiv glad_glVertexAttribI4uiv
4105 GLAD_API_CALL PFNGLVERTEXATTRIBI4USVPROC glad_glVertexAttribI4usv;
4106 #define glVertexAttribI4usv glad_glVertexAttribI4usv
4107 GLAD_API_CALL PFNGLVERTEXATTRIBIPOINTERPROC glad_glVertexAttribIPointer;
4108 #define glVertexAttribIPointer glad_glVertexAttribIPointer
4109 GLAD_API_CALL PFNGLVERTEXATTRIBP1UIPROC glad_glVertexAttribP1ui;
4110 #define glVertexAttribP1ui glad_glVertexAttribP1ui
4111 GLAD_API_CALL PFNGLVERTEXATTRIBP1UIVPROC glad_glVertexAttribP1uiv;
4112 #define glVertexAttribP1uiv glad_glVertexAttribP1uiv
4113 GLAD_API_CALL PFNGLVERTEXATTRIBP2UIPROC glad_glVertexAttribP2ui;
4114 #define glVertexAttribP2ui glad_glVertexAttribP2ui
4115 GLAD_API_CALL PFNGLVERTEXATTRIBP2UIVPROC glad_glVertexAttribP2uiv;
4116 #define glVertexAttribP2uiv glad_glVertexAttribP2uiv
4117 GLAD_API_CALL PFNGLVERTEXATTRIBP3UIPROC glad_glVertexAttribP3ui;
4118 #define glVertexAttribP3ui glad_glVertexAttribP3ui
4119 GLAD_API_CALL PFNGLVERTEXATTRIBP3UIVPROC glad_glVertexAttribP3uiv;
4120 #define glVertexAttribP3uiv glad_glVertexAttribP3uiv
4121 GLAD_API_CALL PFNGLVERTEXATTRIBP4UIPROC glad_glVertexAttribP4ui;
4122 #define glVertexAttribP4ui glad_glVertexAttribP4ui
4123 GLAD_API_CALL PFNGLVERTEXATTRIBP4UIVPROC glad_glVertexAttribP4uiv;
4124 #define glVertexAttribP4uiv glad_glVertexAttribP4uiv
4125 GLAD_API_CALL PFNGLVERTEXATTRIBPOINTERPROC glad_glVertexAttribPointer;
4126 #define glVertexAttribPointer glad_glVertexAttribPointer
4127 GLAD_API_CALL PFNGLVERTEXP2UIPROC glad_glVertexP2ui;
4128 #define glVertexP2ui glad_glVertexP2ui
4129 GLAD_API_CALL PFNGLVERTEXP2UIVPROC glad_glVertexP2uiv;
4130 #define glVertexP2uiv glad_glVertexP2uiv
4131 GLAD_API_CALL PFNGLVERTEXP3UIPROC glad_glVertexP3ui;
4132 #define glVertexP3ui glad_glVertexP3ui
4133 GLAD_API_CALL PFNGLVERTEXP3UIVPROC glad_glVertexP3uiv;
4134 #define glVertexP3uiv glad_glVertexP3uiv
4135 GLAD_API_CALL PFNGLVERTEXP4UIPROC glad_glVertexP4ui;
4136 #define glVertexP4ui glad_glVertexP4ui
4137 GLAD_API_CALL PFNGLVERTEXP4UIVPROC glad_glVertexP4uiv;
4138 #define glVertexP4uiv glad_glVertexP4uiv
4139 GLAD_API_CALL PFNGLVERTEXPOINTERPROC glad_glVertexPointer;
4140 #define glVertexPointer glad_glVertexPointer
4141 GLAD_API_CALL PFNGLVIEWPORTPROC glad_glViewport;
4142 #define glViewport glad_glViewport
4143 GLAD_API_CALL PFNGLWAITSYNCPROC glad_glWaitSync;
4144 #define glWaitSync glad_glWaitSync
4145 GLAD_API_CALL PFNGLWINDOWPOS2DPROC glad_glWindowPos2d;
4146 #define glWindowPos2d glad_glWindowPos2d
4147 GLAD_API_CALL PFNGLWINDOWPOS2DVPROC glad_glWindowPos2dv;
4148 #define glWindowPos2dv glad_glWindowPos2dv
4149 GLAD_API_CALL PFNGLWINDOWPOS2FPROC glad_glWindowPos2f;
4150 #define glWindowPos2f glad_glWindowPos2f
4151 GLAD_API_CALL PFNGLWINDOWPOS2FVPROC glad_glWindowPos2fv;
4152 #define glWindowPos2fv glad_glWindowPos2fv
4153 GLAD_API_CALL PFNGLWINDOWPOS2IPROC glad_glWindowPos2i;
4154 #define glWindowPos2i glad_glWindowPos2i
4155 GLAD_API_CALL PFNGLWINDOWPOS2IVPROC glad_glWindowPos2iv;
4156 #define glWindowPos2iv glad_glWindowPos2iv
4157 GLAD_API_CALL PFNGLWINDOWPOS2SPROC glad_glWindowPos2s;
4158 #define glWindowPos2s glad_glWindowPos2s
4159 GLAD_API_CALL PFNGLWINDOWPOS2SVPROC glad_glWindowPos2sv;
4160 #define glWindowPos2sv glad_glWindowPos2sv
4161 GLAD_API_CALL PFNGLWINDOWPOS3DPROC glad_glWindowPos3d;
4162 #define glWindowPos3d glad_glWindowPos3d
4163 GLAD_API_CALL PFNGLWINDOWPOS3DVPROC glad_glWindowPos3dv;
4164 #define glWindowPos3dv glad_glWindowPos3dv
4165 GLAD_API_CALL PFNGLWINDOWPOS3FPROC glad_glWindowPos3f;
4166 #define glWindowPos3f glad_glWindowPos3f
4167 GLAD_API_CALL PFNGLWINDOWPOS3FVPROC glad_glWindowPos3fv;
4168 #define glWindowPos3fv glad_glWindowPos3fv
4169 GLAD_API_CALL PFNGLWINDOWPOS3IPROC glad_glWindowPos3i;
4170 #define glWindowPos3i glad_glWindowPos3i
4171 GLAD_API_CALL PFNGLWINDOWPOS3IVPROC glad_glWindowPos3iv;
4172 #define glWindowPos3iv glad_glWindowPos3iv
4173 GLAD_API_CALL PFNGLWINDOWPOS3SPROC glad_glWindowPos3s;
4174 #define glWindowPos3s glad_glWindowPos3s
```

```
4175 GLAD_API_CALL PFNGLWINDOWPOS3SVPROC glad_glWindowPos3sv;
4176 #define glWindowPos3sv glad_glWindowPos3sv
4177
4178
4179
4180
4181
4182 GLAD_API_CALL int gladLoadGLUserPtr( GLADuserptrloadfunc load, void *userptr);
4183 GLAD_API_CALL int gladLoadGL( GLADloadfunc load);
4184
4185
4186
4187 #ifdef __cplusplus
4188 }
4189 #endif
4190 #endif
4191
4192 /* Source */
4193 #ifdef GLAD_GL_IMPLEMENTATION
4194 #include <stdio.h>
4195 #include <stdlib.h>
4196 #include <string.h>
4197
4198 #ifndef GLAD_IMPL_UTIL_C_
4199 #define GLAD_IMPL_UTIL_C_
4200
4201 #ifdef _MSC_VER
4202 #define GLAD_IMPL_UTIL_SSCANF sscanf_s
4203 #else
4204 #define GLAD_IMPL_UTIL_SSCANF sscanf
4205 #endif
4206
4207 #endif /* GLAD_IMPL_UTIL_C_ */
4208
4209 #ifdef __cplusplus
4210 extern "C" {
4211 #endif
4212
4213
4214
4215 int GLAD_GL_VERSION_1_0 = 0;
4216 int GLAD_GL_VERSION_1_1 = 0;
4217 int GLAD_GL_VERSION_1_2 = 0;
4218 int GLAD_GL_VERSION_1_3 = 0;
4219 int GLAD_GL_VERSION_1_4 = 0;
4220 int GLAD_GL_VERSION_1_5 = 0;
4221 int GLAD_GL_VERSION_2_0 = 0;
4222 int GLAD_GL_VERSION_2_1 = 0;
4223 int GLAD_GL_VERSION_3_0 = 0;
4224 int GLAD_GL_VERSION_3_1 = 0;
4225 int GLAD_GL_VERSION_3_2 = 0;
4226 int GLAD_GL_VERSION_3_3 = 0;
4227 int GLAD_GL_ARB_multisample = 0;
4228 int GLAD_GL_ARB_robustness = 0;
4229 int GLAD_GL_KHR_debug = 0;
4230
4231
4232
4233 PFNGLACCUMPROC glad_glAccum = NULL;
4234 PFNGLACTIVETEXTUREPROC glad_glActiveTexture = NULL;
4235 PFNGLALPHAFUNCPROC glad_glAlphaFunc = NULL;
4236 PFNGLARETEXTURESRESIDENTPROC glad_glAreTexturesResident = NULL;
4237 PFNGLARRAYELEMENTPROC glad_glArrayElement = NULL;
4238 PFNGLATTACHSHADERPROC glad_glAttachShader = NULL;
4239 PFNGLBEGINPROC glad_glBegin = NULL;
4240 PFNGLBEGINCONDITIONALRENDERPROC glad_glBeginConditionalRender = NULL;
4241 PFNGLBEGINQUERYPROC glad_glBeginQuery = NULL;
4242 PFNGLBEGINTRANSFORMFEEDBACKPROC glad_glBeginTransformFeedback = NULL;
4243 PFNGLBINDATTRIBLOCATIONPROC glad_glBindAttribLocation = NULL;
4244 PFNGLBINDBUFFERPROC glad_glBindBuffer = NULL;
4245 PFNGLBINDBUFFERBASEPROC glad_glBindBufferBase = NULL;
4246 PFNGLBINDBUFFERRANGEPROC glad_glBindBufferRange = NULL;
4247 PFNGLBINDFRAGDATALOCATIONPROC glad_glBindFragDataLocation = NULL;
4248 PFNGLBINDFRAGDATALOCATIONINDEXEDPROC glad_glBindFragDataLocationIndexed = NULL;
4249 PFNGLBINDFRAMEBUFFERPROC glad_glBindFramebuffer = NULL;
4250 PFNGLBINDRENDERBUFFERPROC glad_glBindRenderbuffer = NULL;
4251 PFNGLBINDSAMPLERPROC glad_glBindSampler = NULL;
4252 PFNGLBINDTEXTUREPROC glad_glBindTexture = NULL;
4253 PFNGLBINDVERTEXARRAYPROC glad_glBindVertexArray = NULL;
4254 PFNGLBITMAPPROC glad_glBitmap = NULL;
4255 PFNGLBLENDCOLORPROC glad_glBlendColor = NULL;
4256 PFNGLBLENDEQUATIONPROC glad_glBlendEquation = NULL;
4257 PFNGLBLENDEQUATIONSEPARATEPROC glad_glBlendEquationSeparate = NULL;
4258 PFNGLBLENDFUNCPROC glad_glBlendFunc = NULL;
4259 PFNGLBLENDFUNCSEPARATEPROC glad_glBlendFuncSeparate = NULL;
4260 PFNGLBLITFRAMEBUFFERPROC glad_glBlitFramebuffer = NULL;
4261 PFNGLBUFFERDATAPROC glad_glBufferData = NULL;
```

```
4262 PFNGLBUFFERSUBDATAPROC glad_glBufferSubData = NULL;
4263 PFNGLCALLLISTPROC glad_glCallList = NULL;
4264 PFNGLCALLLISTSPROC glad_glCallLists = NULL;
4265 PFNGLCHECKFRAMEBUFFERSTATUSPROC glad_glCheckFramebufferStatus = NULL;
4266 PFNGLCLAMPCOLORPROC glad_glClampColor = NULL;
4267 PFNGLCLEARPROC glad_glClear = NULL;
4268 PFNGLCLEARACCUMPROC glad_glClearAccum = NULL;
4269 PFNGLCLEARBUFFERFIPROC glad_glClearBufferfi = NULL;
4270 PFNGLCLEARBUFFERFVPROC glad_glClearBufferfv = NULL;
4271 PFNGLCLEARBUFFERIVPROC glad_glClearBufferiv = NULL;
4272 PFNGLCLEARBUFFERUIVPROC glad_glClearBufferuiv = NULL;
4273 PFNGLCLEARCOLORPROC glad_glClearColor = NULL;
4274 PFNGLCLEARDEPTHPROC glad_glClearDepth = NULL;
4275 PFNGLCLEARINDEXPROC glad_glClearIndex = NULL;
4276 PFNGLCLEARSTENCILPROC glad_glClearStencil = NULL;
4277 PFNGLCLIENTACTIVETEXTUREPROC glad_glClientActiveTexture = NULL;
4278 PFNGLCLIENTWAITSYNCPROC glad_glClientWaitSync = NULL;
4279 PFNGLCLIPPLANEPROC glad_glClipPlane = NULL;
4280 PFNGLCOLOR3BPROC glad_glColor3b = NULL;
4281 PFNGLCOLOR3BVPROC glad_glColor3bv = NULL;
4282 PFNGLCOLOR3DPROC glad_glColor3d = NULL;
4283 PFNGLCOLOR3DVPROC glad_glColor3dv = NULL;
4284 PFNGLCOLOR3FPROC glad_glColor3f = NULL;
4285 PFNGLCOLOR3FVPROC glad_glColor3fv = NULL;
4286 PFNGLCOLOR3IPROC glad_glColor3i = NULL;
4287 PFNGLCOLOR3IVPROC glad_glColor3iv = NULL;
4288 PFNGLCOLOR3SPROC glad_glColor3s = NULL;
4289 PFNGLCOLOR3SVPROC glad_glColor3sv = NULL;
4290 PFNGLCOLOR3UBPROC glad_glColor3ub = NULL;
4291 PFNGLCOLOR3UBVPROC glad_glColor3ubv = NULL;
4292 PFNGLCOLOR3UIPROC glad_glColor3ui = NULL;
4293 PFNGLCOLOR3UIVPROC glad_glColor3uiv = NULL;
4294 PFNGLCOLOR3USPROC glad_glColor3us = NULL;
4295 PFNGLCOLOR3USVPROC glad_glColor3usv = NULL;
4296 PFNGLCOLOR4BPROC glad_glColor4b = NULL;
4297 PFNGLCOLOR4BVPROC glad_glColor4bv = NULL;
4298 PFNGLCOLOR4DPROC glad_glColor4d = NULL;
4299 PFNGLCOLOR4DVPROC glad_glColor4dv = NULL;
4300 PFNGLCOLOR4FPROC glad_glColor4f = NULL;
4301 PFNGLCOLOR4FVPROC glad_glColor4fv = NULL;
4302 PFNGLCOLOR4IPROC glad_glColor4i = NULL;
4303 PFNGLCOLOR4IVPROC glad_glColor4iv = NULL;
4304 PFNGLCOLOR4SPROC glad_glColor4s = NULL;
4305 PFNGLCOLOR4SVPROC glad_glColor4sv = NULL;
4306 PFNGLCOLOR4UBPROC glad_glColor4ub = NULL;
4307 PFNGLCOLOR4UBVPROC glad_glColor4ubv = NULL;
4308 PFNGLCOLOR4UIPROC glad_glColor4ui = NULL;
4309 PFNGLCOLOR4UIVPROC glad_glColor4uiv = NULL;
4310 PFNGLCOLOR4USPROC glad_glColor4us = NULL;
4311 PFNGLCOLOR4USVPROC glad_glColor4usv = NULL;
4312 PFNGLCOLORMASKPROC glad_glColorMask = NULL;
4313 PFNGLCOLORMASKIPROC glad_glColorMaski = NULL;
4314 PFNGLCOLORMATERIALPROC glad_glColorMaterial = NULL;
4315 PFNGLCOLORP3UIPROC glad_glColorP3ui = NULL;
4316 PFNGLCOLORP3UIVPROC glad_glColorP3uiv = NULL;
4317 PFNGLCOLORP4UIPROC glad_glColorP4ui = NULL;
4318 PFNGLCOLORP4UIVPROC glad_glColorP4uiv = NULL;
4319 PFNGLCOLORPOINTERPROC glad_glColorPointer = NULL;
4320 PFNGLCOMPILESHADERPROC glad_glCompileShader = NULL;
4321 PFNGLCOMPRESSEDTEXIMAGE1DPROC glad_glCompressedTexImage1D = NULL;
4322 PFNGLCOMPRESSEDTEXIMAGE2DPROC glad_glCompressedTexImage2D = NULL;
4323 PFNGLCOMPRESSEDTEXIMAGE3DPROC glad_glCompressedTexImage3D = NULL;
4324 PFNGLCOMPRESSEDTEXSUBIMAGE1DPROC glad_glCompressedTexSubImage1D = NULL;
4325 PFNGLCOMPRESSEDTEXSUBIMAGE2DPROC glad_glCompressedTexSubImage2D = NULL;
4326 PFNGLCOMPRESSEDTEXSUBIMAGE3DPROC glad_glCompressedTexSubImage3D = NULL;
4327 PFNGLCOPYBUFFERSUBDATAPROC glad_glCopyBufferSubData = NULL;
4328 PFNGLCOPYPIXELSPROC glad_glCopyPixels = NULL;
4329 PFNGLCOPYTEXIMAGE1DPROC glad_glCopyTexImage1D = NULL;
4330 PFNGLCOPYTEXIMAGE2DPROC glad_glCopyTexImage2D = NULL;
4331 PFNGLCOPYTEXSUBIMAGE1DPROC glad_glCopyTexSubImage1D = NULL;
4332 PFNGLCOPYTEXSUBIMAGE2DPROC glad_glCopyTexSubImage2D = NULL;
4333 PFNGLCOPYTEXSUBIMAGE3DPROC glad_glCopyTexSubImage3D = NULL;
4334 PFNGLCREATEPROGRAMPROC glad_glCreateProgram = NULL;
4335 PFNGLCREATESHADERPROC glad_glCreateShader = NULL;
4336 PFNGLCULLFACEPROC glad_glCullFace = NULL;
4337 PFNGLDEBUGMESSAGECALLBACKPROC glad_glDebugMessageCallback = NULL;
4338 PFNGLDEBUGMESSAGECONTROLPROC glad_glDebugMessageControl = NULL;
4339 PFNGLDEBUGMESSAGEINSERTPROC glad_glDebugMessageInsert = NULL;
4340 PFNGLDELETEBUFFERSPROC glad_glDeleteBuffers = NULL;
4341 PFNGLDELETEFRAMEBUFFERSPROC glad_glDeleteFramebuffers = NULL;
4342 PFNGLDELETELISTSPROC glad_glDeleteLists = NULL;
4343 PFNGLDELETEPROGRAMPROC glad_glDeleteProgram = NULL;
4344 PFNGLDELETEQUERIESPROC glad_glDeleteQueries = NULL;
4345 PFNGLDELETERENDERBUFFERSPROC glad_glDeleteRenderbuffers = NULL;
4346 PFNGLDELETESAMPLERSPROC glad_glDeleteSamplers = NULL;
4347 PFNGLDELETESHADERPROC glad_glDeleteShader = NULL;
4348 PFNGLDELETESYNCPROC glad_glDeleteSync = NULL;
```

```
4349 PFNGLDELETETEXTURESPROC glad_glDeleteTextures = NULL;
4350 PFNGLDELETEVERTEXARRAYSPROC glad_glDeleteVertexArrays = NULL;
4351 PFNGLDEPTHFUNCPROC glad_glDepthFunc = NULL;
4352 PFNGLDEPTHMASKPROC glad_glDepthMask = NULL;
4353 PFNGLDEPTHRANGEPROC glad_glDepthRange = NULL;
4354 PFNGLDETACHSHADERPROC glad_glDetachShader = NULL;
4355 PFNGLDISABLEPROC glad_glDisable = NULL;
4356 PFNGLDISABLECLIENTSTATEPROC glad_glDisableClientState = NULL;
4357 PFNGLDISABLEVERTEXATTRIBARRAYPROC glad_glDisableVertexAttribArray = NULL;
4358 PFNGLDISABLEIPROC glad_glDisablei = NULL;
4359 PFNGLDRAWARRAYSPROC glad_glDrawArrays = NULL;
4360 PFNGLDRAWARRAYSINSTANCEDPROC glad_glDrawArraysInstanced = NULL;
4361 PFNGLDRAWBUFFERPROC glad_glDrawBuffer = NULL;
4362 PFNGLDRAWBUFFERSPROC glad_glDrawBuffers = NULL;
4363 PFNGLDRAWELEMENTSPROC glad_glDrawElements = NULL;
4364 PFNGLDRAWELEMENTSBASEVERTEXPROC glad_glDrawElementsBaseVertex = NULL;
4365 PFNGLDRAWELEMENTSINSTANCEDPROC glad_glDrawElementsInstanced = NULL;
4366 PFNGLDRAWELEMENTSINSTANCEDBASEVERTEXPROC glad_glDrawElementsInstancedBaseVertex = NULL;
4367 PFNGLDRAWPIXELSPROC glad_glDrawPixels = NULL;
4368 PFNGLDRAWRANGEELEMENTSPROC glad_glDrawRangeElements = NULL;
4369 PFNGLDRAWRANGEELEMENTSBASEVERTEXPROC glad_glDrawRangeElementsBaseVertex = NULL;
4370 PFNGLEDGEFLAGPROC glad_glEdgeFlag = NULL;
4371 PFNGLEDGEFLAGPOINTERPROC glad_glEdgeFlagPointer = NULL;
4372 PFNGLEDGEFLAGVPROC glad_glEdgeFlagv = NULL;
4373 PFNGLENABLEPROC glad_glEnable = NULL;
4374 PFNGLENABLECLIENTSTATEPROC glad_glEnableClientState = NULL;
4375 PFNGLENABLEVERTEXATTRIBARRAYPROC glad_glEnableVertexAttribArray = NULL;
4376 PFNGLENABLEIPROC glad_glEnablei = NULL;
4377 PFNGLENDPROC glad_glEnd = NULL;
4378 PFNGLENDCONDITIONALRENDERPROC glad_glEndConditionalRender = NULL;
4379 PFNGLENDLISTPROC glad_glEndList = NULL;
4380 PFNGLENDQUERYPROC glad_glEndQuery = NULL;
4381 PFNGLENDTRANSFORMFEEDBACKPROC glad_glEndTransformFeedback = NULL;
4382 PFNGLEVALCOORD1DPROC glad_glEvalCoord1d = NULL;
4383 PFNGLEVALCOORD1DVPROC glad_glEvalCoord1dv = NULL;
4384 PFNGLEVALCOORD1FPROC glad_glEvalCoord1f = NULL;
4385 PFNGLEVALCOORD1FVPROC glad_glEvalCoord1fv = NULL;
4386 PFNGLEVALCOORD2DPROC glad_glEvalCoord2d = NULL;
4387 PFNGLEVALCOORD2DVPROC glad_glEvalCoord2dv = NULL;
4388 PFNGLEVALCOORD2FPROC glad_glEvalCoord2f = NULL;
4389 PFNGLEVALCOORD2FVPROC glad_glEvalCoord2fv = NULL;
4390 PFNGLEVALMESH1PROC glad_glEvalMesh1 = NULL;
4391 PFNGLEVALMESH2PROC glad_glEvalMesh2 = NULL;
4392 PFNGLEVALPOINT1PROC glad_glEvalPoint1 = NULL;
4393 PFNGLEVALPOINT2PROC glad_glEvalPoint2 = NULL;
4394 PFNGLFEEDBACKBUFFERPROC glad_glFeedbackBuffer = NULL;
4395 PFNGLFENCESYNCPROC glad_glFenceSync = NULL;
4396 PFNGLFINISHPROC glad_glFinish = NULL;
4397 PFNGLFLUSHPROC glad_glFlush = NULL;
4398 PFNGLFLUSHMAPPEDBUFFERRANGEPROC glad_glFlushMappedBufferRange = NULL;
4399 PFNGLFOGCOORDPOINTERPROC glad_glFogCoordPointer = NULL;
4400 PFNGLFOGCOORDDPROC glad_glFogCoordd = NULL;
4401 PFNGLFOGCOORDDVPROC glad_glFogCoorddv = NULL;
4402 PFNGLFOGCOORDFPROC glad_glFogCoordf = NULL;
4403 PFNGLFOGCOORDFVPROC glad_glFogCoordfv = NULL;
4404 PFNGLFOGFPROC glad_glFogf = NULL;
4405 PFNGLFOGFVPROC glad_glFogfv = NULL;
4406 PFNGLFOGIPROC glad_glFogi = NULL;
4407 PFNGLFOGIVPROC glad_glFogiv = NULL;
4408 PFNGLFRAMEBUFFERRENDERBUFFERPROC glad_glFramebufferRenderbuffer = NULL;
4409 PFNGLFRAMEBUFFERTEXTUREPROC glad_glFramebufferTexture = NULL;
4410 PFNGLFRAMEBUFFERTEXTURE1DPROC glad_glFramebufferTexture1D = NULL;
4411 PFNGLFRAMEBUFFERTEXTURE2DPROC glad_glFramebufferTexture2D = NULL;
4412 PFNGLFRAMEBUFFERTEXTURE3DPROC glad_glFramebufferTexture3D = NULL;
4413 PFNGLFRAMEBUFFERTEXTURELAYERPROC glad_glFramebufferTextureLayer = NULL;
4414 PFNGLFRONTFACEPROC glad_glFrontFace = NULL;
4415 PFNGLFRUSTUMPROC glad_glFrustum = NULL;
4416 PFNGLGENBUFFERSPROC glad_glGenBuffers = NULL;
4417 PFNGLGENFRAMEBUFFERSPROC glad_glGenFramebuffers = NULL;
4418 PFNGLGENLISTSPROC glad_glGenLists = NULL;
4419 PFNGLGENQUERIESPROC glad_glGenQueries = NULL;
4420 PFNGLGENRENDERBUFFERSPROC glad_glGenRenderbuffers = NULL;
4421 PFNGLGENSAMPLERSPROC glad_glGenSamplers = NULL;
4422 PFNGLGENTEXTURESPROC glad_glGenTextures = NULL;
4423 PFNGLGENVERTEXARRAYSPROC glad_glGenVertexArrays = NULL;
4424 PFNGLGENERATEMIPMAPPROC glad_glGenerateMipmap = NULL;
4425 PFNGLGETACTIVEATTRIBPROC glad_glGetActiveAttrib = NULL;
4426 PFNGLGETACTIVEUNIFORMPROC glad_glGetActiveUniform = NULL;
4427 PFNGLGETACTIVEUNIFORMBLOCKNAMEPROC glad_glGetActiveUniformBlockName = NULL;
4428 PFNGLGETACTIVEUNIFORMBLOCKIVPROC glad_glGetActiveUniformBlockiv = NULL;
4429 PFNGLGETACTIVEUNIFORMNAMEPROC glad_glGetActiveUniformName = NULL;
4430 PFNGLGETACTIVEUNIFORMSIVPROC glad_glGetActiveUniformsiv = NULL;
4431 PFNGLGETATTACHEDSHADERSPROC glad_glGetAttachedShaders = NULL;
4432 PFNGLGETATTRIBLOCATIONPROC glad_glGetAttribLocation = NULL;
4433 PFNGLGETBOOLEANI_VPROC glad_glGetBooleani_v = NULL;
4434 PFNGLGETBOOLEANVPROC glad_glGetBooleanv = NULL;
4435 PFNGLGETBUFFERPARAMETERI64VPROC glad_glGetBufferParameteri64v = NULL;
```

```
4436 PFNGLGETBUFFERPARAMETERIVPROC glad_glGetBufferParameteriv = NULL;
4437 PFNGLGETBUFFERPOINTERVPROC glad_glGetBufferPointerv = NULL;
4438 PFNGLGETBUFFERSUBDATAPROC glad_glGetBufferSubData = NULL;
4439 PFNGLGETCLIPPLANEPROC glad_glGetClipPlane = NULL;
4440 PFNGLGETCOMPRESSEDTEXIMAGEPROC glad_glGetCompressedTexImage = NULL;
4441 PFNGLGETDEBUGMESSAGELOGPROC glad_glGetDebugMessageLog = NULL;
4442 PFNGLGETDOUBLEVPROC glad_glGetDoublev = NULL;
4443 PFNGLGETERRORPROC glad_glGetError = NULL;
4444 PFNGLGETFLOATVPROC glad_glGetFloatv = NULL;
4445 PFNGLGETFRAGDATAINDEXPROC glad_glGetFragDataIndex = NULL;
4446 PFNGLGETFRAGDATALOCATIONPROC glad_glGetFragDataLocation = NULL;
4447 PFNGLGETFRAMEBUFFERATTACHMENTPARAMETERIVPROC glad_glGetFramebufferAttachmentParameteriv = NULL;
4448 PFNGLGETGRAPHICSRESETSTATUSARBPROC glad_glGetGraphicsResetStatusARB = NULL;
4449 PFNGLGETINTEGER64I_VPROC glad_glGetInteger64i_v = NULL;
4450 PFNGLGETINTEGER64VPROC glad_glGetInteger64v = NULL;
4451 PFNGLGETINTEGERI_VPROC glad_glGetIntegeri_v = NULL;
4452 PFNGLGETINTEGERVPROC glad_glGetIntegerv = NULL;
4453 PFNGLGETLIGHTFVPROC glad_glGetLightfv = NULL;
4454 PFNGLGETLIGHTIVPROC glad_glGetLightiv = NULL;
4455 PFNGLGETMAPDVPROC glad_glGetMapdv = NULL;
4456 PFNGLGETMAPFVPROC glad_glGetMapfv = NULL;
4457 PFNGLGETMAPIVPROC glad_glGetMapiv = NULL;
4458 PFNGLGETMATERIALFVPROC glad_glGetMaterialfv = NULL;
4459 PFNGLGETMATERIALIVPROC glad_glGetMaterialiv = NULL;
4460 PFNGLGETMULTISAMPLEFVPROC glad_glGetMultisamplefv = NULL;
4461 PFNGLGETOBJECTLABELPROC glad_glGetObjectLabel = NULL;
4462 PFNGLGETOBJECTPTRLABELPROC glad_glGetObjectPtrLabel = NULL;
4463 PFNGLGETPIXELMAPFVPROC glad_glGetPixelMapfv = NULL;
4464 PFNGLGETPIXELMAPUIVPROC glad_glGetPixelMapuiv = NULL;
4465 PFNGLGETPIXELMAPUSVPROC glad_glGetPixelMapusv = NULL;
4466 PFNGLGETPOINTERVPROC glad_glGetPointerv = NULL;
4467 PFNGLGETPOLYGONSTIPPLEPROC glad_glGetPolygonStipple = NULL;
4468 PFNGLGETPROGRAMINFOLOGPROC glad_glGetProgramInfoLog = NULL;
4469 PFNGLGETPROGRAMIVPROC glad_glGetProgramiv = NULL;
4470 PFNGLGETQUERYOBJECTI64VPROC glad_glGetQueryObjecti64v = NULL;
4471 PFNGLGETQUERYOBJECTIVPROC glad_glGetQueryObjectiv = NULL;
4472 PFNGLGETQUERYOBJECTUI64VPROC glad_glGetQueryObjectui64v = NULL;
4473 PFNGLGETQUERYOBJECTUIVPROC glad_glGetQueryObjectuiv = NULL;
4474 PFNGLGETQUERYIVPROC glad_glGetQueryiv = NULL;
4475 PFNGLGETRENDERBUFFERPARAMETERIVPROC glad_glGetRenderbufferParameteriv = NULL;
4476 PFNGLGETSAMPLERPARAMETERIIVPROC glad_glGetSamplerParameterIiv = NULL;
4477 PFNGLGETSAMPLERPARAMETERIUIVPROC glad_glGetSamplerParameterIuiv = NULL;
4478 PFNGLGETSAMPLERPARAMETERFVPROC glad_glGetSamplerParameterfv = NULL;
4479 PFNGLGETSAMPLERPARAMETERIVPROC glad_glGetSamplerParameteriv = NULL;
4480 PFNGLGETSHADERINFOLOGPROC glad_glGetShaderInfoLog = NULL;
4481 PFNGLGETSHADERSOURCEPROC glad_glGetShaderSource = NULL;
4482 PFNGLGETSHADERIVPROC glad_glGetShaderiv = NULL;
4483 PFNGLGETSTRINGPROC glad_glGetString = NULL;
4484 PFNGLGETSTRINGIPROC glad_glGetStringi = NULL;
4485 PFNGLGETSYNCIVPROC glad_glGetSynciv = NULL;
4486 PFNGLGETTEXENVFVPROC glad_glGetTexEnvfv = NULL;
4487 PFNGLGETTEXENVIVPROC glad_glGetTexEnviv = NULL;
4488 PFNGLGETTEXGENDVPROC glad_glGetTexGendv = NULL;
4489 PFNGLGETTEXGENFVPROC glad_glGetTexGenfv = NULL;
4490 PFNGLGETTEXGENIVPROC glad_glGetTexGeniv = NULL;
4491 PFNGLGETTEXIMAGEPROC glad_glGetTexImage = NULL;
4492 PFNGLGETTEXLEVELPARAMETERFVPROC glad_glGetTexLevelParameterfv = NULL;
4493 PFNGLGETTEXLEVELPARAMETERIVPROC glad_glGetTexLevelParameteriv = NULL;
4494 PFNGLGETTEXPARAMETERIIVPROC glad_glGetTexParameterIiv = NULL;
4495 PFNGLGETTEXPARAMETERIUIVPROC glad_glGetTexParameterIuiv = NULL;
4496 PFNGLGETTEXPARAMETERFVPROC glad_glGetTexParameterfv = NULL;
4497 PFNGLGETTEXPARAMETERIVPROC glad_glGetTexParameteriv = NULL;
4498 PFNGLGETTRANSFORMFEEDBACKVARYINGPROC glad_glGetTransformFeedbackVarying = NULL;
4499 PFNGLGETUNIFORMBLOCKINDEXPROC glad_glGetUniformBlockIndex = NULL;
4500 PFNGLGETUNIFORMINDICESPROC glad_glGetUniformIndices = NULL;
4501 PFNGLGETUNIFORMLOCATIONPROC glad_glGetUniformLocation = NULL;
4502 PFNGLGETUNIFORMFVPROC glad_glGetUniformfv = NULL;
4503 PFNGLGETUNIFORMIVPROC glad_glGetUniformiv = NULL;
4504 PFNGLGETUNIFORMUIVPROC glad_glGetUniformuiv = NULL;
4505 PFNGLGETVERTEXATTRIBIIVPROC glad_glGetVertexAttribIiv = NULL;
4506 PFNGLGETVERTEXATTRIBIUIVPROC glad_glGetVertexAttribIuiv = NULL;
4507 PFNGLGETVERTEXATTRIBPOINTERVPROC glad_glGetVertexAttribPointerv = NULL;
4508 PFNGLGETVERTEXATTRIBDVPROC glad_glGetVertexAttribdv = NULL;
4509 PFNGLGETVERTEXATTRIBFVPROC glad_glGetVertexAttribfv = NULL;
4510 PFNGLGETVERTEXATTRIBIVPROC glad_glGetVertexAttribiv = NULL;
4511 PFNGLGETNCOLORTABLEARBPROC glad_glGetnColorTableARB = NULL;
4512 PFNGLGETNCOMPRESSEDTEXIMAGEARBPROC glad_glGetnCompressedTexImageARB = NULL;
4513 PFNGLGETNCONVOLUTIONFILTERARBPROC glad_glGetnConvolutionFilterARB = NULL;
4514 PFNGLGETNHISTOGRAMARBPROC glad_glGetnHistogramARB = NULL;
4515 PFNGLGETNMAPDVARBPROC glad_glGetnMapdvARB = NULL;
4516 PFNGLGETNMAPFVARBPROC glad_glGetnMapfvARB = NULL;
4517 PFNGLGETNMAPIVARBPROC glad_glGetnMapivARB = NULL;
4518 PFNGLGETNMINMAXARBPROC glad_glGetnMinmaxARB = NULL;
4519 PFNGLGETNPIXELMAPFVARBPROC glad_glGetnPixelMapfvARB = NULL;
4520 PFNGLGETNPIXELMAPUIVARBPROC glad_glGetnPixelMapuivARB = NULL;
4521 PFNGLGETNPIXELMAPUSVARBPROC glad_glGetnPixelMapusvARB = NULL;
4522 PFNGLGETNPOLYGONSTIPPLEARBPROC glad_glGetnPolygonStippleARB = NULL;
```

```
4523 PFNGLGETNSEPARABLEFILTERARBPROC glad_glGetnSeparableFilterARB = NULL;
4524 PFNGLGETNTEXIMAGEARBPROC glad_glGetnTexImageARB = NULL;
4525 PFNGLGETNUNIFORMDVARBPROC glad_glGetnUniformdvARB = NULL;
4526 PFNGLGETNUNIFORMFVARBPROC glad_glGetnUniformfvARB = NULL;
4527 PFNGLGETNUNIFORMIVARBPROC glad_glGetnUniformivARB = NULL;
4528 PFNGLGETNUNIFORMUIVARBPROC glad_glGetnUniformuivARB = NULL;
4529 PFNGLHINTPROC glad_glHint = NULL;
4530 PFNGLINDEXMASKPROC glad_glIndexMask = NULL;
4531 PFNGLINDEXPOINTERPROC glad_glIndexPointer = NULL;
4532 PFNGLINDEXDPROC glad_glIndexd = NULL;
4533 PFNGLINDEXDVPROC glad_glIndexdv = NULL;
4534 PFNGLINDEXFPROC glad_glIndexf = NULL;
4535 PFNGLINDEXFVPROC glad_glIndexfv = NULL;
4536 PFNGLINDEXIPROC glad_glIndexi = NULL;
4537 PFNGLINDEXIVPROC glad_glIndexiv = NULL;
4538 PFNGLINDEXSPROC glad_glIndexs = NULL;
4539 PFNGLINDEXSVPROC glad_glIndexsv = NULL;
4540 PFNGLINDEXUBPROC glad_glIndexub = NULL;
4541 PFNGLINDEXUBVPROC glad_glIndexubv = NULL;
4542 PFNGLINITNAMESPROC glad_glInitNames = NULL;
4543 PFNGLINTERLEAVEDARRAYSPROC glad_glInterleavedArrays = NULL;
4544 PFNGLISBUFFERPROC glad_glIsBuffer = NULL;
4545 PFNGLISENABLEDPROC glad_glIsEnabled = NULL;
4546 PFNGLISENABLEDIPROC glad_glIsEnabledi = NULL;
4547 PFNGLISFRAMEBUFFERPROC glad_glIsFramebuffer = NULL;
4548 PFNGLISLISTPROC glad_glIsList = NULL;
4549 PFNGLISPROGRAMPROC glad_glIsProgram = NULL;
4550 PFNGLISQUERYPROC glad_glIsQuery = NULL;
4551 PFNGLISRENDERBUFFERPROC glad_glIsRenderbuffer = NULL;
4552 PFNGLISSAMPLERPROC glad_glIsSampler = NULL;
4553 PFNGLISSHADERPROC glad_glIsShader = NULL;
4554 PFNGLISSYNCPROC glad_glIsSync = NULL;
4555 PFNGLISTEXTUREPROC glad_glIsTexture = NULL;
4556 PFNGLISVERTEXARRAYPROC glad_glIsVertexArray = NULL;
4557 PFNGLLIGHTMODELFPROC glad_glLightModelf = NULL;
4558 PFNGLLIGHTMODELFVPROC glad_glLightModelfv = NULL;
4559 PFNGLLIGHTMODELIPROC glad_glLightModeli = NULL;
4560 PFNGLLIGHTMODELIVPROC glad_glLightModeliv = NULL;
4561 PFNGLLIGHTFPROC glad_glLightf = NULL;
4562 PFNGLLIGHTFVPROC glad_glLightfv = NULL;
4563 PFNGLLIGHTIPROC glad_glLighti = NULL;
4564 PFNGLLIGHTIVPROC glad_glLightiv = NULL;
4565 PFNGLLINESTIPPLEPROC glad_glLineStipple = NULL;
4566 PFNGLLINEWIDTHPROC glad_glLineWidth = NULL;
4567 PFNGLLINKPROGRAMPROC glad_glLinkProgram = NULL;
4568 PFNGLLISTBASEPROC glad_glListBase = NULL;
4569 PFNGLLOADIDENTITYPROC glad_glLoadIdentity = NULL;
4570 PFNGLLOADMATRIXDPROC glad_glLoadMatrixd = NULL;
4571 PFNGLLOADMATRIXFPROC glad_glLoadMatrixf = NULL;
4572 PFNGLLOADNAMEPROC glad_glLoadName = NULL;
4573 PFNGLLOADTRANSPOSEMATRIXDPROC glad_glLoadTransposeMatrixd = NULL;
4574 PFNGLLOADTRANSPOSEMATRIXFPROC glad_glLoadTransposeMatrixf = NULL;
4575 PFNGLLOGICOPPROC glad_glLogicOp = NULL;
4576 PFNGLMAP1DPROC glad_glMap1d = NULL;
4577 PFNGLMAP1FPROC glad_glMap1f = NULL;
4578 PFNGLMAP2DPROC glad_glMap2d = NULL;
4579 PFNGLMAP2FPROC glad_glMap2f = NULL;
4580 PFNGLMAPBUFFERPROC glad_glMapBuffer = NULL;
4581 PFNGLMAPBUFFERRANGEPROC glad_glMapBufferRange = NULL;
4582 PFNGLMAPGRID1DPROC glad_glMapGrid1d = NULL;
4583 PFNGLMAPGRID1FPROC glad_glMapGrid1f = NULL;
4584 PFNGLMAPGRID2DPROC glad_glMapGrid2d = NULL;
4585 PFNGLMAPGRID2FPROC glad_glMapGrid2f = NULL;
4586 PFNGLMATERIALFPROC glad_glMaterialf = NULL;
4587 PFNGLMATERIALFVPROC glad_glMaterialfv = NULL;
4588 PFNGLMATERIALIPROC glad_glMateriali = NULL;
4589 PFNGLMATERIALIVPROC glad_glMaterialiv = NULL;
4590 PFNGLMATRIXMODEPROC glad_glMatrixMode = NULL;
4591 PFNGLMULTMATRIXDPROC glad_glMultMatrixd = NULL;
4592 PFNGLMULTMATRIXFPROC glad_glMultMatrixf = NULL;
4593 PFNGLMULTTRANSPOSEMATRIXDPROC glad_glMultTransposeMatrixd = NULL;
4594 PFNGLMULTTRANSPOSEMATRIXFPROC glad_glMultTransposeMatrixf = NULL;
4595 PFNGLMULTIDRAWARRAYSPROC glad_glMultiDrawArrays = NULL;
4596 PFNGLMULTIDRAWELEMENTSPROC glad_glMultiDrawElements = NULL;
4597 PFNGLMULTIDRAWELEMENTSBASEVERTEXPROC glad_glMultiDrawElementsBaseVertex = NULL;
4598 PFNGLMULTITEXCOORD1DPROC glad_glMultiTexCoord1d = NULL;
4599 PFNGLMULTITEXCOORD1DVPROC glad_glMultiTexCoord1dv = NULL;
4600 PFNGLMULTITEXCOORD1FPROC glad_glMultiTexCoord1f = NULL;
4601 PFNGLMULTITEXCOORD1FVPROC glad_glMultiTexCoord1fv = NULL;
4602 PFNGLMULTITEXCOORD1IPROC glad_glMultiTexCoord1i = NULL;
4603 PFNGLMULTITEXCOORD1IVPROC glad_glMultiTexCoord1iv = NULL;
4604 PFNGLMULTITEXCOORD1SPROC glad_glMultiTexCoord1s = NULL;
4605 PFNGLMULTITEXCOORD1SVPROC glad_glMultiTexCoord1sv = NULL;
4606 PFNGLMULTITEXCOORD2DPROC glad_glMultiTexCoord2d = NULL;
4607 PFNGLMULTITEXCOORD2DVPROC glad_glMultiTexCoord2dv = NULL;
4608 PFNGLMULTITEXCOORD2FPROC glad_glMultiTexCoord2f = NULL;
4609 PFNGLMULTITEXCOORD2FVPROC glad_glMultiTexCoord2fv = NULL;
```

```
4610 PFNGLMULTITEXCOORD2IPROC glad_glMultiTexCoord2i = NULL;
4611 PFNGLMULTITEXCOORD2IVPROC glad_glMultiTexCoord2iv = NULL;
4612 PFNGLMULTITEXCOORD2SPROC glad_glMultiTexCoord2s = NULL;
4613 PFNGLMULTITEXCOORD2SVPROC glad_glMultiTexCoord2sv = NULL;
4614 PFNGLMULTITEXCOORD3DPROC glad_glMultiTexCoord3d = NULL;
4615 PFNGLMULTITEXCOORD3DVPROC glad_glMultiTexCoord3dv = NULL;
4616 PFNGLMULTITEXCOORD3FPROC glad_glMultiTexCoord3f = NULL;
4617 PFNGLMULTITEXCOORD3FVPROC glad_glMultiTexCoord3fv = NULL;
4618 PFNGLMULTITEXCOORD3IPROC glad_glMultiTexCoord3i = NULL;
4619 PFNGLMULTITEXCOORD3IVPROC glad_glMultiTexCoord3iv = NULL;
4620 PFNGLMULTITEXCOORD3SPROC glad_glMultiTexCoord3s = NULL;
4621 PFNGLMULTITEXCOORD3SVPROC glad_glMultiTexCoord3sv = NULL;
4622 PFNGLMULTITEXCOORD4DPROC glad_glMultiTexCoord4d = NULL;
4623 PFNGLMULTITEXCOORD4DVPROC glad_glMultiTexCoord4dv = NULL;
4624 PFNGLMULTITEXCOORD4FPROC glad_glMultiTexCoord4f = NULL;
4625 PFNGLMULTITEXCOORD4FVPROC glad_glMultiTexCoord4fv = NULL;
4626 PFNGLMULTITEXCOORD4IPROC glad_glMultiTexCoord4i = NULL;
4627 PFNGLMULTITEXCOORD4IVPROC glad_glMultiTexCoord4iv = NULL;
4628 PFNGLMULTITEXCOORD4SPROC glad_glMultiTexCoord4s = NULL;
4629 PFNGLMULTITEXCOORD4SVPROC glad_glMultiTexCoord4sv = NULL;
4630 PFNGLMULTITEXCOORDP1UIPROC glad_glMultiTexCoordP1ui = NULL;
4631 PFNGLMULTITEXCOORDP1UIVPROC glad_glMultiTexCoordP1uiv = NULL;
4632 PFNGLMULTITEXCOORDP2UIPROC glad_glMultiTexCoordP2ui = NULL;
4633 PFNGLMULTITEXCOORDP2UIVPROC glad_glMultiTexCoordP2uiv = NULL;
4634 PFNGLMULTITEXCOORDP3UIPROC glad_glMultiTexCoordP3ui = NULL;
4635 PFNGLMULTITEXCOORDP3UIVPROC glad_glMultiTexCoordP3uiv = NULL;
4636 PFNGLMULTITEXCOORDP4UIPROC glad_glMultiTexCoordP4ui = NULL;
4637 PFNGLMULTITEXCOORDP4UIVPROC glad_glMultiTexCoordP4uiv = NULL;
4638 PFNGLNEWLISTPROC glad_glNewList = NULL;
4639 PFNGLNORMAL3BPROC glad_glNormal3b = NULL;
4640 PFNGLNORMAL3BVPROC glad_glNormal3bv = NULL;
4641 PFNGLNORMAL3DPROC glad_glNormal3d = NULL;
4642 PFNGLNORMAL3DVPROC glad_glNormal3dv = NULL;
4643 PFNGLNORMAL3FPROC glad_glNormal3f = NULL;
4644 PFNGLNORMAL3FVPROC glad_glNormal3fv = NULL;
4645 PFNGLNORMAL3IPROC glad_glNormal3i = NULL;
4646 PFNGLNORMAL3IVPROC glad_glNormal3iv = NULL;
4647 PFNGLNORMAL3SPROC glad_glNormal3s = NULL;
4648 PFNGLNORMAL3SVPROC glad_glNormal3sv = NULL;
4649 PFNGLNORMALP3UIPROC glad_glNormalP3ui = NULL;
4650 PFNGLNORMALP3UIVPROC glad_glNormalP3uiv = NULL;
4651 PFNGLNORMALPOINTERPROC glad_glNormalPointer = NULL;
4652 PFNGLOBJECTLABELPROC glad_glObjectLabel = NULL;
4653 PFNGLOBJECTPTRLABELPROC glad_glObjectPtrLabel = NULL;
4654 PFNGLORTHOPROC glad_glOrtho = NULL;
4655 PFNGLPASSTHROUGHPROC glad_glPassThrough = NULL;
4656 PFNGLPIXELMAPFVPROC glad_glPixelMapfv = NULL;
4657 PFNGLPIXELMAPUIVPROC glad_glPixelMapuiv = NULL;
4658 PFNGLPIXELMAPUSVPROC glad_glPixelMapusv = NULL;
4659 PFNGLPIXELSTOREFPROC glad_glPixelStoref = NULL;
4660 PFNGLPIXELSTOREIPROC glad_glPixelStorei = NULL;
4661 PFNGLPIXELTRANSFERFPROC glad_glPixelTransferf = NULL;
4662 PFNGLPIXELTRANSFERIPROC glad_glPixelTransferi = NULL;
4663 PFNGLPIXELZOOMPROC glad_glPixelZoom = NULL;
4664 PFNGLPOINTPARAMETERFPROC glad_glPointParameterf = NULL;
4665 PFNGLPOINTPARAMETERFVPROC glad_glPointParameterfv = NULL;
4666 PFNGLPOINTPARAMETERIPROC glad_glPointParameteri = NULL;
4667 PFNGLPOINTPARAMETERIVPROC glad_glPointParameteriv = NULL;
4668 PFNGLPOINTSIZEPROC glad_glPointSize = NULL;
4669 PFNGLPOLYGONMODEPROC glad_glPolygonMode = NULL;
4670 PFNGLPOLYGONOFFSETPROC glad_glPolygonOffset = NULL;
4671 PFNGLPOLYGONSTIPPLEPROC glad_glPolygonStipple = NULL;
4672 PFNGLPOPATTRIBPROC glad_glPopAttrib = NULL;
4673 PFNGLPOPCLIENTATTRIBPROC glad_glPopClientAttrib = NULL;
4674 PFNGLPOPDEBUGGROUPPROC glad_glPopDebugGroup = NULL;
4675 PFNGLPOPMATRIXPROC glad_glPopMatrix = NULL;
4676 PFNGLPOPNAMEPROC glad_glPopName = NULL;
4677 PFNGLPRIMITIVERESTARTINDEXPROC glad_glPrimitiveRestartIndex = NULL;
4678 PFNGLPRIORITIZETEXTURESPROC glad_glPrioritizeTextures = NULL;
4679 PFNGLPROVOKINGVERTEXPROC glad_glProvokingVertex = NULL;
4680 PFNGLPUSHATTRIBPROC glad_glPushAttrib = NULL;
4681 PFNGLPUSHCLIENTATTRIBPROC glad_glPushClientAttrib = NULL;
4682 PFNGLPUSHDEBUGGROUPPROC glad_glPushDebugGroup = NULL;
4683 PFNGLPUSHMATRIXPROC glad_glPushMatrix = NULL;
4684 PFNGLPUSHNAMEPROC glad_glPushName = NULL;
4685 PFNGLQUERYCOUNTERPROC glad_glQueryCounter = NULL;
4686 PFNGLRASTERPOS2DPROC glad_glRasterPos2d = NULL;
4687 PFNGLRASTERPOS2DVPROC glad_glRasterPos2dv = NULL;
4688 PFNGLRASTERPOS2FPROC glad_glRasterPos2f = NULL;
4689 PFNGLRASTERPOS2FVPROC glad_glRasterPos2fv = NULL;
4690 PFNGLRASTERPOS2IPROC glad_glRasterPos2i = NULL;
4691 PFNGLRASTERPOS2IVPROC glad_glRasterPos2iv = NULL;
4692 PFNGLRASTERPOS2SPROC glad_glRasterPos2s = NULL;
4693 PFNGLRASTERPOS2SVPROC glad_glRasterPos2sv = NULL;
4694 PFNGLRASTERPOS3DPROC glad_glRasterPos3d = NULL;
4695 PFNGLRASTERPOS3DVPROC glad_glRasterPos3dv = NULL;
4696 PFNGLRASTERPOS3FPROC glad_glRasterPos3f = NULL;
```

```
4697 PFNGLRASTERPOS3FVPROC glad_glRasterPos3fv = NULL;
4698 PFNGLRASTERPOS3IPROC glad_glRasterPos3i = NULL;
4699 PFNGLRASTERPOS3IVPROC glad_glRasterPos3iv = NULL;
4700 PFNGLRASTERPOS3SPROC glad_glRasterPos3s = NULL;
4701 PFNGLRASTERPOS3SVPROC glad_glRasterPos3sv = NULL;
4702 PFNGLRASTERPOS4DPROC glad_glRasterPos4d = NULL;
4703 PFNGLRASTERPOS4DVPROC glad_glRasterPos4dv = NULL;
4704 PFNGLRASTERPOS4FPROC glad_glRasterPos4f = NULL;
4705 PFNGLRASTERPOS4FVPROC glad_glRasterPos4fv = NULL;
4706 PFNGLRASTERPOS4IPROC glad_glRasterPos4i = NULL;
4707 PFNGLRASTERPOS4IVPROC glad_glRasterPos4iv = NULL;
4708 PFNGLRASTERPOS4SPROC glad_glRasterPos4s = NULL;
4709 PFNGLRASTERPOS4SVPROC glad_glRasterPos4sv = NULL;
4710 PFNGLREADBUFFERPROC glad_glReadBuffer = NULL;
4711 PFNGLREADPIXELSPROC glad_glReadPixels = NULL;
4712 PFNGLREADNPIXELSARBPROC glad_glReadnPixelsARB = NULL;
4713 PFNGLRECTDPROC glad_glRectd = NULL;
4714 PFNGLRECTDVPROC glad_glRectdv = NULL;
4715 PFNGLRECTFPROC glad_glRectf = NULL;
4716 PFNGLRECTFVPROC glad_glRectfv = NULL;
4717 PFNGLRECTIPROC glad_glRecti = NULL;
4718 PFNGLRECTIVPROC glad_glRectiv = NULL;
4719 PFNGLRECTSPROC glad_glRects = NULL;
4720 PFNGLRECTSVPROC glad_glRectsv = NULL;
4721 PFNGLRENDERMODEPROC glad_glRenderMode = NULL;
4722 PFNGLRENDERBUFFERSTORAGEPROC glad_glRenderbufferStorage = NULL;
4723 PFNGLRENDERBUFFERSTORAGEMULTISAMPLEPROC glad_glRenderbufferStorageMultisample = NULL;
4724 PFNGLROTATEDPROC glad_glRotated = NULL;
4725 PFNGLROTATEFPROC glad_glRotatef = NULL;
4726 PFNGLSAMPLECOVERAGEPROC glad_glSampleCoverage = NULL;
4727 PFNGLSAMPLECOVERAGEARBPROC glad_glSampleCoverageARB = NULL;
4728 PFNGLSAMPLEMASKIPROC glad_glSampleMaski = NULL;
4729 PFNGLSAMPLERPARAMETERIIVPROC glad_glSamplerParameterIiv = NULL;
4730 PFNGLSAMPLERPARAMETERIUIVPROC glad_glSamplerParameterIuiv = NULL;
4731 PFNGLSAMPLERPARAMETERFPROC glad_glSamplerParameterf = NULL;
4732 PFNGLSAMPLERPARAMETERFVPROC glad_glSamplerParameterfv = NULL;
4733 PFNGLSAMPLERPARAMETERIPROC glad_glSamplerParameteri = NULL;
4734 PFNGLSAMPLERPARAMETERIVPROC glad_glSamplerParameteriv = NULL;
4735 PFNGLSCALEDPROC glad_glScaled = NULL;
4736 PFNGLSCALEFPROC glad_glScalef = NULL;
4737 PFNGLSCISSORPROC glad_glScissor = NULL;
4738 PFNGLSECONDARYCOLOR3BPROC glad_glSecondaryColor3b = NULL;
4739 PFNGLSECONDARYCOLOR3BVPROC glad_glSecondaryColor3bv = NULL;
4740 PFNGLSECONDARYCOLOR3DPROC glad_glSecondaryColor3d = NULL;
4741 PFNGLSECONDARYCOLOR3DVPROC glad_glSecondaryColor3dv = NULL;
4742 PFNGLSECONDARYCOLOR3FPROC glad_glSecondaryColor3f = NULL;
4743 PFNGLSECONDARYCOLOR3FVPROC glad_glSecondaryColor3fv = NULL;
4744 PFNGLSECONDARYCOLOR3IPROC glad_glSecondaryColor3i = NULL;
4745 PFNGLSECONDARYCOLOR3IVPROC glad_glSecondaryColor3iv = NULL;
4746 PFNGLSECONDARYCOLOR3SPROC glad_glSecondaryColor3s = NULL;
4747 PFNGLSECONDARYCOLOR3SVPROC glad_glSecondaryColor3sv = NULL;
4748 PFNGLSECONDARYCOLOR3UBPROC glad_glSecondaryColor3ub = NULL;
4749 PFNGLSECONDARYCOLOR3UBVPROC glad_glSecondaryColor3ubv = NULL;
4750 PFNGLSECONDARYCOLOR3UIPROC glad_glSecondaryColor3ui = NULL;
4751 PFNGLSECONDARYCOLOR3UIVPROC glad_glSecondaryColor3uiv = NULL;
4752 PFNGLSECONDARYCOLOR3USPROC glad_glSecondaryColor3us = NULL;
4753 PFNGLSECONDARYCOLOR3USVPROC glad_glSecondaryColor3usv = NULL;
4754 PFNGLSECONDARYCOLORP3UIPROC glad_glSecondaryColorP3ui = NULL;
4755 PFNGLSECONDARYCOLORP3UIVPROC glad_glSecondaryColorP3uiv = NULL;
4756 PFNGLSECONDARYCOLORPOINTERPROC glad_glSecondaryColorPointer = NULL;
4757 PFNGLSELECTBUFFERPROC glad_glSelectBuffer = NULL;
4758 PFNGLSHADEMODELPROC glad_glShadeModel = NULL;
4759 PFNGLSHADERSOURCEPROC glad_glShaderSource = NULL;
4760 PFNGLSTENCILFUNCPROC glad_glStencilFunc = NULL;
4761 PFNGLSTENCILFUNCSEPARATEPROC glad_glStencilFuncSeparate = NULL;
4762 PFNGLSTENCILMASKPROC glad_glStencilMask = NULL;
4763 PFNGLSTENCILMASKSEPARATEPROC glad_glStencilMaskSeparate = NULL;
4764 PFNGLSTENCILOPPROC glad_glStencilOp = NULL;
4765 PFNGLSTENCILOPSEPARATEPROC glad_glStencilOpSeparate = NULL;
4766 PFNGLTEXBUFFERPROC glad_glTexBuffer = NULL;
4767 PFNGLTEXCOORD1DPROC glad_glTexCoord1d = NULL;
4768 PFNGLTEXCOORD1DVPROC glad_glTexCoord1dv = NULL;
4769 PFNGLTEXCOORD1FPROC glad_glTexCoord1f = NULL;
4770 PFNGLTEXCOORD1FVPROC glad_glTexCoord1fv = NULL;
4771 PFNGLTEXCOORD1IPROC glad_glTexCoord1i = NULL;
4772 PFNGLTEXCOORD1IVPROC glad_glTexCoord1iv = NULL;
4773 PFNGLTEXCOORD1SPROC glad_glTexCoord1s = NULL;
4774 PFNGLTEXCOORD1SVPROC glad_glTexCoord1sv = NULL;
4775 PFNGLTEXCOORD2DPROC glad_glTexCoord2d = NULL;
4776 PFNGLTEXCOORD2DVPROC glad_glTexCoord2dv = NULL;
4777 PFNGLTEXCOORD2FPROC glad_glTexCoord2f = NULL;
4778 PFNGLTEXCOORD2FVPROC glad_glTexCoord2fv = NULL;
4779 PFNGLTEXCOORD2IPROC glad_glTexCoord2i = NULL;
4780 PFNGLTEXCOORD2IVPROC glad_glTexCoord2iv = NULL;
4781 PFNGLTEXCOORD2SPROC glad_glTexCoord2s = NULL;
4782 PFNGLTEXCOORD2SVPROC glad_glTexCoord2sv = NULL;
4783 PFNGLTEXCOORD3DPROC glad_glTexCoord3d = NULL;
```

```
4784 PFNGLTEXCOORD3DVPROC glad_glTexCoord3dv = NULL;
4785 PFNGLTEXCOORD3FPROC glad_glTexCoord3f = NULL;
4786 PFNGLTEXCOORD3FVPROC glad_glTexCoord3fv = NULL;
4787 PFNGLTEXCOORD3IPROC glad_glTexCoord3i = NULL;
4788 PFNGLTEXCOORD3IVPROC glad_glTexCoord3iv = NULL;
4789 PFNGLTEXCOORD3SPROC glad_glTexCoord3s = NULL;
4790 PFNGLTEXCOORD3SVPROC glad_glTexCoord3sv = NULL;
4791 PFNGLTEXCOORD4DPROC glad_glTexCoord4d = NULL;
4792 PFNGLTEXCOORD4DVPROC glad_glTexCoord4dv = NULL;
4793 PFNGLTEXCOORD4FPROC glad_glTexCoord4f = NULL;
4794 PFNGLTEXCOORD4FVPROC glad_glTexCoord4fv = NULL;
4795 PFNGLTEXCOORD4IPROC glad_glTexCoord4i = NULL;
4796 PFNGLTEXCOORD4IVPROC glad_glTexCoord4iv = NULL;
4797 PFNGLTEXCOORD4SPROC glad_glTexCoord4s = NULL;
4798 PFNGLTEXCOORD4SVPROC glad_glTexCoord4sv = NULL;
4799 PFNGLTEXCOORDP1UIPROC glad_glTexCoordP1ui = NULL;
4800 PFNGLTEXCOORDP1UIVPROC glad_glTexCoordP1uiv = NULL;
4801 PFNGLTEXCOORDP2UIPROC glad_glTexCoordP2ui = NULL;
4802 PFNGLTEXCOORDP2UIVPROC glad_glTexCoordP2uiv = NULL;
4803 PFNGLTEXCOORDP3UIPROC glad_glTexCoordP3ui = NULL;
4804 PFNGLTEXCOORDP3UIVPROC glad_glTexCoordP3uiv = NULL;
4805 PFNGLTEXCOORDP4UIPROC glad_glTexCoordP4ui = NULL;
4806 PFNGLTEXCOORDP4UIVPROC glad_glTexCoordP4uiv = NULL;
4807 PFNGLTEXCOORDPOINTERPROC glad_glTexCoordPointer = NULL;
4808 PFNGLTEXENVFPROC glad_glTexEnvf = NULL;
4809 PFNGLTEXENVFVPROC glad_glTexEnvfv = NULL;
4810 PFNGLTEXENVIPROC glad_glTexEnvi = NULL;
4811 PFNGLTEXENVIVPROC glad_glTexEnviv = NULL;
4812 PFNGLTEXGENDPROC glad_glTexGend = NULL;
4813 PFNGLTEXGENDVPROC glad_glTexGendv = NULL;
4814 PFNGLTEXGENFPROC glad_glTexGenf = NULL;
4815 PFNGLTEXGENFVPROC glad_glTexGenfv = NULL;
4816 PFNGLTEXGENIPROC glad_glTexGeni = NULL;
4817 PFNGLTEXGENIVPROC glad_glTexGeniv = NULL;
4818 PFNGLTEXIMAGE1DPROC glad_glTexImage1D = NULL;
4819 PFNGLTEXIMAGE2DPROC glad_glTexImage2D = NULL;
4820 PFNGLTEXIMAGE2DMULTISAMPLEPROC glad_glTexImage2DMultisample = NULL;
4821 PFNGLTEXIMAGE3DPROC glad_glTexImage3D = NULL;
4822 PFNGLTEXIMAGE3DMULTISAMPLEPROC glad_glTexImage3DMultisample = NULL;
4823 PFNGLTEXPARAMETERIIVPROC glad_glTexParameterIiv = NULL;
4824 PFNGLTEXPARAMETERIUIVPROC glad_glTexParameterIuiv = NULL;
4825 PFNGLTEXPARAMETERFPROC glad_glTexParameterf = NULL;
4826 PFNGLTEXPARAMETERFVPROC glad_glTexParameterfv = NULL;
4827 PFNGLTEXPARAMETERIPROC glad_glTexParameteri = NULL;
4828 PFNGLTEXPARAMETERIVPROC glad_glTexParameteriv = NULL;
4829 PFNGLTEXSUBIMAGE1DPROC glad_glTexSubImage1D = NULL;
4830 PFNGLTEXSUBIMAGE2DPROC glad_glTexSubImage2D = NULL;
4831 PFNGLTEXSUBIMAGE3DPROC glad_glTexSubImage3D = NULL;
4832 PFNGLTRANSFORMFEEDBACKVARYINGSPROC glad_glTransformFeedbackVaryings = NULL;
4833 PFNGLTRANSLATEDPROC glad_glTranslated = NULL;
4834 PFNGLTRANSLATEFPROC glad_glTranslatef = NULL;
4835 PFNGLUNIFORM1FPROC glad_glUniform1f = NULL;
4836 PFNGLUNIFORM1FVPROC glad_glUniform1fv = NULL;
4837 PFNGLUNIFORM1IPROC glad_glUniform1i = NULL;
4838 PFNGLUNIFORM1IVPROC glad_glUniform1iv = NULL;
4839 PFNGLUNIFORM1UIPROC glad_glUniform1ui = NULL;
4840 PFNGLUNIFORM1UIVPROC glad_glUniform1uiv = NULL;
4841 PFNGLUNIFORM2FPROC glad_glUniform2f = NULL;
4842 PFNGLUNIFORM2FVPROC glad_glUniform2fv = NULL;
4843 PFNGLUNIFORM2IPROC glad_glUniform2i = NULL;
4844 PFNGLUNIFORM2IVPROC glad_glUniform2iv = NULL;
4845 PFNGLUNIFORM2UIPROC glad_glUniform2ui = NULL;
4846 PFNGLUNIFORM2UIVPROC glad_glUniform2uiv = NULL;
4847 PFNGLUNIFORM3FPROC glad_glUniform3f = NULL;
4848 PFNGLUNIFORM3FVPROC glad_glUniform3fv = NULL;
4849 PFNGLUNIFORM3IPROC glad_glUniform3i = NULL;
4850 PFNGLUNIFORM3IVPROC glad_glUniform3iv = NULL;
4851 PFNGLUNIFORM3UIPROC glad_glUniform3ui = NULL;
4852 PFNGLUNIFORM3UIVPROC glad_glUniform3uiv = NULL;
4853 PFNGLUNIFORM4FPROC glad_glUniform4f = NULL;
4854 PFNGLUNIFORM4FVPROC glad_glUniform4fv = NULL;
4855 PFNGLUNIFORM4IPROC glad_glUniform4i = NULL;
4856 PFNGLUNIFORM4IVPROC glad_glUniform4iv = NULL;
4857 PFNGLUNIFORM4UIPROC glad_glUniform4ui = NULL;
4858 PFNGLUNIFORM4UIVPROC glad_glUniform4uiv = NULL;
4859 PFNGLUNIFORMBLOCKBINDINGPROC glad_glUniformBlockBinding = NULL;
4860 PFNGLUNIFORMMATRIX2FVPROC glad_glUniformMatrix2fv = NULL;
4861 PFNGLUNIFORMMATRIX2X3FVPROC glad_glUniformMatrix2x3fv = NULL;
4862 PFNGLUNIFORMMATRIX2X4FVPROC glad_glUniformMatrix2x4fv = NULL;
4863 PFNGLUNIFORMMATRIX3FVPROC glad_glUniformMatrix3fv = NULL;
4864 PFNGLUNIFORMMATRIX3X2FVPROC glad_glUniformMatrix3x2fv = NULL;
4865 PFNGLUNIFORMMATRIX3X4FVPROC glad_glUniformMatrix3x4fv = NULL;
4866 PFNGLUNIFORMMATRIX4FVPROC glad_glUniformMatrix4fv = NULL;
4867 PFNGLUNIFORMMATRIX4X2FVPROC glad_glUniformMatrix4x2fv = NULL;
4868 PFNGLUNIFORMMATRIX4X3FVPROC glad_glUniformMatrix4x3fv = NULL;
4869 PFNGLUNMAPBUFFERPROC glad_glUnmapBuffer = NULL;
4870 PFNGLUSEPROGRAMPROC glad_glUseProgram = NULL;
```

```
4871 PFNGLVALIDATEPROGRAMPROC glad_glValidateProgram = NULL;
4872 PFNGLVERTEX2DPROC glad_glVertex2d = NULL;
4873 PFNGLVERTEX2DVPROC glad_glVertex2dv = NULL;
4874 PFNGLVERTEX2FPROC glad_glVertex2f = NULL;
4875 PFNGLVERTEX2FVPROC glad_glVertex2fv = NULL;
4876 PFNGLVERTEX2IPROC glad_glVertex2i = NULL;
4877 PFNGLVERTEX2IVPROC glad_glVertex2iv = NULL;
4878 PFNGLVERTEX2SPROC glad_glVertex2s = NULL;
4879 PFNGLVERTEX2SVPROC glad_glVertex2sv = NULL;
4880 PFNGLVERTEX3DPROC glad_glVertex3d = NULL;
4881 PFNGLVERTEX3DVPROC glad_glVertex3dv = NULL;
4882 PFNGLVERTEX3FPROC glad_glVertex3f = NULL;
4883 PFNGLVERTEX3FVPROC glad_glVertex3fv = NULL;
4884 PFNGLVERTEX3IPROC glad_glVertex3i = NULL;
4885 PFNGLVERTEX3IVPROC glad_glVertex3iv = NULL;
4886 PFNGLVERTEX3SPROC glad_glVertex3s = NULL;
4887 PFNGLVERTEX3SVPROC glad_glVertex3sv = NULL;
4888 PFNGLVERTEX4DPROC glad_glVertex4d = NULL;
4889 PFNGLVERTEX4DVPROC glad_glVertex4dv = NULL;
4890 PFNGLVERTEX4FPROC glad_glVertex4f = NULL;
4891 PFNGLVERTEX4FVPROC glad_glVertex4fv = NULL;
4892 PFNGLVERTEX4IPROC glad_glVertex4i = NULL;
4893 PFNGLVERTEX4IVPROC glad_glVertex4iv = NULL;
4894 PFNGLVERTEX4SPROC glad_glVertex4s = NULL;
4895 PFNGLVERTEX4SVPROC glad_glVertex4sv = NULL;
4896 PFNGLVERTEXATTRIB1DPROC glad_glVertexAttrib1d = NULL;
4897 PFNGLVERTEXATTRIB1DVPROC glad_glVertexAttrib1dv = NULL;
4898 PFNGLVERTEXATTRIB1FPROC glad_glVertexAttrib1f = NULL;
4899 PFNGLVERTEXATTRIB1FVPROC glad_glVertexAttrib1fv = NULL;
4900 PFNGLVERTEXATTRIB1SPROC glad_glVertexAttrib1s = NULL;
4901 PFNGLVERTEXATTRIB1SVPROC glad_glVertexAttrib1sv = NULL;
4902 PFNGLVERTEXATTRIB2DPROC glad_glVertexAttrib2d = NULL;
4903 PFNGLVERTEXATTRIB2DVPROC glad_glVertexAttrib2dv = NULL;
4904 PFNGLVERTEXATTRIB2FPROC glad_glVertexAttrib2f = NULL;
4905 PFNGLVERTEXATTRIB2FVPROC glad_glVertexAttrib2fv = NULL;
4906 PFNGLVERTEXATTRIB2SPROC glad_glVertexAttrib2s = NULL;
4907 PFNGLVERTEXATTRIB2SVPROC glad_glVertexAttrib2sv = NULL;
4908 PFNGLVERTEXATTRIB3DPROC glad_glVertexAttrib3d = NULL;
4909 PFNGLVERTEXATTRIB3DVPROC glad_glVertexAttrib3dv = NULL;
4910 PFNGLVERTEXATTRIB3FPROC glad_glVertexAttrib3f = NULL;
4911 PFNGLVERTEXATTRIB3FVPROC glad_glVertexAttrib3fv = NULL;
4912 PFNGLVERTEXATTRIB3SPROC glad_glVertexAttrib3s = NULL;
4913 PFNGLVERTEXATTRIB3SVPROC glad_glVertexAttrib3sv = NULL;
4914 PFNGLVERTEXATTRIB4NBVPROC glad_glVertexAttrib4Nbv = NULL;
4915 PFNGLVERTEXATTRIB4NIVPROC glad_glVertexAttrib4Niv = NULL;
4916 PFNGLVERTEXATTRIB4NSVPROC glad_glVertexAttrib4Nsv = NULL;
4917 PFNGLVERTEXATTRIB4NUBPROC glad_glVertexAttrib4Nub = NULL;
4918 PFNGLVERTEXATTRIB4NUBVPROC glad_glVertexAttrib4Nubv = NULL;
4919 PFNGLVERTEXATTRIB4NUIVPROC glad_glVertexAttrib4Nuiv = NULL;
4920 PFNGLVERTEXATTRIB4NUSVPROC glad_glVertexAttrib4Nusv = NULL;
4921 PFNGLVERTEXATTRIB4BVPROC glad_glVertexAttrib4bv = NULL;
4922 PFNGLVERTEXATTRIB4DPROC glad_glVertexAttrib4d = NULL;
4923 PFNGLVERTEXATTRIB4DVPROC glad_glVertexAttrib4dv = NULL;
4924 PFNGLVERTEXATTRIB4FPROC glad_glVertexAttrib4f = NULL;
4925 PFNGLVERTEXATTRIB4FVPROC glad_glVertexAttrib4fv = NULL;
4926 PFNGLVERTEXATTRIB4IVPROC glad_glVertexAttrib4iv = NULL;
4927 PFNGLVERTEXATTRIB4SPROC glad_glVertexAttrib4s = NULL;
4928 PFNGLVERTEXATTRIB4SVPROC glad_glVertexAttrib4sv = NULL;
4929 PFNGLVERTEXATTRIB4UBVPROC glad_glVertexAttrib4ubv = NULL;
4930 PFNGLVERTEXATTRIB4UIVPROC glad_glVertexAttrib4uiv = NULL;
4931 PFNGLVERTEXATTRIB4USVPROC glad_glVertexAttrib4usv = NULL;
4932 PFNGLVERTEXATTRIBDIVISORPROC glad_glVertexAttribDivisor = NULL;
4933 PFNGLVERTEXATTRIBI1IPROC glad_glVertexAttribI1i = NULL;
4934 PFNGLVERTEXATTRIBI1IVPROC glad_glVertexAttribI1iv = NULL;
4935 PFNGLVERTEXATTRIBI1UIPROC glad_glVertexAttribI1ui = NULL;
4936 PFNGLVERTEXATTRIBI1UIVPROC glad_glVertexAttribI1uiv = NULL;
4937 PFNGLVERTEXATTRIBI2IPROC glad_glVertexAttribI2i = NULL;
4938 PFNGLVERTEXATTRIBI2IVPROC glad_glVertexAttribI2iv = NULL;
4939 PFNGLVERTEXATTRIBI2UIPROC glad_glVertexAttribI2ui = NULL;
4940 PFNGLVERTEXATTRIBI2UIVPROC glad_glVertexAttribI2uiv = NULL;
4941 PFNGLVERTEXATTRIBI3IPROC glad_glVertexAttribI3i = NULL;
4942 PFNGLVERTEXATTRIBI3IVPROC glad_glVertexAttribI3iv = NULL;
4943 PFNGLVERTEXATTRIBI3UIPROC glad_glVertexAttribI3ui = NULL;
4944 PFNGLVERTEXATTRIBI3UIVPROC glad_glVertexAttribI3uiv = NULL;
4945 PFNGLVERTEXATTRIBI4BVPROC glad_glVertexAttribI4bv = NULL;
4946 PFNGLVERTEXATTRIBI4IPROC glad_glVertexAttribI4i = NULL;
4947 PFNGLVERTEXATTRIBI4IVPROC glad_glVertexAttribI4iv = NULL;
4948 PFNGLVERTEXATTRIBI4SVPROC glad_glVertexAttribI4sv = NULL;
4949 PFNGLVERTEXATTRIBI4UBVPROC glad_glVertexAttribI4ubv = NULL;
4950 PFNGLVERTEXATTRIBI4UIPROC glad_glVertexAttribI4ui = NULL;
4951 PFNGLVERTEXATTRIBI4UIVPROC glad_glVertexAttribI4uiv = NULL;
4952 PFNGLVERTEXATTRIBI4USVPROC glad_glVertexAttribI4usv = NULL;
4953 PFNGLVERTEXATTRIBIPOINTERPROC glad_glVertexAttribIPointer = NULL;
4954 PFNGLVERTEXATTRIBP1UIPROC glad_glVertexAttribP1ui = NULL;
4955 PFNGLVERTEXATTRIBP1UIVPROC glad_glVertexAttribP1uiv = NULL;
4956 PFNGLVERTEXATTRIBP2UIPROC glad_glVertexAttribP2ui = NULL;
4957 PFNGLVERTEXATTRIBP2UIVPROC glad_glVertexAttribP2uiv = NULL;
```

```
4958 PFNGLVERTEXATTRIBP3UIPROC glad_glVertexAttribP3ui = NULL;
4959 PFNGLVERTEXATTRIBP3UIVPROC glad_glVertexAttribP3uiv = NULL;
4960 PFNGLVERTEXATTRIBP4UIPROC glad_glVertexAttribP4ui = NULL;
4961 PFNGLVERTEXATTRIBP4UIVPROC glad_glVertexAttribP4uiv = NULL;
4962 PFNGLVERTEXATTRIBPOINTERPROC glad_glVertexAttribPointer = NULL;
4963 PFNGLVERTEXP2UIPROC glad_glVertexP2ui = NULL;
4964 PFNGLVERTEXP2UIVPROC glad_glVertexP2uiv = NULL;
4965 PFNGLVERTEXP3UIPROC glad_glVertexP3ui = NULL;
4966 PFNGLVERTEXP3UIVPROC glad_glVertexP3uiv = NULL;
4967 PFNGLVERTEXP4UIPROC glad_glVertexP4ui = NULL;
4968 PFNGLVERTEXP4UIVPROC glad_glVertexP4uiv = NULL;
4969 PFNGLVERTEXPOINTERPROC glad_glVertexPointer = NULL;
4970 PFNGLVIEWPORTPROC glad_glViewport = NULL;
4971 PFNGLWAITSYNCPROC glad_glWaitSync = NULL;
4972 PFNGLWINDOWPOS2DPROC glad_glWindowPos2d = NULL;
4973 PFNGLWINDOWPOS2DVPROC glad_glWindowPos2dv = NULL;
4974 PFNGLWINDOWPOS2FPROC glad_glWindowPos2f = NULL;
4975 PFNGLWINDOWPOS2FVPROC glad_glWindowPos2fv = NULL;
4976 PFNGLWINDOWPOS2IPROC glad_glWindowPos2i = NULL;
4977 PFNGLWINDOWPOS2IVPROC glad_glWindowPos2iv = NULL;
4978 PFNGLWINDOWPOS2SPROC glad_glWindowPos2s = NULL;
4979 PFNGLWINDOWPOS2SVPROC glad_glWindowPos2sv = NULL;
4980 PFNGLWINDOWPOS3DPROC glad_glWindowPos3d = NULL;
4981 PFNGLWINDOWPOS3DVPROC glad_glWindowPos3dv = NULL;
4982 PFNGLWINDOWPOS3FPROC glad_glWindowPos3f = NULL;
4983 PFNGLWINDOWPOS3FVPROC glad_glWindowPos3fv = NULL;
4984 PFNGLWINDOWPOS3IPROC glad_glWindowPos3i = NULL;
4985 PFNGLWINDOWPOS3IVPROC glad_glWindowPos3iv = NULL;
4986 PFNGLWINDOWPOS3SPROC glad_glWindowPos3s = NULL;
4987 PFNGLWINDOWPOS3SVPROC glad_glWindowPos3sv = NULL;
4988
4989
4990 static void glad_gl_load_GL_VERSION_1_0( GLADuserptrloadfunc load, void* userptr) {
4991     if(!GLAD_GL_VERSION_1_0) return;
4992     glad_glAccum = (PFNGLACCUMPROC) load(userptr, "glAccum");
4993     glad_glAlphaFunc = (PFNGLALPHAFUNCPROC) load(userptr, "glAlphaFunc");
4994     glad_glBegin = (PFNGLBEGINPROC) load(userptr, "glBegin");
4995     glad_glBitmap = (PFNGLBITMAPPROC) load(userptr, "glBitmap");
4996     glad_glBlendFunc = (PFNGLBLENDFUNCPROC) load(userptr, "glBlendFunc");
4997     glad_glCallList = (PFNGLCALLLISTPROC) load(userptr, "glCallList");
4998     glad_glCallLists = (PFNGLCALLLISTSPROC) load(userptr, "glCallLists");
4999     glad_glClear = (PFNGLCLEARPROC) load(userptr, "glClear");
5000     glad_glClearAccum = (PFNGLCLEARACCUMPROC) load(userptr, "glClearAccum");
5001     glad_glClearColor = (PFNGLCLEARCOLORPROC) load(userptr, "glClearColor");
5002     glad_glClearDepth = (PFNGLCLEARDEPTHPROC) load(userptr, "glClearDepth");
5003     glad_glClearIndex = (PFNGLCLEARINDEXPROC) load(userptr, "glClearIndex");
5004     glad_glClearStencil = (PFNGLCLEARSTENCILPROC) load(userptr, "glClearStencil");
5005     glad_glClipPlane = (PFNGLCLIPPLANEPROC) load(userptr, "glClipPlane");
5006     glad_glColor3b = (PFNGLCOLOR3BPROC) load(userptr, "glColor3b");
5007     glad_glColor3bv = (PFNGLCOLOR3BVPROC) load(userptr, "glColor3bv");
5008     glad_glColor3d = (PFNGLCOLOR3DPROC) load(userptr, "glColor3d");
5009     glad_glColor3dv = (PFNGLCOLOR3DVPROC) load(userptr, "glColor3dv");
5010     glad_glColor3f = (PFNGLCOLOR3FPROC) load(userptr, "glColor3f");
5011     glad_glColor3fv = (PFNGLCOLOR3FVPROC) load(userptr, "glColor3fv");
5012     glad_glColor3i = (PFNGLCOLOR3IPROC) load(userptr, "glColor3i");
5013     glad_glColor3iv = (PFNGLCOLOR3IVPROC) load(userptr, "glColor3iv");
5014     glad_glColor3s = (PFNGLCOLOR3SPROC) load(userptr, "glColor3s");
5015     glad_glColor3sv = (PFNGLCOLOR3SVPROC) load(userptr, "glColor3sv");
5016     glad_glColor3ub = (PFNGLCOLOR3UBPROC) load(userptr, "glColor3ub");
5017     glad_glColor3ubv = (PFNGLCOLOR3UBVPROC) load(userptr, "glColor3ubv");
5018     glad_glColor3ui = (PFNGLCOLOR3UIPROC) load(userptr, "glColor3ui");
5019     glad_glColor3uiv = (PFNGLCOLOR3UIVPROC) load(userptr, "glColor3uiv");
5020     glad_glColor3us = (PFNGLCOLOR3USPROC) load(userptr, "glColor3us");
5021     glad_glColor3usv = (PFNGLCOLOR3USVPROC) load(userptr, "glColor3usv");
5022     glad_glColor4b = (PFNGLCOLOR4BPROC) load(userptr, "glColor4b");
5023     glad_glColor4bv = (PFNGLCOLOR4BVPROC) load(userptr, "glColor4bv");
5024     glad_glColor4d = (PFNGLCOLOR4DPROC) load(userptr, "glColor4d");
5025     glad_glColor4dv = (PFNGLCOLOR4DVPROC) load(userptr, "glColor4dv");
5026     glad_glColor4f = (PFNGLCOLOR4FPROC) load(userptr, "glColor4f");
5027     glad_glColor4fv = (PFNGLCOLOR4FVPROC) load(userptr, "glColor4fv");
5028     glad_glColor4i = (PFNGLCOLOR4IPROC) load(userptr, "glColor4i");
5029     glad_glColor4iv = (PFNGLCOLOR4IVPROC) load(userptr, "glColor4iv");
5030     glad_glColor4s = (PFNGLCOLOR4SPROC) load(userptr, "glColor4s");
5031     glad_glColor4sv = (PFNGLCOLOR4SVPROC) load(userptr, "glColor4sv");
5032     glad_glColor4ub = (PFNGLCOLOR4UBPROC) load(userptr, "glColor4ub");
5033     glad_glColor4ubv = (PFNGLCOLOR4UBVPROC) load(userptr, "glColor4ubv");
5034     glad_glColor4ui = (PFNGLCOLOR4UIPROC) load(userptr, "glColor4ui");
5035     glad_glColor4uiv = (PFNGLCOLOR4UIVPROC) load(userptr, "glColor4uiv");
5036     glad_glColor4us = (PFNGLCOLOR4USPROC) load(userptr, "glColor4us");
5037     glad_glColor4usv = (PFNGLCOLOR4USVPROC) load(userptr, "glColor4usv");
5038     glad_glColorMask = (PFNGLCOLORMASKPROC) load(userptr, "glColorMask");
5039     glad_glColorMaterial = (PFNGLCOLORMATERIALPROC) load(userptr, "glColorMaterial");
5040     glad_glCopyPixels = (PFNGLCOPYPIXELSPROC) load(userptr, "glCopyPixels");
5041     glad_glCullFace = (PFNGLCULLFACEPROC) load(userptr, "glCullFace");
5042     glad_glDeleteLists = (PFNGLDELETELISTSPROC) load(userptr, "glDeleteLists");
5043     glad_glDepthFunc = (PFNGLDEPTHFUNCPROC) load(userptr, "glDepthFunc");
5044     glad_glDepthMask = (PFNGLDEPTHMASKPROC) load(userptr, "glDepthMask");
```

```
5045     glad_glDepthRange = (PFNGLDEPTHRANGEPROC) load(userptr, "glDepthRange");
5046     glad_glDisable = (PFNGLDISABLEPROC) load(userptr, "glDisable");
5047     glad_glDrawBuffer = (PFNGLDRAWBUFFERPROC) load(userptr, "glDrawBuffer");
5048     glad_glDrawPixels = (PFNGLDRAWPIXELSPROC) load(userptr, "glDrawPixels");
5049     glad_glEdgeFlag = (PFNGLEDGEFLAGPROC) load(userptr, "glEdgeFlag");
5050     glad_glEdgeFlagv = (PFNGLEDGEFLAGVPROC) load(userptr, "glEdgeFlagv");
5051     glad_glEnable = (PFNGLENABLEPROC) load(userptr, "glEnable");
5052     glad_glEnd = (PFNGLENDPROC) load(userptr, "glEnd");
5053     glad_glEndList = (PFNGLENDLISTPROC) load(userptr, "glEndList");
5054     glad_glEvalCoord1d = (PFNGLEVALCOORD1DPROC) load(userptr, "glEvalCoord1d");
5055     glad_glEvalCoord1dv = (PFNGLEVALCOORD1DVPROC) load(userptr, "glEvalCoord1dv");
5056     glad_glEvalCoord1f = (PFNGLEVALCOORD1FPROC) load(userptr, "glEvalCoord1f");
5057     glad_glEvalCoord1fv = (PFNGLEVALCOORD1FVPROC) load(userptr, "glEvalCoord1fv");
5058     glad_glEvalCoord2d = (PFNGLEVALCOORD2DPROC) load(userptr, "glEvalCoord2d");
5059     glad_glEvalCoord2dv = (PFNGLEVALCOORD2DVPROC) load(userptr, "glEvalCoord2dv");
5060     glad_glEvalCoord2f = (PFNGLEVALCOORD2FPROC) load(userptr, "glEvalCoord2f");
5061     glad_glEvalCoord2fv = (PFNGLEVALCOORD2FVPROC) load(userptr, "glEvalCoord2fv");
5062     glad_glEvalMesh1 = (PFNGLEVALMESH1PROC) load(userptr, "glEvalMesh1");
5063     glad_glEvalMesh2 = (PFNGLEVALMESH2PROC) load(userptr, "glEvalMesh2");
5064     glad_glEvalPoint1 = (PFNGLEVALPOINT1PROC) load(userptr, "glEvalPoint1");
5065     glad_glEvalPoint2 = (PFNGLEVALPOINT2PROC) load(userptr, "glEvalPoint2");
5066     glad_glFeedbackBuffer = (PFNGLFEEDBACKBUFFERPROC) load(userptr, "glFeedbackBuffer");
5067     glad_glFinish = (PFNGLFINISHPROC) load(userptr, "glFinish");
5068     glad_glFlush = (PFNGLFLUSHPROC) load(userptr, "glFlush");
5069     glad_glFogf = (PFNGLFOGFPROC) load(userptr, "glFogf");
5070     glad_glFogfv = (PFNGLFOGFVPROC) load(userptr, "glFogfv");
5071     glad_glFogi = (PFNGLFOGIPROC) load(userptr, "glFogi");
5072     glad_glFogiv = (PFNGLFOGIVPROC) load(userptr, "glFogiv");
5073     glad_glFrontFace = (PFNGLFRONTFACEPROC) load(userptr, "glFrontFace");
5074     glad_glFrustum = (PFNGLFRUSTUMPROC) load(userptr, "glFrustum");
5075     glad_glGenLists = (PFNGLGENLISTSPROC) load(userptr, "glGenLists");
5076     glad_glGetBooleanv = (PFNGLGETBOOLEANVPROC) load(userptr, "glGetBooleanv");
5077     glad_glGetClipPlane = (PFNGLGETCLIPPLANEPROC) load(userptr, "glGetClipPlane");
5078     glad_glGetDoublev = (PFNGLGETDOUBLEVPROC) load(userptr, "glGetDoublev");
5079     glad_glGetError = (PFNGLGETERRORPROC) load(userptr, "glGetError");
5080     glad_glGetFloatv = (PFNGLGETFLOATVPROC) load(userptr, "glGetFloatv");
5081     glad_glGetIntegerv = (PFNGLGETINTEGERVPROC) load(userptr, "glGetIntegerv");
5082     glad_glGetLightfv = (PFNGLGETLIGHTFVPROC) load(userptr, "glGetLightfv");
5083     glad_glGetLightiv = (PFNGLGETLIGHTIVPROC) load(userptr, "glGetLightiv");
5084     glad_glGetMapdv = (PFNGLGETMAPDVPROC) load(userptr, "glGetMapdv");
5085     glad_glGetMapfv = (PFNGLGETMAPFVPROC) load(userptr, "glGetMapfv");
5086     glad_glGetMapiv = (PFNGLGETMAPIVPROC) load(userptr, "glGetMapiv");
5087     glad_glGetMaterialfv = (PFNGLGETMATERIALFVPROC) load(userptr, "glGetMaterialfv");
5088     glad_glGetMaterialiv = (PFNGLGETMATERIALIVPROC) load(userptr, "glGetMaterialiv");
5089     glad_glGetPixelMapfv = (PFNGLGETPIXELMAPFVPROC) load(userptr, "glGetPixelMapfv");
5090     glad_glGetPixelMapuiv = (PFNGLGETPIXELMAPUIVPROC) load(userptr, "glGetPixelMapuiv");
5091     glad_glGetPixelMapusv = (PFNGLGETPIXELMAPUSVPROC) load(userptr, "glGetPixelMapusv");
5092     glad_glGetPolygonStipple = (PFNGLGETPOLYGONSTIPPLEPROC) load(userptr, "glGetPolygonStipple");
5093     glad_glGetString = (PFNGLGETSTRINGPROC) load(userptr, "glGetString");
5094     glad_glGetTexEnvfv = (PFNGLGETTEXENVFVPROC) load(userptr, "glGetTexEnvfv");
5095     glad_glGetTexEnviv = (PFNGLGETTEXENVIVPROC) load(userptr, "glGetTexEnviv");
5096     glad_glGetTexGendv = (PFNGLGETTEXGENDVPROC) load(userptr, "glGetTexGendv");
5097     glad_glGetTexGenfv = (PFNGLGETTEXGENFVPROC) load(userptr, "glGetTexGenfv");
5098     glad_glGetTexGeniv = (PFNGLGETTEXGENIVPROC) load(userptr, "glGetTexGeniv");
5099     glad_glGetTexImage = (PFNGLGETTEXIMAGEPROC) load(userptr, "glGetTexImage");
5100     glad_glGetTexLevelParameterfv = (PFNGLGETTEXLEVELPARAMETERFVPROC) load(userptr,
    "glGetTexLevelParameterfv");
5101     glad_glGetTexLevelParameteriv = (PFNGLGETTEXLEVELPARAMETERIVPROC) load(userptr,
    "glGetTexLevelParameteriv");
5102     glad_glGetTexParameterfv = (PFNGLGETTEXPARAMETERFVPROC) load(userptr, "glGetTexParameterfv");
5103     glad_glGetTexParameteriv = (PFNGLGETTEXPARAMETERIVPROC) load(userptr, "glGetTexParameteriv");
5104     glad_glHint = (PFNGLHINTPROC) load(userptr, "glHint");
5105     glad_glIndexMask = (PFNGLINDEXMASKPROC) load(userptr, "glIndexMask");
5106     glad_glIndexd = (PFNGLINDEXDPROC) load(userptr, "glIndexd");
5107     glad_glIndexdv = (PFNGLINDEXDVPROC) load(userptr, "glIndexdv");
5108     glad_glIndexf = (PFNGLINDEXFPROC) load(userptr, "glIndexf");
5109     glad_glIndexfv = (PFNGLINDEXFVPROC) load(userptr, "glIndexfv");
5110     glad_glIndexi = (PFNGLINDEXIPROC) load(userptr, "glIndexi");
5111     glad_glIndexiv = (PFNGLINDEXIVPROC) load(userptr, "glIndexiv");
5112     glad_glIndexs = (PFNGLINDEXSPROC) load(userptr, "glIndexs");
5113     glad_glIndexsv = (PFNGLINDEXSVPROC) load(userptr, "glIndexsv");
5114     glad_glInitNames = (PFNGLINITNAMESPROC) load(userptr, "glInitNames");
5115     glad_glIsEnabled = (PFNGLISENABLEDPROC) load(userptr, "glIsEnabled");
5116     glad_glIsList = (PFNGLISLISTPROC) load(userptr, "glIsList");
5117     glad_glLightModelf = (PFNGLLIGHTMODELFPROC) load(userptr, "glLightModelf");
5118     glad_glLightModelfv = (PFNGLLIGHTMODELFVPROC) load(userptr, "glLightModelfv");
5119     glad_glLightModeli = (PFNGLLIGHTMODELIPROC) load(userptr, "glLightModeli");
5120     glad_glLightModeliv = (PFNGLLIGHTMODELIVPROC) load(userptr, "glLightModeliv");
5121     glad_glLightf = (PFNGLLIGHTFPROC) load(userptr, "glLightf");
5122     glad_glLightfv = (PFNGLLIGHTFVPROC) load(userptr, "glLightfv");
5123     glad_glLighti = (PFNGLLIGHTIPROC) load(userptr, "glLighti");
5124     glad_glLightiv = (PFNGLLIGHTIVPROC) load(userptr, "glLightiv");
5125     glad_glLineStipple = (PFNGLLINESTIPPLEPROC) load(userptr, "glLineStipple");
5126     glad_glLineWidth = (PFNGLLINEWIDTHPROC) load(userptr, "glLineWidth");
5127     glad_glListBase = (PFNGLLISTBASEPROC) load(userptr, "glListBase");
5128     glad_glLoadIdentity = (PFNGLLOADIDENTITYPROC) load(userptr, "glLoadIdentity");
5129     glad_glLoadMatrixd = (PFNGLLOADMATRIXDPROC) load(userptr, "glLoadMatrixd");
```

```
5130     glad_glLoadMatrixf = (PFNGLLOADMATRIXFPROC) load(userptr, "glLoadMatrixf");
5131     glad_glLoadName = (PFNGLLOADNAMEPROC) load(userptr, "glLoadName");
5132     glad_glLogicOp = (PFNGLLOGICOPPROC) load(userptr, "glLogicOp");
5133     glad_glMap1d = (PFNGLMAP1DPROC) load(userptr, "glMap1d");
5134     glad_glMap1f = (PFNGLMAP1FPROC) load(userptr, "glMap1f");
5135     glad_glMap2d = (PFNGLMAP2DPROC) load(userptr, "glMap2d");
5136     glad_glMap2f = (PFNGLMAP2FPROC) load(userptr, "glMap2f");
5137     glad_glMapGrid1d = (PFNGLMAPGRID1DPROC) load(userptr, "glMapGrid1d");
5138     glad_glMapGrid1f = (PFNGLMAPGRID1FPROC) load(userptr, "glMapGrid1f");
5139     glad_glMapGrid2d = (PFNGLMAPGRID2DPROC) load(userptr, "glMapGrid2d");
5140     glad_glMapGrid2f = (PFNGLMAPGRID2FPROC) load(userptr, "glMapGrid2f");
5141     glad_glMaterialf = (PFNGLMATERIALFPROC) load(userptr, "glMaterialf");
5142     glad_glMaterialfv = (PFNGLMATERIALFVPROC) load(userptr, "glMaterialfv");
5143     glad_glMateriali = (PFNGLMATERIALIPROC) load(userptr, "glMateriali");
5144     glad_glMaterialiv = (PFNGLMATERIALIVPROC) load(userptr, "glMaterialiv");
5145     glad_glMatrixMode = (PFNGLMATRIXMODEPROC) load(userptr, "glMatrixMode");
5146     glad_glMultMatrixd = (PFNGLMULTMATRIXDPROC) load(userptr, "glMultMatrixd");
5147     glad_glMultMatrixf = (PFNGLMULTMATRIXFPROC) load(userptr, "glMultMatrixf");
5148     glad_glNewList = (PFNGLNEWLISTPROC) load(userptr, "glNewList");
5149     glad_glNormal3b = (PFNGLNORMAL3BPROC) load(userptr, "glNormal3b");
5150     glad_glNormal3bv = (PFNGLNORMAL3BVPROC) load(userptr, "glNormal3bv");
5151     glad_glNormal3d = (PFNGLNORMAL3DPROC) load(userptr, "glNormal3d");
5152     glad_glNormal3dv = (PFNGLNORMAL3DVPROC) load(userptr, "glNormal3dv");
5153     glad_glNormal3f = (PFNGLNORMAL3FPROC) load(userptr, "glNormal3f");
5154     glad_glNormal3fv = (PFNGLNORMAL3FVPROC) load(userptr, "glNormal3fv");
5155     glad_glNormal3i = (PFNGLNORMAL3IPROC) load(userptr, "glNormal3i");
5156     glad_glNormal3iv = (PFNGLNORMAL3IVPROC) load(userptr, "glNormal3iv");
5157     glad_glNormal3s = (PFNGLNORMAL3SPROC) load(userptr, "glNormal3s");
5158     glad_glNormal3sv = (PFNGLNORMAL3SVPROC) load(userptr, "glNormal3sv");
5159     glad_glOrtho = (PFNGLORTHOPROC) load(userptr, "glOrtho");
5160     glad_glPassThrough = (PFNGLPASSTHROUGHPROC) load(userptr, "glPassThrough");
5161     glad_glPixelMapfv = (PFNGLPIXELMAPFVPROC) load(userptr, "glPixelMapfv");
5162     glad_glPixelMapuiv = (PFNGLPIXELMAPUIVPROC) load(userptr, "glPixelMapuiv");
5163     glad_glPixelMapusv = (PFNGLPIXELMAPUSVPROC) load(userptr, "glPixelMapusv");
5164     glad_glPixelStoref = (PFNGLPIXELSTOREFPROC) load(userptr, "glPixelStoref");
5165     glad_glPixelStorei = (PFNGLPIXELSTOREIPROC) load(userptr, "glPixelStorei");
5166     glad_glPixelTransferf = (PFNGLPIXELTRANSFERFPROC) load(userptr, "glPixelTransferf");
5167     glad_glPixelTransferi = (PFNGLPIXELTRANSFERIPROC) load(userptr, "glPixelTransferi");
5168     glad_glPixelZoom = (PFNGLPIXELZOOMPROC) load(userptr, "glPixelZoom");
5169     glad_glPointSize = (PFNGLPOINTSIZEPROC) load(userptr, "glPointSize");
5170     glad_glPolygonMode = (PFNGLPOLYGONMODEPROC) load(userptr, "glPolygonMode");
5171     glad_glPolygonStipple = (PFNGLPOLYGONSTIPPLEPROC) load(userptr, "glPolygonStipple");
5172     glad_glPopAttrib = (PFNGLPOPATTRIBPROC) load(userptr, "glPopAttrib");
5173     glad_glPopMatrix = (PFNGLPOPMATRIXPROC) load(userptr, "glPopMatrix");
5174     glad_glPopName = (PFNGLPOPNAMEPROC) load(userptr, "glPopName");
5175     glad_glPushAttrib = (PFNGLPUSHATTRIBPROC) load(userptr, "glPushAttrib");
5176     glad_glPushMatrix = (PFNGLPUSHMATRIXPROC) load(userptr, "glPushMatrix");
5177     glad_glPushName = (PFNGLPUSHNAMEPROC) load(userptr, "glPushName");
5178     glad_glRasterPos2d = (PFNGLRASTERPOS2DPROC) load(userptr, "glRasterPos2d");
5179     glad_glRasterPos2dv = (PFNGLRASTERPOS2DVPROC) load(userptr, "glRasterPos2dv");
5180     glad_glRasterPos2f = (PFNGLRASTERPOS2FPROC) load(userptr, "glRasterPos2f");
5181     glad_glRasterPos2fv = (PFNGLRASTERPOS2FVPROC) load(userptr, "glRasterPos2fv");
5182     glad_glRasterPos2i = (PFNGLRASTERPOS2IPROC) load(userptr, "glRasterPos2i");
5183     glad_glRasterPos2iv = (PFNGLRASTERPOS2IVPROC) load(userptr, "glRasterPos2iv");
5184     glad_glRasterPos2s = (PFNGLRASTERPOS2SPROC) load(userptr, "glRasterPos2s");
5185     glad_glRasterPos2sv = (PFNGLRASTERPOS2SVPROC) load(userptr, "glRasterPos2sv");
5186     glad_glRasterPos3d = (PFNGLRASTERPOS3DPROC) load(userptr, "glRasterPos3d");
5187     glad_glRasterPos3dv = (PFNGLRASTERPOS3DVPROC) load(userptr, "glRasterPos3dv");
5188     glad_glRasterPos3f = (PFNGLRASTERPOS3FPROC) load(userptr, "glRasterPos3f");
5189     glad_glRasterPos3fv = (PFNGLRASTERPOS3FVPROC) load(userptr, "glRasterPos3fv");
5190     glad_glRasterPos3i = (PFNGLRASTERPOS3IPROC) load(userptr, "glRasterPos3i");
5191     glad_glRasterPos3iv = (PFNGLRASTERPOS3IVPROC) load(userptr, "glRasterPos3iv");
5192     glad_glRasterPos3s = (PFNGLRASTERPOS3SPROC) load(userptr, "glRasterPos3s");
5193     glad_glRasterPos3sv = (PFNGLRASTERPOS3SVPROC) load(userptr, "glRasterPos3sv");
5194     glad_glRasterPos4d = (PFNGLRASTERPOS4DPROC) load(userptr, "glRasterPos4d");
5195     glad_glRasterPos4dv = (PFNGLRASTERPOS4DVPROC) load(userptr, "glRasterPos4dv");
5196     glad_glRasterPos4f = (PFNGLRASTERPOS4FPROC) load(userptr, "glRasterPos4f");
5197     glad_glRasterPos4fv = (PFNGLRASTERPOS4FVPROC) load(userptr, "glRasterPos4fv");
5198     glad_glRasterPos4i = (PFNGLRASTERPOS4IPROC) load(userptr, "glRasterPos4i");
5199     glad_glRasterPos4iv = (PFNGLRASTERPOS4IVPROC) load(userptr, "glRasterPos4iv");
5200     glad_glRasterPos4s = (PFNGLRASTERPOS4SPROC) load(userptr, "glRasterPos4s");
5201     glad_glRasterPos4sv = (PFNGLRASTERPOS4SVPROC) load(userptr, "glRasterPos4sv");
5202     glad_glReadBuffer = (PFNGLREADBUFFERPROC) load(userptr, "glReadBuffer");
5203     glad_glReadPixels = (PFNGLREADPIXELSPROC) load(userptr, "glReadPixels");
5204     glad_glRectd = (PFNGLRECTDPROC) load(userptr, "glRectd");
5205     glad_glRectdv = (PFNGLRECTDVPROC) load(userptr, "glRectdv");
5206     glad_glRectf = (PFNGLRECTFPROC) load(userptr, "glRectf");
5207     glad_glRectfv = (PFNGLRECTFVPROC) load(userptr, "glRectfv");
5208     glad_glRecti = (PFNGLRECTIPROC) load(userptr, "glRecti");
5209     glad_glRectiv = (PFNGLRECTIVPROC) load(userptr, "glRectiv");
5210     glad_glRects = (PFNGLRECTSPROC) load(userptr, "glRects");
5211     glad_glRectsv = (PFNGLRECTSVPROC) load(userptr, "glRectsv");
5212     glad_glRenderMode = (PFNGLRENDERMODEPROC) load(userptr, "glRenderMode");
5213     glad_glRotated = (PFNGLROTATEDPROC) load(userptr, "glRotated");
5214     glad_glRotatef = (PFNGLROTATEFPROC) load(userptr, "glRotatef");
5215     glad_glScaled = (PFNGLSCALEDPROC) load(userptr, "glScaled");
5216     glad_glScalef = (PFNGLSCALEFPROC) load(userptr, "glScalef");
```

```
5217     glad_glScissor = (PFNGLSCISSORPROC) load(userptr, "glScissor");
5218     glad_glSelectBuffer = (PFNGLSELECTBUFFERPROC) load(userptr, "glSelectBuffer");
5219     glad_glShadeModel = (PFNGLSHADEMODELPROC) load(userptr, "glShadeModel");
5220     glad_glStencilFunc = (PFNGLSTENCILFUNCPROC) load(userptr, "glStencilFunc");
5221     glad_glStencilMask = (PFNGLSTENCILMASKPROC) load(userptr, "glStencilMask");
5222     glad_glStencilOp = (PFNGLSTENCILOPPROC) load(userptr, "glStencilOp");
5223     glad_glTexCoord1d = (PFNGLTEXCOORD1DPROC) load(userptr, "glTexCoord1d");
5224     glad_glTexCoord1dv = (PFNGLTEXCOORD1DVPROC) load(userptr, "glTexCoord1dv");
5225     glad_glTexCoord1f = (PFNGLTEXCOORD1FPROC) load(userptr, "glTexCoord1f");
5226     glad_glTexCoord1fv = (PFNGLTEXCOORD1FVPROC) load(userptr, "glTexCoord1fv");
5227     glad_glTexCoord1i = (PFNGLTEXCOORD1IPROC) load(userptr, "glTexCoord1i");
5228     glad_glTexCoord1iv = (PFNGLTEXCOORD1IVPROC) load(userptr, "glTexCoord1iv");
5229     glad_glTexCoord1s = (PFNGLTEXCOORD1SPROC) load(userptr, "glTexCoord1s");
5230     glad_glTexCoord1sv = (PFNGLTEXCOORD1SVPROC) load(userptr, "glTexCoord1sv");
5231     glad_glTexCoord2d = (PFNGLTEXCOORD2DPROC) load(userptr, "glTexCoord2d");
5232     glad_glTexCoord2dv = (PFNGLTEXCOORD2DVPROC) load(userptr, "glTexCoord2dv");
5233     glad_glTexCoord2f = (PFNGLTEXCOORD2FPROC) load(userptr, "glTexCoord2f");
5234     glad_glTexCoord2fv = (PFNGLTEXCOORD2FVPROC) load(userptr, "glTexCoord2fv");
5235     glad_glTexCoord2i = (PFNGLTEXCOORD2IPROC) load(userptr, "glTexCoord2i");
5236     glad_glTexCoord2iv = (PFNGLTEXCOORD2IVPROC) load(userptr, "glTexCoord2iv");
5237     glad_glTexCoord2s = (PFNGLTEXCOORD2SPROC) load(userptr, "glTexCoord2s");
5238     glad_glTexCoord2sv = (PFNGLTEXCOORD2SVPROC) load(userptr, "glTexCoord2sv");
5239     glad_glTexCoord3d = (PFNGLTEXCOORD3DPROC) load(userptr, "glTexCoord3d");
5240     glad_glTexCoord3dv = (PFNGLTEXCOORD3DVPROC) load(userptr, "glTexCoord3dv");
5241     glad_glTexCoord3f = (PFNGLTEXCOORD3FPROC) load(userptr, "glTexCoord3f");
5242     glad_glTexCoord3fv = (PFNGLTEXCOORD3FVPROC) load(userptr, "glTexCoord3fv");
5243     glad_glTexCoord3i = (PFNGLTEXCOORD3IPROC) load(userptr, "glTexCoord3i");
5244     glad_glTexCoord3iv = (PFNGLTEXCOORD3IVPROC) load(userptr, "glTexCoord3iv");
5245     glad_glTexCoord3s = (PFNGLTEXCOORD3SPROC) load(userptr, "glTexCoord3s");
5246     glad_glTexCoord3sv = (PFNGLTEXCOORD3SVPROC) load(userptr, "glTexCoord3sv");
5247     glad_glTexCoord4d = (PFNGLTEXCOORD4DPROC) load(userptr, "glTexCoord4d");
5248     glad_glTexCoord4dv = (PFNGLTEXCOORD4DVPROC) load(userptr, "glTexCoord4dv");
5249     glad_glTexCoord4f = (PFNGLTEXCOORD4FPROC) load(userptr, "glTexCoord4f");
5250     glad_glTexCoord4fv = (PFNGLTEXCOORD4FVPROC) load(userptr, "glTexCoord4fv");
5251     glad_glTexCoord4i = (PFNGLTEXCOORD4IPROC) load(userptr, "glTexCoord4i");
5252     glad_glTexCoord4iv = (PFNGLTEXCOORD4IVPROC) load(userptr, "glTexCoord4iv");
5253     glad_glTexCoord4s = (PFNGLTEXCOORD4SPROC) load(userptr, "glTexCoord4s");
5254     glad_glTexCoord4sv = (PFNGLTEXCOORD4SVPROC) load(userptr, "glTexCoord4sv");
5255     glad_glTexEnvf = (PFNGLTEXENVFPROC) load(userptr, "glTexEnvf");
5256     glad_glTexEnvfv = (PFNGLTEXENVFVPROC) load(userptr, "glTexEnvfv");
5257     glad_glTexEnvi = (PFNGLTEXENVIPROC) load(userptr, "glTexEnvi");
5258     glad_glTexEnviv = (PFNGLTEXENVIVPROC) load(userptr, "glTexEnviv");
5259     glad_glTexGend = (PFNGLTEXGENDPROC) load(userptr, "glTexGend");
5260     glad_glTexGendv = (PFNGLTEXGENDVPROC) load(userptr, "glTexGendv");
5261     glad_glTexGenf = (PFNGLTEXGENFPROC) load(userptr, "glTexGenf");
5262     glad_glTexGenfv = (PFNGLTEXGENFVPROC) load(userptr, "glTexGenfv");
5263     glad_glTexGeni = (PFNGLTEXGENIPROC) load(userptr, "glTexGeni");
5264     glad_glTexGeniv = (PFNGLTEXGENIVPROC) load(userptr, "glTexGeniv");
5265     glad_glTexImage1D = (PFNGLTEXIMAGE1DPROC) load(userptr, "glTexImage1D");
5266     glad_glTexImage2D = (PFNGLTEXIMAGE2DPROC) load(userptr, "glTexImage2D");
5267     glad_glTexParameterf = (PFNGLTEXPARAMETERFPROC) load(userptr, "glTexParameterf");
5268     glad_glTexParameterfv = (PFNGLTEXPARAMETERFVPROC) load(userptr, "glTexParameterfv");
5269     glad_glTexParameteri = (PFNGLTEXPARAMETERIPROC) load(userptr, "glTexParameteri");
5270     glad_glTexParameteriv = (PFNGLTEXPARAMETERIVPROC) load(userptr, "glTexParameteriv");
5271     glad_glTranslated = (PFNGLTRANSLATEDPROC) load(userptr, "glTranslated");
5272     glad_glTranslatef = (PFNGLTRANSLATEFPROC) load(userptr, "glTranslatef");
5273     glad_glVertex2d = (PFNGLVERTEX2DPROC) load(userptr, "glVertex2d");
5274     glad_glVertex2dv = (PFNGLVERTEX2DVPROC) load(userptr, "glVertex2dv");
5275     glad_glVertex2f = (PFNGLVERTEX2FPROC) load(userptr, "glVertex2f");
5276     glad_glVertex2fv = (PFNGLVERTEX2FVPROC) load(userptr, "glVertex2fv");
5277     glad_glVertex2i = (PFNGLVERTEX2IPROC) load(userptr, "glVertex2i");
5278     glad_glVertex2iv = (PFNGLVERTEX2IVPROC) load(userptr, "glVertex2iv");
5279     glad_glVertex2s = (PFNGLVERTEX2SPROC) load(userptr, "glVertex2s");
5280     glad_glVertex2sv = (PFNGLVERTEX2SVPROC) load(userptr, "glVertex2sv");
5281     glad_glVertex3d = (PFNGLVERTEX3DPROC) load(userptr, "glVertex3d");
5282     glad_glVertex3dv = (PFNGLVERTEX3DVPROC) load(userptr, "glVertex3dv");
5283     glad_glVertex3f = (PFNGLVERTEX3FPROC) load(userptr, "glVertex3f");
5284     glad_glVertex3fv = (PFNGLVERTEX3FVPROC) load(userptr, "glVertex3fv");
5285     glad_glVertex3i = (PFNGLVERTEX3IPROC) load(userptr, "glVertex3i");
5286     glad_glVertex3iv = (PFNGLVERTEX3IVPROC) load(userptr, "glVertex3iv");
5287     glad_glVertex3s = (PFNGLVERTEX3SPROC) load(userptr, "glVertex3s");
5288     glad_glVertex3sv = (PFNGLVERTEX3SVPROC) load(userptr, "glVertex3sv");
5289     glad_glVertex4d = (PFNGLVERTEX4DPROC) load(userptr, "glVertex4d");
5290     glad_glVertex4dv = (PFNGLVERTEX4DVPROC) load(userptr, "glVertex4dv");
5291     glad_glVertex4f = (PFNGLVERTEX4FPROC) load(userptr, "glVertex4f");
5292     glad_glVertex4fv = (PFNGLVERTEX4FVPROC) load(userptr, "glVertex4fv");
5293     glad_glVertex4i = (PFNGLVERTEX4IPROC) load(userptr, "glVertex4i");
5294     glad_glVertex4iv = (PFNGLVERTEX4IVPROC) load(userptr, "glVertex4iv");
5295     glad_glVertex4s = (PFNGLVERTEX4SPROC) load(userptr, "glVertex4s");
5296     glad_glVertex4sv = (PFNGLVERTEX4SVPROC) load(userptr, "glVertex4sv");
5297     glad_glViewport = (PFNGLVIEWPORTPROC) load(userptr, "glViewport");
5298 }
5299 static void glad_gl_load_GL_VERSION_1_1( GLADuserptrloadfunc load, void* userptr) {
5300     if(!GLAD_GL_VERSION_1_1) return;
5301     glad_glAreTexturesResident = (PFNGLARETEXTURESRESIDENTPROC) load(userptr, "glAreTexturesResident");
5302     glad_glArrayElement = (PFNGLARRAYELEMENTPROC) load(userptr, "glArrayElement");
5303     glad_glBindTexture = (PFNGLBINDTEXTUREPROC) load(userptr, "glBindTexture");
```

```
5304     glad_glColorPointer = (PFNGLCOLORPOINTERPROC) load(userptr, "glColorPointer");
5305     glad_glCopyTexImage1D = (PFNGLCOPYTEXIMAGE1DPROC) load(userptr, "glCopyTexImage1D");
5306     glad_glCopyTexImage2D = (PFNGLCOPYTEXIMAGE2DPROC) load(userptr, "glCopyTexImage2D");
5307     glad_glCopyTexSubImage1D = (PFNGLCOPYTEXSUBIMAGE1DPROC) load(userptr, "glCopyTexSubImage1D");
5308     glad_glCopyTexSubImage2D = (PFNGLCOPYTEXSUBIMAGE2DPROC) load(userptr, "glCopyTexSubImage2D");
5309     glad_glDeleteTextures = (PFNGLDELETETEXTURESPROC) load(userptr, "glDeleteTextures");
5310     glad_glDisableClientState = (PFNGLDISABLECLIENTSTATEPROC) load(userptr, "glDisableClientState");
5311     glad_glDrawArrays = (PFNGLDRAWARRAYSPROC) load(userptr, "glDrawArrays");
5312     glad_glDrawElements = (PFNGLDRAWELEMENTSPROC) load(userptr, "glDrawElements");
5313     glad_glEdgeFlagPointer = (PFNGLEDGEFLAGPOINTERPROC) load(userptr, "glEdgeFlagPointer");
5314     glad_glEnableClientState = (PFNGLENABLECLIENTSTATEPROC) load(userptr, "glEnableClientState");
5315     glad_glGenTextures = (PFNGLGENTEXTURESPROC) load(userptr, "glGenTextures");
5316     glad_glGetPointerv = (PFNGLGETPOINTERVPROC) load(userptr, "glGetPointerv");
5317     glad_glIndexPointer = (PFNGLINDEXPOINTERPROC) load(userptr, "glIndexPointer");
5318     glad_glIndexub = (PFNGLINDEXUBPROC) load(userptr, "glIndexub");
5319     glad_glIndexubv = (PFNGLINDEXUBVPROC) load(userptr, "glIndexubv");
5320     glad_glInterleavedArrays = (PFNGLINTERLEAVEDARRAYSPROC) load(userptr, "glInterleavedArrays");
5321     glad_glIsTexture = (PFNGLISTEXTUREPROC) load(userptr, "glIsTexture");
5322     glad_glNormalPointer = (PFNGLNORMALPOINTERPROC) load(userptr, "glNormalPointer");
5323     glad_glPolygonOffset = (PFNGLPOLYGONOFFSETPROC) load(userptr, "glPolygonOffset");
5324     glad_glPopClientAttrib = (PFNGLPOPCLIENTATTRIBPROC) load(userptr, "glPopClientAttrib");
5325     glad_glPrioritizeTextures = (PFNGLPRIORITIZETEXTURESPROC) load(userptr, "glPrioritizeTextures");
5326     glad_glPushClientAttrib = (PFNGLPUSHCLIENTATTRIBPROC) load(userptr, "glPushClientAttrib");
5327     glad_glTexCoordPointer = (PFNGLTEXCOORDPOINTERPROC) load(userptr, "glTexCoordPointer");
5328     glad_glTexSubImage1D = (PFNGLTEXSUBIMAGE1DPROC) load(userptr, "glTexSubImage1D");
5329     glad_glTexSubImage2D = (PFNGLTEXSUBIMAGE2DPROC) load(userptr, "glTexSubImage2D");
5330     glad_glVertexPointer = (PFNGLVERTEXPOINTERPROC) load(userptr, "glVertexPointer");
5331 }
5332 static void glad_gl_load_GL_VERSION_1_2( GLADuserptrloadfunc load, void* userptr) {
5333     if(!GLAD_GL_VERSION_1_2) return;
5334     glad_glCopyTexSubImage3D = (PFNGLCOPYTEXSUBIMAGE3DPROC) load(userptr, "glCopyTexSubImage3D");
5335     glad_glDrawRangeElements = (PFNGLDRAWRANGEELEMENTSPROC) load(userptr, "glDrawRangeElements");
5336     glad_glTexImage3D = (PFNGLTEXIMAGE3DPROC) load(userptr, "glTexImage3D");
5337     glad_glTexSubImage3D = (PFNGLTEXSUBIMAGE3DPROC) load(userptr, "glTexSubImage3D");
5338 }
5339 static void glad_gl_load_GL_VERSION_1_3( GLADuserptrloadfunc load, void* userptr) {
5340     if(!GLAD_GL_VERSION_1_3) return;
5341     glad_glActiveTexture = (PFNGLACTIVETEXTUREPROC) load(userptr, "glActiveTexture");
5342     glad_glClientActiveTexture = (PFNGLCLIENTACTIVETEXTUREPROC) load(userptr, "glClientActiveTexture");
5343     glad_glCompressedTexImage1D = (PFNGLCOMPRESSEDTEXIMAGE1DPROC) load(userptr,
      "glCompressedTexImage1D");
5344     glad_glCompressedTexImage2D = (PFNGLCOMPRESSEDTEXIMAGE2DPROC) load(userptr,
      "glCompressedTexImage2D");
5345     glad_glCompressedTexImage3D = (PFNGLCOMPRESSEDTEXIMAGE3DPROC) load(userptr,
      "glCompressedTexImage3D");
5346     glad_glCompressedTexSubImage1D = (PFNGLCOMPRESSEDTEXSUBIMAGE1DPROC) load(userptr,
      "glCompressedTexSubImage1D");
5347     glad_glCompressedTexSubImage2D = (PFNGLCOMPRESSEDTEXSUBIMAGE2DPROC) load(userptr,
      "glCompressedTexSubImage2D");
5348     glad_glCompressedTexSubImage3D = (PFNGLCOMPRESSEDTEXSUBIMAGE3DPROC) load(userptr,
      "glCompressedTexSubImage3D");
5349     glad_glGetCompressedTexImage = (PFNGLGETCOMPRESSEDTEXIMAGEPROC) load(userptr,
      "glGetCompressedTexImage");
5350     glad_glLoadTransposeMatrixd = (PFNGLLOADTRANSPOSEMATRIXDPROC) load(userptr,
      "glLoadTransposeMatrixd");
5351     glad_glLoadTransposeMatrixf = (PFNGLLOADTRANSPOSEMATRIXFPROC) load(userptr,
      "glLoadTransposeMatrixf");
5352     glad_glMultTransposeMatrixd = (PFNGLMULTTRANSPOSEMATRIXDPROC) load(userptr,
      "glMultTransposeMatrixd");
5353     glad_glMultTransposeMatrixf = (PFNGLMULTTRANSPOSEMATRIXFPROC) load(userptr,
      "glMultTransposeMatrixf");
5354     glad_glMultiTexCoord1d = (PFNGLMULTITEXCOORD1DPROC) load(userptr, "glMultiTexCoord1d");
5355     glad_glMultiTexCoord1dv = (PFNGLMULTITEXCOORD1DVPROC) load(userptr, "glMultiTexCoord1dv");
5356     glad_glMultiTexCoord1f = (PFNGLMULTITEXCOORD1FPROC) load(userptr, "glMultiTexCoord1f");
5357     glad_glMultiTexCoord1fv = (PFNGLMULTITEXCOORD1FVPROC) load(userptr, "glMultiTexCoord1fv");
5358     glad_glMultiTexCoord1i = (PFNGLMULTITEXCOORD1IPROC) load(userptr, "glMultiTexCoord1i");
5359     glad_glMultiTexCoord1iv = (PFNGLMULTITEXCOORD1IVPROC) load(userptr, "glMultiTexCoord1iv");
5360     glad_glMultiTexCoord1s = (PFNGLMULTITEXCOORD1SPROC) load(userptr, "glMultiTexCoord1s");
5361     glad_glMultiTexCoord1sv = (PFNGLMULTITEXCOORD1SVPROC) load(userptr, "glMultiTexCoord1sv");
5362     glad_glMultiTexCoord2d = (PFNGLMULTITEXCOORD2DPROC) load(userptr, "glMultiTexCoord2d");
5363     glad_glMultiTexCoord2dv = (PFNGLMULTITEXCOORD2DVPROC) load(userptr, "glMultiTexCoord2dv");
5364     glad_glMultiTexCoord2f = (PFNGLMULTITEXCOORD2FPROC) load(userptr, "glMultiTexCoord2f");
5365     glad_glMultiTexCoord2fv = (PFNGLMULTITEXCOORD2FVPROC) load(userptr, "glMultiTexCoord2fv");
5366     glad_glMultiTexCoord2i = (PFNGLMULTITEXCOORD2IPROC) load(userptr, "glMultiTexCoord2i");
5367     glad_glMultiTexCoord2iv = (PFNGLMULTITEXCOORD2IVPROC) load(userptr, "glMultiTexCoord2iv");
5368     glad_glMultiTexCoord2s = (PFNGLMULTITEXCOORD2SPROC) load(userptr, "glMultiTexCoord2s");
5369     glad_glMultiTexCoord2sv = (PFNGLMULTITEXCOORD2SVPROC) load(userptr, "glMultiTexCoord2sv");
5370     glad_glMultiTexCoord3d = (PFNGLMULTITEXCOORD3DPROC) load(userptr, "glMultiTexCoord3d");
5371     glad_glMultiTexCoord3dv = (PFNGLMULTITEXCOORD3DVPROC) load(userptr, "glMultiTexCoord3dv");
5372     glad_glMultiTexCoord3f = (PFNGLMULTITEXCOORD3FPROC) load(userptr, "glMultiTexCoord3f");
5373     glad_glMultiTexCoord3fv = (PFNGLMULTITEXCOORD3FVPROC) load(userptr, "glMultiTexCoord3fv");
5374     glad_glMultiTexCoord3i = (PFNGLMULTITEXCOORD3IPROC) load(userptr, "glMultiTexCoord3i");
5375     glad_glMultiTexCoord3iv = (PFNGLMULTITEXCOORD3IVPROC) load(userptr, "glMultiTexCoord3iv");
5376     glad_glMultiTexCoord3s = (PFNGLMULTITEXCOORD3SPROC) load(userptr, "glMultiTexCoord3s");
5377     glad_glMultiTexCoord3sv = (PFNGLMULTITEXCOORD3SVPROC) load(userptr, "glMultiTexCoord3sv");
5378     glad_glMultiTexCoord4d = (PFNGLMULTITEXCOORD4DPROC) load(userptr, "glMultiTexCoord4d");
5379     glad_glMultiTexCoord4dv = (PFNGLMULTITEXCOORD4DVPROC) load(userptr, "glMultiTexCoord4dv");
```

```
5380     glad_glMultiTexCoord4f = (PFNGLMULTITEXCOORD4FPROC) load(userptr, "glMultiTexCoord4f");
5381     glad_glMultiTexCoord4fv = (PFNGLMULTITEXCOORD4FVPROC) load(userptr, "glMultiTexCoord4fv");
5382     glad_glMultiTexCoord4i = (PFNGLMULTITEXCOORD4IPROC) load(userptr, "glMultiTexCoord4i");
5383     glad_glMultiTexCoord4iv = (PFNGLMULTITEXCOORD4IVPROC) load(userptr, "glMultiTexCoord4iv");
5384     glad_glMultiTexCoord4s = (PFNGLMULTITEXCOORD4SPROC) load(userptr, "glMultiTexCoord4s");
5385     glad_glMultiTexCoord4sv = (PFNGLMULTITEXCOORD4SVPROC) load(userptr, "glMultiTexCoord4sv");
5386     glad_glSampleCoverage = (PFNGLSAMPLECOVERAGEPROC) load(userptr, "glSampleCoverage");
5387 }
5388 static void glad_gl_load_GL_VERSION_1_4( GLADuserptrloadfunc load, void* userptr) {
5389     if(!GLAD_GL_VERSION_1_4) return;
5390     glad_glBlendColor = (PFNGLBLENDCOLORPROC) load(userptr, "glBlendColor");
5391     glad_glBlendEquation = (PFNGLBLENDEQUATIONPROC) load(userptr, "glBlendEquation");
5392     glad_glBlendFuncSeparate = (PFNGLBLENDFUNCSEPARATEPROC) load(userptr, "glBlendFuncSeparate");
5393     glad_glFogCoordPointer = (PFNGLFOGCOORDPOINTERPROC) load(userptr, "glFogCoordPointer");
5394     glad_glFogCoordd = (PFNGLFOGCOORDDPROC) load(userptr, "glFogCoordd");
5395     glad_glFogCoorddv = (PFNGLFOGCOORDDVPROC) load(userptr, "glFogCoorddv");
5396     glad_glFogCoordf = (PFNGLFOGCOORDFPROC) load(userptr, "glFogCoordf");
5397     glad_glFogCoordfv = (PFNGLFOGCOORDFVPROC) load(userptr, "glFogCoordfv");
5398     glad_glMultiDrawArrays = (PFNGLMULTIDRAWARRAYSPROC) load(userptr, "glMultiDrawArrays");
5399     glad_glMultiDrawElements = (PFNGLMULTIDRAWELEMENTSPROC) load(userptr, "glMultiDrawElements");
5400     glad_glPointParameterf = (PFNGLPOINTPARAMETERFPROC) load(userptr, "glPointParameterf");
5401     glad_glPointParameterfv = (PFNGLPOINTPARAMETERFVPROC) load(userptr, "glPointParameterfv");
5402     glad_glPointParameteri = (PFNGLPOINTPARAMETERIPROC) load(userptr, "glPointParameteri");
5403     glad_glPointParameteriv = (PFNGLPOINTPARAMETERIVPROC) load(userptr, "glPointParameteriv");
5404     glad_glSecondaryColor3b = (PFNGLSECONDARYCOLOR3BPROC) load(userptr, "glSecondaryColor3b");
5405     glad_glSecondaryColor3bv = (PFNGLSECONDARYCOLOR3BVPROC) load(userptr, "glSecondaryColor3bv");
5406     glad_glSecondaryColor3d = (PFNGLSECONDARYCOLOR3DPROC) load(userptr, "glSecondaryColor3d");
5407     glad_glSecondaryColor3dv = (PFNGLSECONDARYCOLOR3DVPROC) load(userptr, "glSecondaryColor3dv");
5408     glad_glSecondaryColor3f = (PFNGLSECONDARYCOLOR3FPROC) load(userptr, "glSecondaryColor3f");
5409     glad_glSecondaryColor3fv = (PFNGLSECONDARYCOLOR3FVPROC) load(userptr, "glSecondaryColor3fv");
5410     glad_glSecondaryColor3i = (PFNGLSECONDARYCOLOR3IPROC) load(userptr, "glSecondaryColor3i");
5411     glad_glSecondaryColor3iv = (PFNGLSECONDARYCOLOR3IVPROC) load(userptr, "glSecondaryColor3iv");
5412     glad_glSecondaryColor3s = (PFNGLSECONDARYCOLOR3SPROC) load(userptr, "glSecondaryColor3s");
5413     glad_glSecondaryColor3sv = (PFNGLSECONDARYCOLOR3SVPROC) load(userptr, "glSecondaryColor3sv");
5414     glad_glSecondaryColor3ub = (PFNGLSECONDARYCOLOR3UBPROC) load(userptr, "glSecondaryColor3ub");
5415     glad_glSecondaryColor3ubv = (PFNGLSECONDARYCOLOR3UBVPROC) load(userptr, "glSecondaryColor3ubv");
5416     glad_glSecondaryColor3ui = (PFNGLSECONDARYCOLOR3UIPROC) load(userptr, "glSecondaryColor3ui");
5417     glad_glSecondaryColor3uiv = (PFNGLSECONDARYCOLOR3UIVPROC) load(userptr, "glSecondaryColor3uiv");
5418     glad_glSecondaryColor3us = (PFNGLSECONDARYCOLOR3USPROC) load(userptr, "glSecondaryColor3us");
5419     glad_glSecondaryColor3usv = (PFNGLSECONDARYCOLOR3USVPROC) load(userptr, "glSecondaryColor3usv");
5420     glad_glSecondaryColorPointer = (PFNGLSECONDARYCOLORPOINTERPROC) load(userptr,
     "glSecondaryColorPointer");
5421     glad_glWindowPos2d = (PFNGLWINDOWPOS2DPROC) load(userptr, "glWindowPos2d");
5422     glad_glWindowPos2dv = (PFNGLWINDOWPOS2DVPROC) load(userptr, "glWindowPos2dv");
5423     glad_glWindowPos2f = (PFNGLWINDOWPOS2FPROC) load(userptr, "glWindowPos2f");
5424     glad_glWindowPos2fv = (PFNGLWINDOWPOS2FVPROC) load(userptr, "glWindowPos2fv");
5425     glad_glWindowPos2i = (PFNGLWINDOWPOS2IPROC) load(userptr, "glWindowPos2i");
5426     glad_glWindowPos2iv = (PFNGLWINDOWPOS2IVPROC) load(userptr, "glWindowPos2iv");
5427     glad_glWindowPos2s = (PFNGLWINDOWPOS2SPROC) load(userptr, "glWindowPos2s");
5428     glad_glWindowPos2sv = (PFNGLWINDOWPOS2SVPROC) load(userptr, "glWindowPos2sv");
5429     glad_glWindowPos3d = (PFNGLWINDOWPOS3DPROC) load(userptr, "glWindowPos3d");
5430     glad_glWindowPos3dv = (PFNGLWINDOWPOS3DVPROC) load(userptr, "glWindowPos3dv");
5431     glad_glWindowPos3f = (PFNGLWINDOWPOS3FPROC) load(userptr, "glWindowPos3f");
5432     glad_glWindowPos3fv = (PFNGLWINDOWPOS3FVPROC) load(userptr, "glWindowPos3fv");
5433     glad_glWindowPos3i = (PFNGLWINDOWPOS3IPROC) load(userptr, "glWindowPos3i");
5434     glad_glWindowPos3iv = (PFNGLWINDOWPOS3IVPROC) load(userptr, "glWindowPos3iv");
5435     glad_glWindowPos3s = (PFNGLWINDOWPOS3SPROC) load(userptr, "glWindowPos3s");
5436     glad_glWindowPos3sv = (PFNGLWINDOWPOS3SVPROC) load(userptr, "glWindowPos3sv");
5437 }
5438 static void glad_gl_load_GL_VERSION_1_5( GLADuserptrloadfunc load, void* userptr) {
5439     if(!GLAD_GL_VERSION_1_5) return;
5440     glad_glBeginQuery = (PFNGLBEGINQUERYPROC) load(userptr, "glBeginQuery");
5441     glad_glBindBuffer = (PFNGLBINDBUFFERPROC) load(userptr, "glBindBuffer");
5442     glad_glBufferData = (PFNGLBUFFERDATAPROC) load(userptr, "glBufferData");
5443     glad_glBufferSubData = (PFNGLBUFFERSUBDATAPROC) load(userptr, "glBufferSubData");
5444     glad_glDeleteBuffers = (PFNGLDELETEBUFFERSPROC) load(userptr, "glDeleteBuffers");
5445     glad_glDeleteQueries = (PFNGLDELETEQUERIESPROC) load(userptr, "glDeleteQueries");
5446     glad_glEndQuery = (PFNGLENDQUERYPROC) load(userptr, "glEndQuery");
5447     glad_glGenBuffers = (PFNGLGENBUFFERSPROC) load(userptr, "glGenBuffers");
5448     glad_glGenQueries = (PFNGLGENQUERIESPROC) load(userptr, "glGenQueries");
5449     glad_glGetBufferParameteriv = (PFNGLGETBUFFERPARAMETERIVPROC) load(userptr,
     "glGetBufferParameteriv");
5450     glad_glGetBufferPointerv = (PFNGLGETBUFFERPOINTERVPROC) load(userptr, "glGetBufferPointerv");
5451     glad_glGetBufferSubData = (PFNGLGETBUFFERSUBDATAPROC) load(userptr, "glGetBufferSubData");
5452     glad_glGetQueryObjectiv = (PFNGLGETQUERYOBJECTIVPROC) load(userptr, "glGetQueryObjectiv");
5453     glad_glGetQueryObjectuiv = (PFNGLGETQUERYOBJECTUIVPROC) load(userptr, "glGetQueryObjectuiv");
5454     glad_glGetQueryiv = (PFNGLGETQUERYIVPROC) load(userptr, "glGetQueryiv");
5455     glad_glIsBuffer = (PFNGLISBUFFERPROC) load(userptr, "glIsBuffer");
5456     glad_glIsQuery = (PFNGLISQUERYPROC) load(userptr, "glIsQuery");
5457     glad_glMapBuffer = (PFNGLMAPBUFFERPROC) load(userptr, "glMapBuffer");
5458     glad_glUnmapBuffer = (PFNGLUNMAPBUFFERPROC) load(userptr, "glUnmapBuffer");
5459 }
5460 static void glad_gl_load_GL_VERSION_2_0( GLADuserptrloadfunc load, void* userptr) {
5461     if(!GLAD_GL_VERSION_2_0) return;
5462     glad_glAttachShader = (PFNGLATTACHSHADERPROC) load(userptr, "glAttachShader");
5463     glad_glBindAttribLocation = (PFNGLBINDATTRIBLOCATIONPROC) load(userptr, "glBindAttribLocation");
5464     glad_glBlendEquationSeparate = (PFNGLBLENDEQUATIONSEPARATEPROC) load(userptr,
```

```
    "glBlendEquationSeparate");
5465    glad_glCompileShader = (PFNGLCOMPILESHADERPROC) load(userptr, "glCompileShader");
5466    glad_glCreateProgram = (PFNGLCREATEPROGRAMPROC) load(userptr, "glCreateProgram");
5467    glad_glCreateShader = (PFNGLCREATESHADERPROC) load(userptr, "glCreateShader");
5468    glad_glDeleteProgram = (PFNGLDELETEPROGRAMPROC) load(userptr, "glDeleteProgram");
5469    glad_glDeleteShader = (PFNGLDELETESHADERPROC) load(userptr, "glDeleteShader");
5470    glad_glDetachShader = (PFNGLDETACHSHADERPROC) load(userptr, "glDetachShader");
5471    glad_glDisableVertexAttribArray = (PFNGLDISABLEVERTEXATTRIBARRAYPROC) load(userptr,
    "glDisableVertexAttribArray");
5472    glad_glDrawBuffers = (PFNGLDRAWBUFFERSPROC) load(userptr, "glDrawBuffers");
5473    glad_glEnableVertexAttribArray = (PFNGLENABLEVERTEXATTRIBARRAYPROC) load(userptr,
    "glEnableVertexAttribArray");
5474    glad_glGetActiveAttrib = (PFNGLGETACTIVEATTRIBPROC) load(userptr, "glGetActiveAttrib");
5475    glad_glGetActiveUniform = (PFNGLGETACTIVEUNIFORMPROC) load(userptr, "glGetActiveUniform");
5476    glad_glGetAttachedShaders = (PFNGLGETATTACHEDSHADERSPROC) load(userptr, "glGetAttachedShaders");
5477    glad_glGetAttribLocation = (PFNGLGETATTRIBLOCATIONPROC) load(userptr, "glGetAttribLocation");
5478    glad_glGetProgramInfoLog = (PFNGLGETPROGRAMINFOLOGPROC) load(userptr, "glGetProgramInfoLog");
5479    glad_glGetProgramiv = (PFNGLGETPROGRAMIVPROC) load(userptr, "glGetProgramiv");
5480    glad_glGetShaderInfoLog = (PFNGLGETSHADERINFOLOGPROC) load(userptr, "glGetShaderInfoLog");
5481    glad_glGetShaderSource = (PFNGLGETSHADERSOURCEPROC) load(userptr, "glGetShaderSource");
5482    glad_glGetShaderiv = (PFNGLGETSHADERIVPROC) load(userptr, "glGetShaderiv");
5483    glad_glGetUniformLocation = (PFNGLGETUNIFORMLOCATIONPROC) load(userptr, "glGetUniformLocation");
5484    glad_glGetUniformfv = (PFNGLGETUNIFORMFVPROC) load(userptr, "glGetUniformfv");
5485    glad_glGetUniformiv = (PFNGLGETUNIFORMIVPROC) load(userptr, "glGetUniformiv");
5486    glad_glGetVertexAttribPointerv = (PFNGLGETVERTEXATTRIBPOINTERVPROC) load(userptr,
    "glGetVertexAttribPointerv");
5487    glad_glGetVertexAttribdv = (PFNGLGETVERTEXATTRIBDVPROC) load(userptr, "glGetVertexAttribdv");
5488    glad_glGetVertexAttribfv = (PFNGLGETVERTEXATTRIBFVPROC) load(userptr, "glGetVertexAttribfv");
5489    glad_glGetVertexAttribiv = (PFNGLGETVERTEXATTRIBIVPROC) load(userptr, "glGetVertexAttribiv");
5490    glad_glIsProgram = (PFNGLISPROGRAMPROC) load(userptr, "glIsProgram");
5491    glad_glIsShader = (PFNGLISSHADERPROC) load(userptr, "glIsShader");
5492    glad_glLinkProgram = (PFNGLLINKPROGRAMPROC) load(userptr, "glLinkProgram");
5493    glad_glShaderSource = (PFNGLSHADERSOURCEPROC) load(userptr, "glShaderSource");
5494    glad_glStencilFuncSeparate = (PFNGLSTENCILFUNCSEPARATEPROC) load(userptr, "glStencilFuncSeparate");
5495    glad_glStencilMaskSeparate = (PFNGLSTENCILMASKSEPARATEPROC) load(userptr, "glStencilMaskSeparate");
5496    glad_glStencilOpSeparate = (PFNGLSTENCILOPSEPARATEPROC) load(userptr, "glStencilOpSeparate");
5497    glad_glUniform1f = (PFNGLUNIFORM1FPROC) load(userptr, "glUniform1f");
5498    glad_glUniform1fv = (PFNGLUNIFORM1FVPROC) load(userptr, "glUniform1fv");
5499    glad_glUniform1i = (PFNGLUNIFORM1IPROC) load(userptr, "glUniform1i");
5500    glad_glUniform1iv = (PFNGLUNIFORM1IVPROC) load(userptr, "glUniform1iv");
5501    glad_glUniform2f = (PFNGLUNIFORM2FPROC) load(userptr, "glUniform2f");
5502    glad_glUniform2fv = (PFNGLUNIFORM2FVPROC) load(userptr, "glUniform2fv");
5503    glad_glUniform2i = (PFNGLUNIFORM2IPROC) load(userptr, "glUniform2i");
5504    glad_glUniform2iv = (PFNGLUNIFORM2IVPROC) load(userptr, "glUniform2iv");
5505    glad_glUniform3f = (PFNGLUNIFORM3FPROC) load(userptr, "glUniform3f");
5506    glad_glUniform3fv = (PFNGLUNIFORM3FVPROC) load(userptr, "glUniform3fv");
5507    glad_glUniform3i = (PFNGLUNIFORM3IPROC) load(userptr, "glUniform3i");
5508    glad_glUniform3iv = (PFNGLUNIFORM3IVPROC) load(userptr, "glUniform3iv");
5509    glad_glUniform4f = (PFNGLUNIFORM4FPROC) load(userptr, "glUniform4f");
5510    glad_glUniform4fv = (PFNGLUNIFORM4FVPROC) load(userptr, "glUniform4fv");
5511    glad_glUniform4i = (PFNGLUNIFORM4IPROC) load(userptr, "glUniform4i");
5512    glad_glUniform4iv = (PFNGLUNIFORM4IVPROC) load(userptr, "glUniform4iv");
5513    glad_glUniformMatrix2fv = (PFNGLUNIFORMMATRIX2FVPROC) load(userptr, "glUniformMatrix2fv");
5514    glad_glUniformMatrix3fv = (PFNGLUNIFORMMATRIX3FVPROC) load(userptr, "glUniformMatrix3fv");
5515    glad_glUniformMatrix4fv = (PFNGLUNIFORMMATRIX4FVPROC) load(userptr, "glUniformMatrix4fv");
5516    glad_glUseProgram = (PFNGLUSEPROGRAMPROC) load(userptr, "glUseProgram");
5517    glad_glValidateProgram = (PFNGLVALIDATEPROGRAMPROC) load(userptr, "glValidateProgram");
5518    glad_glVertexAttrib1d = (PFNGLVERTEXATTRIB1DPROC) load(userptr, "glVertexAttrib1d");
5519    glad_glVertexAttrib1dv = (PFNGLVERTEXATTRIB1DVPROC) load(userptr, "glVertexAttrib1dv");
5520    glad_glVertexAttrib1f = (PFNGLVERTEXATTRIB1FPROC) load(userptr, "glVertexAttrib1f");
5521    glad_glVertexAttrib1fv = (PFNGLVERTEXATTRIB1FVPROC) load(userptr, "glVertexAttrib1fv");
5522    glad_glVertexAttrib1s = (PFNGLVERTEXATTRIB1SPROC) load(userptr, "glVertexAttrib1s");
5523    glad_glVertexAttrib1sv = (PFNGLVERTEXATTRIB1SVPROC) load(userptr, "glVertexAttrib1sv");
5524    glad_glVertexAttrib2d = (PFNGLVERTEXATTRIB2DPROC) load(userptr, "glVertexAttrib2d");
5525    glad_glVertexAttrib2dv = (PFNGLVERTEXATTRIB2DVPROC) load(userptr, "glVertexAttrib2dv");
5526    glad_glVertexAttrib2f = (PFNGLVERTEXATTRIB2FPROC) load(userptr, "glVertexAttrib2f");
5527    glad_glVertexAttrib2fv = (PFNGLVERTEXATTRIB2FVPROC) load(userptr, "glVertexAttrib2fv");
5528    glad_glVertexAttrib2s = (PFNGLVERTEXATTRIB2SPROC) load(userptr, "glVertexAttrib2s");
5529    glad_glVertexAttrib2sv = (PFNGLVERTEXATTRIB2SVPROC) load(userptr, "glVertexAttrib2sv");
5530    glad_glVertexAttrib3d = (PFNGLVERTEXATTRIB3DPROC) load(userptr, "glVertexAttrib3d");
5531    glad_glVertexAttrib3dv = (PFNGLVERTEXATTRIB3DVPROC) load(userptr, "glVertexAttrib3dv");
5532    glad_glVertexAttrib3f = (PFNGLVERTEXATTRIB3FPROC) load(userptr, "glVertexAttrib3f");
5533    glad_glVertexAttrib3fv = (PFNGLVERTEXATTRIB3FVPROC) load(userptr, "glVertexAttrib3fv");
5534    glad_glVertexAttrib3s = (PFNGLVERTEXATTRIB3SPROC) load(userptr, "glVertexAttrib3s");
5535    glad_glVertexAttrib3sv = (PFNGLVERTEXATTRIB3SVPROC) load(userptr, "glVertexAttrib3sv");
5536    glad_glVertexAttrib4Nbv = (PFNGLVERTEXATTRIB4NBVPROC) load(userptr, "glVertexAttrib4Nbv");
5537    glad_glVertexAttrib4Niv = (PFNGLVERTEXATTRIB4NIVPROC) load(userptr, "glVertexAttrib4Niv");
5538    glad_glVertexAttrib4Nsv = (PFNGLVERTEXATTRIB4NSVPROC) load(userptr, "glVertexAttrib4Nsv");
5539    glad_glVertexAttrib4Nub = (PFNGLVERTEXATTRIB4NUBPROC) load(userptr, "glVertexAttrib4Nub");
5540    glad_glVertexAttrib4Nubv = (PFNGLVERTEXATTRIB4NUBVPROC) load(userptr, "glVertexAttrib4Nubv");
5541    glad_glVertexAttrib4Nuiv = (PFNGLVERTEXATTRIB4NUIVPROC) load(userptr, "glVertexAttrib4Nuiv");
5542    glad_glVertexAttrib4Nusv = (PFNGLVERTEXATTRIB4NUSVPROC) load(userptr, "glVertexAttrib4Nusv");
5543    glad_glVertexAttrib4bv = (PFNGLVERTEXATTRIB4BVPROC) load(userptr, "glVertexAttrib4bv");
5544    glad_glVertexAttrib4d = (PFNGLVERTEXATTRIB4DPROC) load(userptr, "glVertexAttrib4d");
5545    glad_glVertexAttrib4dv = (PFNGLVERTEXATTRIB4DVPROC) load(userptr, "glVertexAttrib4dv");
5546    glad_glVertexAttrib4f = (PFNGLVERTEXATTRIB4FPROC) load(userptr, "glVertexAttrib4f");
5547    glad_glVertexAttrib4fv = (PFNGLVERTEXATTRIB4FVPROC) load(userptr, "glVertexAttrib4fv");
```

```
5548     glad_glVertexAttrib4iv = (PFNGLVERTEXATTRIB4IVPROC) load(userptr, "glVertexAttrib4iv");
5549     glad_glVertexAttrib4s = (PFNGLVERTEXATTRIB4SPROC) load(userptr, "glVertexAttrib4s");
5550     glad_glVertexAttrib4sv = (PFNGLVERTEXATTRIB4SVPROC) load(userptr, "glVertexAttrib4sv");
5551     glad_glVertexAttrib4ubv = (PFNGLVERTEXATTRIB4UBVPROC) load(userptr, "glVertexAttrib4ubv");
5552     glad_glVertexAttrib4uiv = (PFNGLVERTEXATTRIB4UIVPROC) load(userptr, "glVertexAttrib4uiv");
5553     glad_glVertexAttrib4usv = (PFNGLVERTEXATTRIB4USVPROC) load(userptr, "glVertexAttrib4usv");
5554     glad_glVertexAttribPointer = (PFNGLVERTEXATTRIBPOINTERPROC) load(userptr, "glVertexAttribPointer");
5555 }
5556 static void glad_gl_load_GL_VERSION_2_1( GLADuserptrloadfunc load, void* userptr) {
5557     if(!GLAD_GL_VERSION_2_1) return;
5558     glad_glUniformMatrix2x3fv = (PFNGLUNIFORMMATRIX2X3FVPROC) load(userptr, "glUniformMatrix2x3fv");
5559     glad_glUniformMatrix2x4fv = (PFNGLUNIFORMMATRIX2X4FVPROC) load(userptr, "glUniformMatrix2x4fv");
5560     glad_glUniformMatrix3x2fv = (PFNGLUNIFORMMATRIX3X2FVPROC) load(userptr, "glUniformMatrix3x2fv");
5561     glad_glUniformMatrix3x4fv = (PFNGLUNIFORMMATRIX3X4FVPROC) load(userptr, "glUniformMatrix3x4fv");
5562     glad_glUniformMatrix4x2fv = (PFNGLUNIFORMMATRIX4X2FVPROC) load(userptr, "glUniformMatrix4x2fv");
5563     glad_glUniformMatrix4x3fv = (PFNGLUNIFORMMATRIX4X3FVPROC) load(userptr, "glUniformMatrix4x3fv");
5564 }
5565 static void glad_gl_load_GL_VERSION_3_0( GLADuserptrloadfunc load, void* userptr) {
5566     if(!GLAD_GL_VERSION_3_0) return;
5567     glad_glBeginConditionalRender = (PFNGLBEGINCONDITIONALRENDERPROC) load(userptr,
      "glBeginConditionalRender");
5568     glad_glBeginTransformFeedback = (PFNGLBEGINTRANSFORMFEEDBACKPROC) load(userptr,
      "glBeginTransformFeedback");
5569     glad_glBindBufferBase = (PFNGLBINDBUFFERBASEPROC) load(userptr, "glBindBufferBase");
5570     glad_glBindBufferRange = (PFNGLBINDBUFFERRANGEPROC) load(userptr, "glBindBufferRange");
5571     glad_glBindFragDataLocation = (PFNGLBINDFRAGDATALOCATIONPROC) load(userptr,
      "glBindFragDataLocation");
5572     glad_glBindFramebuffer = (PFNGLBINDFRAMEBUFFERPROC) load(userptr, "glBindFramebuffer");
5573     glad_glBindRenderbuffer = (PFNGLBINDRENDERBUFFERPROC) load(userptr, "glBindRenderbuffer");
5574     glad_glBindVertexArray = (PFNGLBINDVERTEXARRAYPROC) load(userptr, "glBindVertexArray");
5575     glad_glBlitFramebuffer = (PFNGLBLITFRAMEBUFFERPROC) load(userptr, "glBlitFramebuffer");
5576     glad_glCheckFramebufferStatus = (PFNGLCHECKFRAMEBUFFERSTATUSPROC) load(userptr,
      "glCheckFramebufferStatus");
5577     glad_glClampColor = (PFNGLCLAMPCOLORPROC) load(userptr, "glClampColor");
5578     glad_glClearBufferfi = (PFNGLCLEARBUFFERFIPROC) load(userptr, "glClearBufferfi");
5579     glad_glClearBufferfv = (PFNGLCLEARBUFFERFVPROC) load(userptr, "glClearBufferfv");
5580     glad_glClearBufferiv = (PFNGLCLEARBUFFERIVPROC) load(userptr, "glClearBufferiv");
5581     glad_glClearBufferuiv = (PFNGLCLEARBUFFERUIVPROC) load(userptr, "glClearBufferuiv");
5582     glad_glColorMaski = (PFNGLCOLORMASKIPROC) load(userptr, "glColorMaski");
5583     glad_glDeleteFramebuffers = (PFNGLDELETEFRAMEBUFFERSPROC) load(userptr, "glDeleteFramebuffers");
5584     glad_glDeleteRenderbuffers = (PFNGLDELETERENDERBUFFERSPROC) load(userptr, "glDeleteRenderbuffers");
5585     glad_glDeleteVertexArrays = (PFNGLDELETEVERTEXARRAYSPROC) load(userptr, "glDeleteVertexArrays");
5586     glad_glDisablei = (PFNGLDISABLEIPROC) load(userptr, "glDisablei");
5587     glad_glEnablei = (PFNGLENABLEIPROC) load(userptr, "glEnablei");
5588     glad_glEndConditionalRender = (PFNGLENDCONDITIONALRENDERPROC) load(userptr,
      "glEndConditionalRender");
5589     glad_glEndTransformFeedback = (PFNGLENDTRANSFORMFEEDBACKPROC) load(userptr,
      "glEndTransformFeedback");
5590     glad_glFlushMappedBufferRange = (PFNGLFLUSHMAPPEDBUFFERRANGEPROC) load(userptr,
      "glFlushMappedBufferRange");
5591     glad_glFramebufferRenderbuffer = (PFNGLFRAMEBUFFERRENDERBUFFERPROC) load(userptr,
      "glFramebufferRenderbuffer");
5592     glad_glFramebufferTexture1D = (PFNGLFRAMEBUFFERTEXTURE1DPROC) load(userptr,
      "glFramebufferTexture1D");
5593     glad_glFramebufferTexture2D = (PFNGLFRAMEBUFFERTEXTURE2DPROC) load(userptr,
      "glFramebufferTexture2D");
5594     glad_glFramebufferTexture3D = (PFNGLFRAMEBUFFERTEXTURE3DPROC) load(userptr,
      "glFramebufferTexture3D");
5595     glad_glFramebufferTextureLayer = (PFNGLFRAMEBUFFERTEXTURELAYERPROC) load(userptr,
      "glFramebufferTextureLayer");
5596     glad_glGenFramebuffers = (PFNGLGENFRAMEBUFFERSPROC) load(userptr, "glGenFramebuffers");
5597     glad_glGenRenderbuffers = (PFNGLGENRENDERBUFFERSPROC) load(userptr, "glGenRenderbuffers");
5598     glad_glGenVertexArrays = (PFNGLGENVERTEXARRAYSPROC) load(userptr, "glGenVertexArrays");
5599     glad_glGenerateMipmap = (PFNGLGENERATEMIPMAPPROC) load(userptr, "glGenerateMipmap");
5600     glad_glGetBooleani_v = (PFNGLGETBOOLEANI_VPROC) load(userptr, "glGetBooleani_v");
5601     glad_glGetFragDataLocation = (PFNGLGETFRAGDATALOCATIONPROC) load(userptr, "glGetFragDataLocation");
5602     glad_glGetFramebufferAttachmentParameteriv = (PFNGLGETFRAMEBUFFERATTACHMENTPARAMETERIVPROC)
      load(userptr, "glGetFramebufferAttachmentParameteriv");
5603     glad_glGetIntegeri_v = (PFNGLGETINTEGERI_VPROC) load(userptr, "glGetIntegeri_v");
5604     glad_glGetRenderbufferParameteriv = (PFNGLGETRENDERBUFFERPARAMETERIVPROC) load(userptr,
      "glGetRenderbufferParameteriv");
5605     glad_glGetStringi = (PFNGLGETSTRINGIPROC) load(userptr, "glGetStringi");
5606     glad_glGetTexParameterIiv = (PFNGLGETTEXPARAMETERIIVPROC) load(userptr, "glGetTexParameterIiv");
5607     glad_glGetTexParameterIuiv = (PFNGLGETTEXPARAMETERIUIVPROC) load(userptr, "glGetTexParameterIuiv");
5608     glad_glGetTransformFeedbackVarying = (PFNGLGETTRANSFORMFEEDBACKVARYINGPROC) load(userptr,
      "glGetTransformFeedbackVarying");
5609     glad_glGetUniformuiv = (PFNGLGETUNIFORMUIVPROC) load(userptr, "glGetUniformuiv");
5610     glad_glGetVertexAttribIiv = (PFNGLGETVERTEXATTRIBIIVPROC) load(userptr, "glGetVertexAttribIiv");
5611     glad_glGetVertexAttribIuiv = (PFNGLGETVERTEXATTRIBIUIVPROC) load(userptr, "glGetVertexAttribIuiv");
5612     glad_glIsEnabledi = (PFNGLISENABLEDIPROC) load(userptr, "glIsEnabledi");
5613     glad_glIsFramebuffer = (PFNGLISFRAMEBUFFERPROC) load(userptr, "glIsFramebuffer");
5614     glad_glIsRenderbuffer = (PFNGLISRENDERBUFFERPROC) load(userptr, "glIsRenderbuffer");
5615     glad_glIsVertexArray = (PFNGLISVERTEXARRAYPROC) load(userptr, "glIsVertexArray");
5616     glad_glMapBufferRange = (PFNGLMAPBUFFERRANGEPROC) load(userptr, "glMapBufferRange");
5617     glad_glRenderbufferStorage = (PFNGLRENDERBUFFERSTORAGEPROC) load(userptr, "glRenderbufferStorage");
5618     glad_glRenderbufferStorageMultisample = (PFNGLRENDERBUFFERSTORAGEMULTISAMPLEPROC) load(userptr,
      "glRenderbufferStorageMultisample");
```

```
5619     glad_glTexParameterIiv = (PFNGLTEXPARAMETERIIVPROC) load(userptr, "glTexParameterIiv");
5620     glad_glTexParameterIuiv = (PFNGLTEXPARAMETERIUIVPROC) load(userptr, "glTexParameterIuiv");
5621     glad_glTransformFeedbackVaryings = (PFNGLTRANSFORMFEEDBACKVARYINGSPROC) load(userptr,
     "glTransformFeedbackVaryings");
5622     glad_glUniform1ui = (PFNGLUNIFORM1UIPROC) load(userptr, "glUniform1ui");
5623     glad_glUniform1uiv = (PFNGLUNIFORM1UIVPROC) load(userptr, "glUniform1uiv");
5624     glad_glUniform2ui = (PFNGLUNIFORM2UIPROC) load(userptr, "glUniform2ui");
5625     glad_glUniform2uiv = (PFNGLUNIFORM2UIVPROC) load(userptr, "glUniform2uiv");
5626     glad_glUniform3ui = (PFNGLUNIFORM3UIPROC) load(userptr, "glUniform3ui");
5627     glad_glUniform3uiv = (PFNGLUNIFORM3UIVPROC) load(userptr, "glUniform3uiv");
5628     glad_glUniform4ui = (PFNGLUNIFORM4UIPROC) load(userptr, "glUniform4ui");
5629     glad_glUniform4uiv = (PFNGLUNIFORM4UIVPROC) load(userptr, "glUniform4uiv");
5630     glad_glVertexAttribI1i = (PFNGLVERTEXATTRIBI1IPROC) load(userptr, "glVertexAttribI1i");
5631     glad_glVertexAttribI1iv = (PFNGLVERTEXATTRIBI1IVPROC) load(userptr, "glVertexAttribI1iv");
5632     glad_glVertexAttribI1ui = (PFNGLVERTEXATTRIBI1UIPROC) load(userptr, "glVertexAttribI1ui");
5633     glad_glVertexAttribI1uiv = (PFNGLVERTEXATTRIBI1UIVPROC) load(userptr, "glVertexAttribI1uiv");
5634     glad_glVertexAttribI2i = (PFNGLVERTEXATTRIBI2IPROC) load(userptr, "glVertexAttribI2i");
5635     glad_glVertexAttribI2iv = (PFNGLVERTEXATTRIBI2IVPROC) load(userptr, "glVertexAttribI2iv");
5636     glad_glVertexAttribI2ui = (PFNGLVERTEXATTRIBI2UIPROC) load(userptr, "glVertexAttribI2ui");
5637     glad_glVertexAttribI2uiv = (PFNGLVERTEXATTRIBI2UIVPROC) load(userptr, "glVertexAttribI2uiv");
5638     glad_glVertexAttribI3i = (PFNGLVERTEXATTRIBI3IPROC) load(userptr, "glVertexAttribI3i");
5639     glad_glVertexAttribI3iv = (PFNGLVERTEXATTRIBI3IVPROC) load(userptr, "glVertexAttribI3iv");
5640     glad_glVertexAttribI3ui = (PFNGLVERTEXATTRIBI3UIPROC) load(userptr, "glVertexAttribI3ui");
5641     glad_glVertexAttribI3uiv = (PFNGLVERTEXATTRIBI3UIVPROC) load(userptr, "glVertexAttribI3uiv");
5642     glad_glVertexAttribI4bv = (PFNGLVERTEXATTRIBI4BVPROC) load(userptr, "glVertexAttribI4bv");
5643     glad_glVertexAttribI4i = (PFNGLVERTEXATTRIBI4IPROC) load(userptr, "glVertexAttribI4i");
5644     glad_glVertexAttribI4iv = (PFNGLVERTEXATTRIBI4IVPROC) load(userptr, "glVertexAttribI4iv");
5645     glad_glVertexAttribI4sv = (PFNGLVERTEXATTRIBI4SVPROC) load(userptr, "glVertexAttribI4sv");
5646     glad_glVertexAttribI4ubv = (PFNGLVERTEXATTRIBI4UBVPROC) load(userptr, "glVertexAttribI4ubv");
5647     glad_glVertexAttribI4ui = (PFNGLVERTEXATTRIBI4UIPROC) load(userptr, "glVertexAttribI4ui");
5648     glad_glVertexAttribI4uiv = (PFNGLVERTEXATTRIBI4UIVPROC) load(userptr, "glVertexAttribI4uiv");
5649     glad_glVertexAttribI4usv = (PFNGLVERTEXATTRIBI4USVPROC) load(userptr, "glVertexAttribI4usv");
5650     glad_glVertexAttribIPointer = (PFNGLVERTEXATTRIBIPOINTERPROC) load(userptr,
     "glVertexAttribIPointer");
5651 }
5652 static void glad_gl_load_GL_VERSION_3_1( GLADuserptrloadfunc load, void* userptr) {
5653     if(!GLAD_GL_VERSION_3_1) return;
5654     glad_glBindBufferBase = (PFNGLBINDBUFFERBASEPROC) load(userptr, "glBindBufferBase");
5655     glad_glBindBufferRange = (PFNGLBINDBUFFERRANGEPROC) load(userptr, "glBindBufferRange");
5656     glad_glCopyBufferSubData = (PFNGLCOPYBUFFERSUBDATAPROC) load(userptr, "glCopyBufferSubData");
5657     glad_glDrawArraysInstanced = (PFNGLDRAWARRAYSINSTANCEDPROC) load(userptr, "glDrawArraysInstanced");
5658     glad_glDrawElementsInstanced = (PFNGLDRAWELEMENTSINSTANCEDPROC) load(userptr,
     "glDrawElementsInstanced");
5659     glad_glGetActiveUniformBlockName = (PFNGLGETACTIVEUNIFORMBLOCKNAMEPROC) load(userptr,
     "glGetActiveUniformBlockName");
5660     glad_glGetActiveUniformBlockiv = (PFNGLGETACTIVEUNIFORMBLOCKIVPROC) load(userptr,
     "glGetActiveUniformBlockiv");
5661     glad_glGetActiveUniformName = (PFNGLGETACTIVEUNIFORMNAMEPROC) load(userptr,
     "glGetActiveUniformName");
5662     glad_glGetActiveUniformsiv = (PFNGLGETACTIVEUNIFORMSIVPROC) load(userptr, "glGetActiveUniformsiv");
5663     glad_glGetIntegeri_v = (PFNGLGETINTEGERI_VPROC) load(userptr, "glGetIntegeri_v");
5664     glad_glGetUniformBlockIndex = (PFNGLGETUNIFORMBLOCKINDEXPROC) load(userptr,
     "glGetUniformBlockIndex");
5665     glad_glGetUniformIndices = (PFNGLGETUNIFORMINDICESPROC) load(userptr, "glGetUniformIndices");
5666     glad_glPrimitiveRestartIndex = (PFNGLPRIMITIVERESTARTINDEXPROC) load(userptr,
     "glPrimitiveRestartIndex");
5667     glad_glTexBuffer = (PFNGLTEXBUFFERPROC) load(userptr, "glTexBuffer");
5668     glad_glUniformBlockBinding = (PFNGLUNIFORMBLOCKBINDINGPROC) load(userptr, "glUniformBlockBinding");
5669 }
5670 static void glad_gl_load_GL_VERSION_3_2( GLADuserptrloadfunc load, void* userptr) {
5671     if(!GLAD_GL_VERSION_3_2) return;
5672     glad_glClientWaitSync = (PFNGLCLIENTWAITSYNCPROC) load(userptr, "glClientWaitSync");
5673     glad_glDeleteSync = (PFNGLDELETESYNCPROC) load(userptr, "glDeleteSync");
5674     glad_glDrawElementsBaseVertex = (PFNGLDRAWELEMENTSBASEVERTEXPROC) load(userptr,
     "glDrawElementsBaseVertex");
5675     glad_glDrawElementsInstancedBaseVertex = (PFNGLDRAWELEMENTSINSTANCEDBASEVERTEXPROC) load(userptr,
     "glDrawElementsInstancedBaseVertex");
5676     glad_glDrawRangeElementsBaseVertex = (PFNGLDRAWRANGEELEMENTSBASEVERTEXPROC) load(userptr,
     "glDrawRangeElementsBaseVertex");
5677     glad_glFenceSync = (PFNGLFENCESYNCPROC) load(userptr, "glFenceSync");
5678     glad_glFramebufferTexture = (PFNGLFRAMEBUFFERTEXTUREPROC) load(userptr, "glFramebufferTexture");
5679     glad_glGetBufferParameteri64v = (PFNGLGETBUFFERPARAMETERI64VPROC) load(userptr,
     "glGetBufferParameteri64v");
5680     glad_glGetInteger64i_v = (PFNGLGETINTEGER64I_VPROC) load(userptr, "glGetInteger64i_v");
5681     glad_glGetInteger64v = (PFNGLGETINTEGER64VPROC) load(userptr, "glGetInteger64v");
5682     glad_glGetMultisamplefv = (PFNGLGETMULTISAMPLEFVPROC) load(userptr, "glGetMultisamplefv");
5683     glad_glGetSynciv = (PFNGLGETSYNCIVPROC) load(userptr, "glGetSynciv");
5684     glad_glIsSync = (PFNGLISSYNCPROC) load(userptr, "glIsSync");
5685     glad_glMultiDrawElementsBaseVertex = (PFNGLMULTIDRAWELEMENTSBASEVERTEXPROC) load(userptr,
     "glMultiDrawElementsBaseVertex");
5686     glad_glProvokingVertex = (PFNGLPROVOKINGVERTEXPROC) load(userptr, "glProvokingVertex");
5687     glad_glSampleMaski = (PFNGLSAMPLEMASKIPROC) load(userptr, "glSampleMaski");
5688     glad_glTexImage2DMultisample = (PFNGLTEXIMAGE2DMULTISAMPLEPROC) load(userptr,
     "glTexImage2DMultisample");
5689     glad_glTexImage3DMultisample = (PFNGLTEXIMAGE3DMULTISAMPLEPROC) load(userptr,
     "glTexImage3DMultisample");
5690     glad_glWaitSync = (PFNGLWAITSYNCPROC) load(userptr, "glWaitSync");
```

```
5691 }
5692 static void glad_gl_load_GL_VERSION_3_3( GLADuserptrloadfunc load, void* userptr) {
5693     if(!GLAD_GL_VERSION_3_3) return;
5694     glad_glBindFragDataLocationIndexed = (PFNGLBINDFRAGDATALOCATIONINDEXEDPROC) load(userptr,
      "glBindFragDataLocationIndexed");
5695     glad_glBindSampler = (PFNGLBINDSAMPLERPROC) load(userptr, "glBindSampler");
5696     glad_glColorP3ui = (PFNGLCOLORP3UIPROC) load(userptr, "glColorP3ui");
5697     glad_glColorP3uiv = (PFNGLCOLORP3UIVPROC) load(userptr, "glColorP3uiv");
5698     glad_glColorP4ui = (PFNGLCOLORP4UIPROC) load(userptr, "glColorP4ui");
5699     glad_glColorP4uiv = (PFNGLCOLORP4UIVPROC) load(userptr, "glColorP4uiv");
5700     glad_glDeleteSamplers = (PFNGLDELETESAMPLERSPROC) load(userptr, "glDeleteSamplers");
5701     glad_glGenSamplers = (PFNGLGENSAMPLERSPROC) load(userptr, "glGenSamplers");
5702     glad_glGetFragDataIndex = (PFNGLGETFRAGDATAINDEXPROC) load(userptr, "glGetFragDataIndex");
5703     glad_glGetQueryObjecti64v = (PFNGLGETQUERYOBJECTI64VPROC) load(userptr, "glGetQueryObjecti64v");
5704     glad_glGetQueryObjectui64v = (PFNGLGETQUERYOBJECTUI64VPROC) load(userptr, "glGetQueryObjectui64v");
5705     glad_glGetSamplerParameterIiv = (PFNGLGETSAMPLERPARAMETERIIVPROC) load(userptr,
      "glGetSamplerParameterIiv");
5706     glad_glGetSamplerParameterIuiv = (PFNGLGETSAMPLERPARAMETERIUIVPROC) load(userptr,
      "glGetSamplerParameterIuiv");
5707     glad_glGetSamplerParameterfv = (PFNGLGETSAMPLERPARAMETERFVPROC) load(userptr,
      "glGetSamplerParameterfv");
5708     glad_glGetSamplerParameteriv = (PFNGLGETSAMPLERPARAMETERIVPROC) load(userptr,
      "glGetSamplerParameteriv");
5709     glad_glIsSampler = (PFNGLISSAMPLERPROC) load(userptr, "glIsSampler");
5710     glad_glMultiTexCoordP1ui = (PFNGLMULTITEXCOORDP1UIPROC) load(userptr, "glMultiTexCoordP1ui");
5711     glad_glMultiTexCoordP1uiv = (PFNGLMULTITEXCOORDP1UIVPROC) load(userptr, "glMultiTexCoordP1uiv");
5712     glad_glMultiTexCoordP2ui = (PFNGLMULTITEXCOORDP2UIPROC) load(userptr, "glMultiTexCoordP2ui");
5713     glad_glMultiTexCoordP2uiv = (PFNGLMULTITEXCOORDP2UIVPROC) load(userptr, "glMultiTexCoordP2uiv");
5714     glad_glMultiTexCoordP3ui = (PFNGLMULTITEXCOORDP3UIPROC) load(userptr, "glMultiTexCoordP3ui");
5715     glad_glMultiTexCoordP3uiv = (PFNGLMULTITEXCOORDP3UIVPROC) load(userptr, "glMultiTexCoordP3uiv");
5716     glad_glMultiTexCoordP4ui = (PFNGLMULTITEXCOORDP4UIPROC) load(userptr, "glMultiTexCoordP4ui");
5717     glad_glMultiTexCoordP4uiv = (PFNGLMULTITEXCOORDP4UIVPROC) load(userptr, "glMultiTexCoordP4uiv");
5718     glad_glNormalP3ui = (PFNGLNORMALP3UIPROC) load(userptr, "glNormalP3ui");
5719     glad_glNormalP3uiv = (PFNGLNORMALP3UIVPROC) load(userptr, "glNormalP3uiv");
5720     glad_glQueryCounter = (PFNGLQUERYCOUNTERPROC) load(userptr, "glQueryCounter");
5721     glad_glSamplerParameterIiv = (PFNGLSAMPLERPARAMETERIIVPROC) load(userptr, "glSamplerParameterIiv");
5722     glad_glSamplerParameterIuiv = (PFNGLSAMPLERPARAMETERIUIVPROC) load(userptr,
      "glSamplerParameterIuiv");
5723     glad_glSamplerParameterf = (PFNGLSAMPLERPARAMETERFPROC) load(userptr, "glSamplerParameterf");
5724     glad_glSamplerParameterfv = (PFNGLSAMPLERPARAMETERFVPROC) load(userptr, "glSamplerParameterfv");
5725     glad_glSamplerParameteri = (PFNGLSAMPLERPARAMETERIPROC) load(userptr, "glSamplerParameteri");
5726     glad_glSamplerParameteriv = (PFNGLSAMPLERPARAMETERIVPROC) load(userptr, "glSamplerParameteriv");
5727     glad_glSecondaryColorP3ui = (PFNGLSECONDARYCOLORP3UIPROC) load(userptr, "glSecondaryColorP3ui");
5728     glad_glSecondaryColorP3uiv = (PFNGLSECONDARYCOLORP3UIVPROC) load(userptr, "glSecondaryColorP3uiv");
5729     glad_glTexCoordP1ui = (PFNGLTEXCOORDP1UIPROC) load(userptr, "glTexCoordP1ui");
5730     glad_glTexCoordP1uiv = (PFNGLTEXCOORDP1UIVPROC) load(userptr, "glTexCoordP1uiv");
5731     glad_glTexCoordP2ui = (PFNGLTEXCOORDP2UIPROC) load(userptr, "glTexCoordP2ui");
5732     glad_glTexCoordP2uiv = (PFNGLTEXCOORDP2UIVPROC) load(userptr, "glTexCoordP2uiv");
5733     glad_glTexCoordP3ui = (PFNGLTEXCOORDP3UIPROC) load(userptr, "glTexCoordP3ui");
5734     glad_glTexCoordP3uiv = (PFNGLTEXCOORDP3UIVPROC) load(userptr, "glTexCoordP3uiv");
5735     glad_glTexCoordP4ui = (PFNGLTEXCOORDP4UIPROC) load(userptr, "glTexCoordP4ui");
5736     glad_glTexCoordP4uiv = (PFNGLTEXCOORDP4UIVPROC) load(userptr, "glTexCoordP4uiv");
5737     glad_glVertexAttribDivisor = (PFNGLVERTEXATTRIBDIVISORPROC) load(userptr, "glVertexAttribDivisor");
5738     glad_glVertexAttribP1ui = (PFNGLVERTEXATTRIBP1UIPROC) load(userptr, "glVertexAttribP1ui");
5739     glad_glVertexAttribP1uiv = (PFNGLVERTEXATTRIBP1UIVPROC) load(userptr, "glVertexAttribP1uiv");
5740     glad_glVertexAttribP2ui = (PFNGLVERTEXATTRIBP2UIPROC) load(userptr, "glVertexAttribP2ui");
5741     glad_glVertexAttribP2uiv = (PFNGLVERTEXATTRIBP2UIVPROC) load(userptr, "glVertexAttribP2uiv");
5742     glad_glVertexAttribP3ui = (PFNGLVERTEXATTRIBP3UIPROC) load(userptr, "glVertexAttribP3ui");
5743     glad_glVertexAttribP3uiv = (PFNGLVERTEXATTRIBP3UIVPROC) load(userptr, "glVertexAttribP3uiv");
5744     glad_glVertexAttribP4ui = (PFNGLVERTEXATTRIBP4UIPROC) load(userptr, "glVertexAttribP4ui");
5745     glad_glVertexAttribP4uiv = (PFNGLVERTEXATTRIBP4UIVPROC) load(userptr, "glVertexAttribP4uiv");
5746     glad_glVertexP2ui = (PFNGLVERTEXP2UIPROC) load(userptr, "glVertexP2ui");
5747     glad_glVertexP2uiv = (PFNGLVERTEXP2UIVPROC) load(userptr, "glVertexP2uiv");
5748     glad_glVertexP3ui = (PFNGLVERTEXP3UIPROC) load(userptr, "glVertexP3ui");
5749     glad_glVertexP3uiv = (PFNGLVERTEXP3UIVPROC) load(userptr, "glVertexP3uiv");
5750     glad_glVertexP4ui = (PFNGLVERTEXP4UIPROC) load(userptr, "glVertexP4ui");
5751     glad_glVertexP4uiv = (PFNGLVERTEXP4UIVPROC) load(userptr, "glVertexP4uiv");
5752 }
5753 static void glad_gl_load_GL_ARB_multisample( GLADuserptrloadfunc load, void* userptr) {
5754     if(!GLAD_GL_ARB_multisample) return;
5755     glad_glSampleCoverageARB = (PFNGLSAMPLECOVERAGEARBPROC) load(userptr, "glSampleCoverageARB");
5756 }
5757 static void glad_gl_load_GL_ARB_robustness( GLADuserptrloadfunc load, void* userptr) {
5758     if(!GLAD_GL_ARB_robustness) return;
5759     glad_glGetGraphicsResetStatusARB = (PFNGLGETGRAPHICSRESETSTATUSARBPROC) load(userptr,
      "glGetGraphicsResetStatusARB");
5760     glad_glGetnColorTableARB = (PFNGLGETNCOLORTABLEARBPROC) load(userptr, "glGetnColorTableARB");
5761     glad_glGetnCompressedTexImageARB = (PFNGLGETNCOMPRESSEDTEXIMAGEARBPROC) load(userptr,
      "glGetnCompressedTexImageARB");
5762     glad_glGetnConvolutionFilterARB = (PFNGLGETNCONVOLUTIONFILTERARBPROC) load(userptr,
      "glGetnConvolutionFilterARB");
5763     glad_glGetnHistogramARB = (PFNGLGETNHISTOGRAMARBPROC) load(userptr, "glGetnHistogramARB");
5764     glad_glGetnMapdvARB = (PFNGLGETNMAPDVARBPROC) load(userptr, "glGetnMapdvARB");
5765     glad_glGetnMapfvARB = (PFNGLGETNMAPFVARBPROC) load(userptr, "glGetnMapfvARB");
5766     glad_glGetnMapivARB = (PFNGLGETNMAPIVARBPROC) load(userptr, "glGetnMapivARB");
5767     glad_glGetnMinmaxARB = (PFNGLGETNMINMAXARBPROC) load(userptr, "glGetnMinmaxARB");
5768     glad_glGetnPixelMapfvARB = (PFNGLGETNPIXELMAPFVARBPROC) load(userptr, "glGetnPixelMapfvARB");
```

```
5769     glad_glGetnPixelMapuivARB = (PFNGLGETNPIXELMAPUIVARBPROC) load(userptr, "glGetnPixelMapuivARB");
5770     glad_glGetnPixelMapusvARB = (PFNGLGETNPIXELMAPUSVARBPROC) load(userptr, "glGetnPixelMapusvARB");
5771     glad_glGetnPolygonStippleARB = (PFNGLGETNPOLYGONSTIPPLEARBPROC) load(userptr,
       "glGetnPolygonStippleARB");
5772     glad_glGetnSeparableFilterARB = (PFNGLGETNSEPARABLEFILTERARBPROC) load(userptr,
       "glGetnSeparableFilterARB");
5773     glad_glGetnTexImageARB = (PFNGLGETNTEXIMAGEARBPROC) load(userptr, "glGetnTexImageARB");
5774     glad_glGetnUniformdvARB = (PFNGLGETNUNIFORMDVARBPROC) load(userptr, "glGetnUniformdvARB");
5775     glad_glGetnUniformfvARB = (PFNGLGETNUNIFORMFVARBPROC) load(userptr, "glGetnUniformfvARB");
5776     glad_glGetnUniformivARB = (PFNGLGETNUNIFORMIVARBPROC) load(userptr, "glGetnUniformivARB");
5777     glad_glGetnUniformuivARB = (PFNGLGETNUNIFORMUIVARBPROC) load(userptr, "glGetnUniformuivARB");
5778     glad_glReadnPixelsARB = (PFNGLREADNPIXELSARBPROC) load(userptr, "glReadnPixelsARB");
5779 }
5780 static void glad_gl_load_GL_KHR_debug( GLADuserptrloadfunc load, void* userptr) {
5781     if(!GLAD_GL_KHR_debug) return;
5782     glad_glDebugMessageCallback = (PFNGLDEBUGMESSAGECALLBACKPROC) load(userptr,
       "glDebugMessageCallback");
5783     glad_glDebugMessageControl = (PFNGLDEBUGMESSAGECONTROLPROC) load(userptr, "glDebugMessageControl");
5784     glad_glDebugMessageInsert = (PFNGLDEBUGMESSAGEINSERTPROC) load(userptr, "glDebugMessageInsert");
5785     glad_glGetDebugMessageLog = (PFNGLGETDEBUGMESSAGELOGPROC) load(userptr, "glGetDebugMessageLog");
5786     glad_glGetObjectLabel = (PFNGLGETOBJECTLABELPROC) load(userptr, "glGetObjectLabel");
5787     glad_glGetObjectPtrLabel = (PFNGLGETOBJECTPTRLABELPROC) load(userptr, "glGetObjectPtrLabel");
5788     glad_glGetPointerv = (PFNGLGETPOINTERVPROC) load(userptr, "glGetPointerv");
5789     glad_glObjectLabel = (PFNGLOBJECTLABELPROC) load(userptr, "glObjectLabel");
5790     glad_glObjectPtrLabel = (PFNGLOBJECTPTRLABELPROC) load(userptr, "glObjectPtrLabel");
5791     glad_glPopDebugGroup = (PFNGLPOPDEBUGGROUPPROC) load(userptr, "glPopDebugGroup");
5792     glad_glPushDebugGroup = (PFNGLPUSHDEBUGGROUPPROC) load(userptr, "glPushDebugGroup");
5793 }
5794
5795
5796
5797 #if defined(GL_ES_VERSION_3_0) || defined(GL_VERSION_3_0)
5798 #define GLAD_GL_IS_SOME_NEW_VERSION 1
5799 #else
5800 #define GLAD_GL_IS_SOME_NEW_VERSION 0
5801 #endif
5802
5803 static int glad_gl_get_extensions( int version, const char **out_exts, unsigned int *out_num_exts_i,
       char ***out_exts_i) {
5804 #if GLAD_GL_IS_SOME_NEW_VERSION
5805     if(GLAD_VERSION_MAJOR(version) < 3) {
5806 #else
5807     (void) version;
5808     (void) out_num_exts_i;
5809     (void) out_exts_i;
5810 #endif
5811         if (glad_glGetString == NULL) {
5812             return 0;
5813         }
5814         *out_exts = (const char *)glad_glGetString(GL_EXTENSIONS);
5815 #if GLAD_GL_IS_SOME_NEW_VERSION
5816     } else {
5817         unsigned int index = 0;
5818         unsigned int num_exts_i = 0;
5819         char **exts_i = NULL;
5820         if (glad_glGetStringi == NULL || glad_glGetIntegerv == NULL) {
5821             return 0;
5822         }
5823         glad_glGetIntegerv(GL_NUM_EXTENSIONS, (int*) &num_exts_i);
5824         if (num_exts_i > 0) {
5825             exts_i = (char **) malloc(num_exts_i * (sizeof *exts_i));
5826         }
5827         if (exts_i == NULL) {
5828             return 0;
5829         }
5830         for(index = 0; index < num_exts_i; index++) {
5831             const char *gl_str_tmp = (const char*) glad_glGetStringi(GL_EXTENSIONS, index);
5832             size_t len = strlen(gl_str_tmp) + 1;
5833
5834             char *local_str = (char*) malloc(len * sizeof(char));
5835             if(local_str != NULL) {
5836                 memcpy(local_str, gl_str_tmp, len * sizeof(char));
5837             }
5838
5839             exts_i[index] = local_str;
5840         }
5841
5842         *out_num_exts_i = num_exts_i;
5843         *out_exts_i = exts_i;
5844     }
5845 #endif
5846     return 1;
5847 }
5848 static void glad_gl_free_extensions(char **exts_i, unsigned int num_exts_i) {
5849     if (exts_i != NULL) {
5850         unsigned int index;
5851         for(index = 0; index < num_exts_i; index++) {
```

```
5852             free((void *) (exts_i[index]));
5853         }
5854         free((void *)exts_i);
5855         exts_i = NULL;
5856     }
5857 }
5858 static int glad_gl_has_extension(int version, const char *exts, unsigned int num_exts_i, char **exts_i,
      const char *ext) {
5859     if(GLAD_VERSION_MAJOR(version) < 3 || !GLAD_GL_IS_SOME_NEW_VERSION) {
5860         const char *extensions;
5861         const char *loc;
5862         const char *terminator;
5863         extensions = exts;
5864         if(extensions == NULL || ext == NULL) {
5865             return 0;
5866         }
5867         while(1) {
5868             loc = strstr(extensions, ext);
5869             if(loc == NULL) {
5870                 return 0;
5871             }
5872             terminator = loc + strlen(ext);
5873             if((loc == extensions || *(loc - 1) == ' ') &&
5874                 (*terminator == ' ' || *terminator == '\0')) {
5875                 return 1;
5876             }
5877             extensions = terminator;
5878         }
5879     } else {
5880         unsigned int index;
5881         for(index = 0; index < num_exts_i; index++) {
5882             const char *e = exts_i[index];
5883             if(strcmp(e, ext) == 0) {
5884                 return 1;
5885             }
5886         }
5887     }
5888     return 0;
5889 }
5890
5891 static GLADapiproc glad_gl_get_proc_from_userptr(void *userptr, const char* name) {
5892     return (GLAD_GNUC_EXTENSION (GLADapiproc (*)(const char *name)) userptr)(name);
5893 }
5894
5895 static int glad_gl_find_extensions_gl( int version) {
5896     const char *exts = NULL;
5897     unsigned int num_exts_i = 0;
5898     char **exts_i = NULL;
5899     if (!glad_gl_get_extensions(version, &exts, &num_exts_i, &exts_i)) return 0;
5900
5901     GLAD_GL_ARB_multisample = glad_gl_has_extension(version, exts, num_exts_i, exts_i,
      "GL_ARB_multisample");
5902     GLAD_GL_ARB_robustness = glad_gl_has_extension(version, exts, num_exts_i, exts_i,
      "GL_ARB_robustness");
5903     GLAD_GL_KHR_debug = glad_gl_has_extension(version, exts, num_exts_i, exts_i, "GL_KHR_debug");
5904
5905     glad_gl_free_extensions(exts_i, num_exts_i);
5906
5907     return 1;
5908 }
5909
5910 static int glad_gl_find_core_gl(void) {
5911     int i;
5912     const char* version;
5913     const char* prefixes[] = {
5914         "OpenGL ES-CM ",
5915         "OpenGL ES-CL ",
5916         "OpenGL ES ",
5917         "OpenGL SC ",
5918         NULL
5919     };
5920     int major = 0;
5921     int minor = 0;
5922     version = (const char*) glad_glGetString(GL_VERSION);
5923     if (!version) return 0;
5924     for (i = 0;  prefixes[i];  i++) {
5925         const size_t length = strlen(prefixes[i]);
5926         if (strncmp(version, prefixes[i], length) == 0) {
5927             version += length;
5928             break;
5929         }
5930     }
5931
5932     GLAD_IMPL_UTIL_SSCANF(version, "%d.%d", &major, &minor);
5933
5934     GLAD_GL_VERSION_1_0 = (major == 1 && minor >= 0) || major > 1;
5935     GLAD_GL_VERSION_1_1 = (major == 1 && minor >= 1) || major > 1;
```

```
5936     GLAD_GL_VERSION_1_2 = (major == 1 && minor >= 2) || major > 1;
5937     GLAD_GL_VERSION_1_3 = (major == 1 && minor >= 3) || major > 1;
5938     GLAD_GL_VERSION_1_4 = (major == 1 && minor >= 4) || major > 1;
5939     GLAD_GL_VERSION_1_5 = (major == 1 && minor >= 5) || major > 1;
5940     GLAD_GL_VERSION_2_0 = (major == 2 && minor >= 0) || major > 2;
5941     GLAD_GL_VERSION_2_1 = (major == 2 && minor >= 1) || major > 2;
5942     GLAD_GL_VERSION_3_0 = (major == 3 && minor >= 0) || major > 3;
5943     GLAD_GL_VERSION_3_1 = (major == 3 && minor >= 1) || major > 3;
5944     GLAD_GL_VERSION_3_2 = (major == 3 && minor >= 2) || major > 3;
5945     GLAD_GL_VERSION_3_3 = (major == 3 && minor >= 3) || major > 3;
5946
5947     return GLAD_MAKE_VERSION(major, minor);
5948 }
5949
5950 int gladLoadGLUserPtr( GLADuserptrloadfunc load, void *userptr) {
5951     int version;
5952
5953     glad_glGetString = (PFNGLGETSTRINGPROC) load(userptr, "glGetString");
5954     if(glad_glGetString == NULL) return 0;
5955     if(glad_glGetString(GL_VERSION) == NULL) return 0;
5956     version = glad_gl_find_core_gl();
5957
5958     glad_gl_load_GL_VERSION_1_0(load, userptr);
5959     glad_gl_load_GL_VERSION_1_1(load, userptr);
5960     glad_gl_load_GL_VERSION_1_2(load, userptr);
5961     glad_gl_load_GL_VERSION_1_3(load, userptr);
5962     glad_gl_load_GL_VERSION_1_4(load, userptr);
5963     glad_gl_load_GL_VERSION_1_5(load, userptr);
5964     glad_gl_load_GL_VERSION_2_0(load, userptr);
5965     glad_gl_load_GL_VERSION_2_1(load, userptr);
5966     glad_gl_load_GL_VERSION_3_0(load, userptr);
5967     glad_gl_load_GL_VERSION_3_1(load, userptr);
5968     glad_gl_load_GL_VERSION_3_2(load, userptr);
5969     glad_gl_load_GL_VERSION_3_3(load, userptr);
5970
5971     if (!glad_gl_find_extensions_gl(version)) return 0;
5972     glad_gl_load_GL_ARB_multisample(load, userptr);
5973     glad_gl_load_GL_ARB_robustness(load, userptr);
5974     glad_gl_load_GL_KHR_debug(load, userptr);
5975
5976
5977
5978     return version;
5979 }
5980
5981
5982 int gladLoadGL( GLADloadfunc load) {
5983     return gladLoadGLUserPtr( glad_gl_get_proc_from_userptr, GLAD_GNUC_EXTENSION (void*) load);
5984 }
5985
5986
5987
5988
5989
5990
5991 #ifdef __cplusplus
5992 }
5993 #endif
5994
5995 #endif /* GLAD_GL_IMPLEMENTATION */
5996
```

27.3 gles2.h

```
1
28 #ifndef GLAD_GLES2_H_
29 #define GLAD_GLES2_H_
30
31 #ifdef __clang__
32 #pragma clang diagnostic push
33 #pragma clang diagnostic ignored "-Wreserved-id-macro"
34 #endif
35 #ifdef __gl2_h_
36   #error OpenGL ES 2 header already included (API: gles2), remove previous include!
37 #endif
38 #define __gl2_h_ 1
39 #ifdef __gl3_h_
40   #error OpenGL ES 3 header already included (API: gles2), remove previous include!
41 #endif
42 #define __gl3_h_ 1
43 #ifdef __clang__
44 #pragma clang diagnostic pop
45 #endif
```

```
46
47 #define GLAD_GLES2
48 #define GLAD_OPTION_GLES2_HEADER_ONLY
49
50 #ifdef __cplusplus
51 extern "C" {
52 #endif
53
54 #ifndef GLAD_PLATFORM_H_
55 #define GLAD_PLATFORM_H_
56
57 #ifndef GLAD_PLATFORM_WIN32
58   #if defined(_WIN32) || defined(__WIN32__) || defined(WIN32) || defined(__MINGW32__)
59     #define GLAD_PLATFORM_WIN32 1
60   #else
61     #define GLAD_PLATFORM_WIN32 0
62   #endif
63 #endif
64
65 #ifndef GLAD_PLATFORM_APPLE
66   #ifdef __APPLE__
67     #define GLAD_PLATFORM_APPLE 1
68   #else
69     #define GLAD_PLATFORM_APPLE 0
70   #endif
71 #endif
72
73 #ifndef GLAD_PLATFORM_EMSCRIPTEN
74   #ifdef __EMSCRIPTEN__
75     #define GLAD_PLATFORM_EMSCRIPTEN 1
76   #else
77     #define GLAD_PLATFORM_EMSCRIPTEN 0
78   #endif
79 #endif
80
81 #ifndef GLAD_PLATFORM_UWP
82   #if defined(_MSC_VER) && !defined(GLAD_INTERNAL_HAVE_WINAPIFAMILY)
83     #ifdef __has_include
84       #if __has_include(<winapifamily.h>)
85         #define GLAD_INTERNAL_HAVE_WINAPIFAMILY 1
86       #endif
87     #elif _MSC_VER >= 1700 && !_USING_V110_SDK71_
88       #define GLAD_INTERNAL_HAVE_WINAPIFAMILY 1
89     #endif
90   #endif
91
92   #ifdef GLAD_INTERNAL_HAVE_WINAPIFAMILY
93     #include <winapifamily.h>
94     #if !WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_DESKTOP) &&
       WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_APP)
95       #define GLAD_PLATFORM_UWP 1
96     #endif
97   #endif
98
99   #ifndef GLAD_PLATFORM_UWP
100    #define GLAD_PLATFORM_UWP 0
101  #endif
102 #endif
103
104 #ifdef __GNUC__
105   #define GLAD_GNUC_EXTENSION __extension__
106 #else
107   #define GLAD_GNUC_EXTENSION
108 #endif
109
110 #ifndef GLAD_API_CALL
111   #if defined(GLAD_API_CALL_EXPORT)
112     #if GLAD_PLATFORM_WIN32 || defined(__CYGWIN__)
113       #if defined(GLAD_API_CALL_EXPORT_BUILD)
114         #if defined(__GNUC__)
115           #define GLAD_API_CALL __attribute__ ((dllexport)) extern
116         #else
117           #define GLAD_API_CALL __declspec(dllexport) extern
118         #endif
119       #else
120         #if defined(__GNUC__)
121           #define GLAD_API_CALL __attribute__ ((dllimport)) extern
122         #else
123           #define GLAD_API_CALL __declspec(dllimport) extern
124         #endif
125       #endif
126     #elif defined(__GNUC__) && defined(GLAD_API_CALL_EXPORT_BUILD)
127       #define GLAD_API_CALL __attribute__ ((visibility ("default"))) extern
128     #else
129       #define GLAD_API_CALL extern
130     #endif
131   #else
```

```
132     #define GLAD_API_CALL extern
133   #endif
134 #endif
135
136 #ifdef APIENTRY
137   #define GLAD_API_PTR APIENTRY
138 #elif GLAD_PLATFORM_WIN32
139   #define GLAD_API_PTR __stdcall
140 #else
141   #define GLAD_API_PTR
142 #endif
143
144 #ifndef GLAPI
145 #define GLAPI GLAD_API_CALL
146 #endif
147
148 #ifndef GLAPIENTRY
149 #define GLAPIENTRY GLAD_API_PTR
150 #endif
151
152 #define GLAD_MAKE_VERSION(major, minor) (major * 10000 + minor)
153 #define GLAD_VERSION_MAJOR(version) (version / 10000)
154 #define GLAD_VERSION_MINOR(version) (version % 10000)
155
156 #define GLAD_GENERATOR_VERSION "2.0.0-beta"
157
158 typedef void (*GLADapiproc)(void);
159
160 typedef GLADapiproc (*GLADloadfunc)(const char *name);
161 typedef GLADapiproc (*GLADuserptrloadfunc)(void *userptr, const char *name);
162
163 typedef void (*GLADprecallback)(const char *name, GLADapiproc apiproc, int len_args, ...);
164 typedef void (*GLADpostcallback)(void *ret, const char *name, GLADapiproc apiproc, int len_args, ...);
165
166 #endif /* GLAD_PLATFORM_H_ */
167
168 #define GL_ACTIVE_ATTRIBUTES 0x8B89
169 #define GL_ACTIVE_ATTRIBUTE_MAX_LENGTH 0x8B8A
170 #define GL_ACTIVE_TEXTURE 0x84E0
171 #define GL_ACTIVE_UNIFORMS 0x8B86
172 #define GL_ACTIVE_UNIFORM_MAX_LENGTH 0x8B87
173 #define GL_ALIASED_LINE_WIDTH_RANGE 0x846E
174 #define GL_ALIASED_POINT_SIZE_RANGE 0x846D
175 #define GL_ALPHA 0x1906
176 #define GL_ALPHA_BITS 0x0D55
177 #define GL_ALWAYS 0x0207
178 #define GL_ARRAY_BUFFER 0x8892
179 #define GL_ARRAY_BUFFER_BINDING 0x8894
180 #define GL_ATTACHED_SHADERS 0x8B85
181 #define GL_BACK 0x0405
182 #define GL_BLEND 0x0BE2
183 #define GL_BLEND_COLOR 0x8005
184 #define GL_BLEND_DST_ALPHA 0x80CA
185 #define GL_BLEND_DST_RGB 0x80C8
186 #define GL_BLEND_EQUATION 0x8009
187 #define GL_BLEND_EQUATION_ALPHA 0x883D
188 #define GL_BLEND_EQUATION_RGB 0x8009
189 #define GL_BLEND_SRC_ALPHA 0x80CB
190 #define GL_BLEND_SRC_RGB 0x80C9
191 #define GL_BLUE_BITS 0x0D54
192 #define GL_BOOL 0x8B56
193 #define GL_BOOL_VEC2 0x8B57
194 #define GL_BOOL_VEC3 0x8B58
195 #define GL_BOOL_VEC4 0x8B59
196 #define GL_BUFFER_SIZE 0x8764
197 #define GL_BUFFER_USAGE 0x8765
198 #define GL_BYTE 0x1400
199 #define GL_CCW 0x0901
200 #define GL_CLAMP_TO_EDGE 0x812F
201 #define GL_COLOR_ATTACHMENT0 0x8CE0
202 #define GL_COLOR_BUFFER_BIT 0x00004000
203 #define GL_COLOR_CLEAR_VALUE 0x0C22
204 #define GL_COLOR_WRITEMASK 0x0C23
205 #define GL_COMPILE_STATUS 0x8B81
206 #define GL_COMPRESSED_TEXTURE_FORMATS 0x86A3
207 #define GL_CONSTANT_ALPHA 0x8003
208 #define GL_CONSTANT_COLOR 0x8001
209 #define GL_CULL_FACE 0x0B44
210 #define GL_CULL_FACE_MODE 0x0B45
211 #define GL_CURRENT_PROGRAM 0x8B8D
212 #define GL_CURRENT_VERTEX_ATTRIB 0x8626
213 #define GL_CW 0x0900
214 #define GL_DECR 0x1E03
215 #define GL_DECR_WRAP 0x8508
216 #define GL_DELETE_STATUS 0x8B80
217 #define GL_DEPTH_ATTACHMENT 0x8D00
218 #define GL_DEPTH_BITS 0x0D56
```

```
219 #define GL_DEPTH_BUFFER_BIT 0x00000100
220 #define GL_DEPTH_CLEAR_VALUE 0x0B73
221 #define GL_DEPTH_COMPONENT 0x1902
222 #define GL_DEPTH_COMPONENT16 0x81A5
223 #define GL_DEPTH_FUNC 0x0B74
224 #define GL_DEPTH_RANGE 0x0B70
225 #define GL_DEPTH_TEST 0x0B71
226 #define GL_DEPTH_WRITEMASK 0x0B72
227 #define GL_DITHER 0x0BD0
228 #define GL_DONT_CARE 0x1100
229 #define GL_DST_ALPHA 0x0304
230 #define GL_DST_COLOR 0x0306
231 #define GL_DYNAMIC_DRAW 0x88E8
232 #define GL_ELEMENT_ARRAY_BUFFER 0x8893
233 #define GL_ELEMENT_ARRAY_BUFFER_BINDING 0x8895
234 #define GL_EQUAL 0x0202
235 #define GL_EXTENSIONS 0x1F03
236 #define GL_FALSE 0
237 #define GL_FASTEST 0x1101
238 #define GL_FIXED 0x140C
239 #define GL_FLOAT 0x1406
240 #define GL_FLOAT_MAT2 0x8B5A
241 #define GL_FLOAT_MAT3 0x8B5B
242 #define GL_FLOAT_MAT4 0x8B5C
243 #define GL_FLOAT_VEC2 0x8B50
244 #define GL_FLOAT_VEC3 0x8B51
245 #define GL_FLOAT_VEC4 0x8B52
246 #define GL_FRAGMENT_SHADER 0x8B30
247 #define GL_FRAMEBUFFER 0x8D40
248 #define GL_FRAMEBUFFER_ATTACHMENT_OBJECT_NAME 0x8CD1
249 #define GL_FRAMEBUFFER_ATTACHMENT_OBJECT_TYPE 0x8CD0
250 #define GL_FRAMEBUFFER_ATTACHMENT_TEXTURE_CUBE_MAP_FACE 0x8CD3
251 #define GL_FRAMEBUFFER_ATTACHMENT_TEXTURE_LEVEL 0x8CD2
252 #define GL_FRAMEBUFFER_BINDING 0x8CA6
253 #define GL_FRAMEBUFFER_COMPLETE 0x8CD5
254 #define GL_FRAMEBUFFER_INCOMPLETE_ATTACHMENT 0x8CD6
255 #define GL_FRAMEBUFFER_INCOMPLETE_DIMENSIONS 0x8CD9
256 #define GL_FRAMEBUFFER_INCOMPLETE_MISSING_ATTACHMENT 0x8CD7
257 #define GL_FRAMEBUFFER_UNSUPPORTED 0x8CDD
258 #define GL_FRONT 0x0404
259 #define GL_FRONT_AND_BACK 0x0408
260 #define GL_FRONT_FACE 0x0B46
261 #define GL_FUNC_ADD 0x8006
262 #define GL_FUNC_REVERSE_SUBTRACT 0x800B
263 #define GL_FUNC_SUBTRACT 0x800A
264 #define GL_GENERATE_MIPMAP_HINT 0x8192
265 #define GL_GEQUAL 0x0206
266 #define GL_GREATER 0x0204
267 #define GL_GREEN_BITS 0x0D53
268 #define GL_HIGH_FLOAT 0x8DF2
269 #define GL_HIGH_INT 0x8DF5
270 #define GL_IMPLEMENTATION_COLOR_READ_FORMAT 0x8B9B
271 #define GL_IMPLEMENTATION_COLOR_READ_TYPE 0x8B9A
272 #define GL_INCR 0x1E02
273 #define GL_INCR_WRAP 0x8507
274 #define GL_INFO_LOG_LENGTH 0x8B84
275 #define GL_INT 0x1404
276 #define GL_INT_VEC2 0x8B53
277 #define GL_INT_VEC3 0x8B54
278 #define GL_INT_VEC4 0x8B55
279 #define GL_INVALID_ENUM 0x0500
280 #define GL_INVALID_FRAMEBUFFER_OPERATION 0x0506
281 #define GL_INVALID_OPERATION 0x0502
282 #define GL_INVALID_VALUE 0x0501
283 #define GL_INVERT 0x150A
284 #define GL_KEEP 0x1E00
285 #define GL_LEQUAL 0x0203
286 #define GL_LESS 0x0201
287 #define GL_LINEAR 0x2601
288 #define GL_LINEAR_MIPMAP_LINEAR 0x2703
289 #define GL_LINEAR_MIPMAP_NEAREST 0x2701
290 #define GL_LINES 0x0001
291 #define GL_LINE_LOOP 0x0002
292 #define GL_LINE_STRIP 0x0003
293 #define GL_LINE_WIDTH 0x0B21
294 #define GL_LINK_STATUS 0x8B82
295 #define GL_LOW_FLOAT 0x8DF0
296 #define GL_LOW_INT 0x8DF3
297 #define GL_LUMINANCE 0x1909
298 #define GL_LUMINANCE_ALPHA 0x190A
299 #define GL_MAX_COMBINED_TEXTURE_IMAGE_UNITS 0x8B4D
300 #define GL_MAX_CUBE_MAP_TEXTURE_SIZE 0x851C
301 #define GL_MAX_FRAGMENT_UNIFORM_VECTORS 0x8DFD
302 #define GL_MAX_RENDERBUFFER_SIZE 0x84E8
303 #define GL_MAX_TEXTURE_IMAGE_UNITS 0x8872
304 #define GL_MAX_TEXTURE_SIZE 0x0D33
305 #define GL_MAX_VARYING_VECTORS 0x8DFC
```

```
306 #define GL_MAX_VERTEX_ATTRIBS 0x8869
307 #define GL_MAX_VERTEX_TEXTURE_IMAGE_UNITS 0x8B4C
308 #define GL_MAX_VERTEX_UNIFORM_VECTORS 0x8DFB
309 #define GL_MAX_VIEWPORT_DIMS 0x0D3A
310 #define GL_MEDIUM_FLOAT 0x8DF1
311 #define GL_MEDIUM_INT 0x8DF4
312 #define GL_MIRRORED_REPEAT 0x8370
313 #define GL_NEAREST 0x2600
314 #define GL_NEAREST_MIPMAP_LINEAR 0x2702
315 #define GL_NEAREST_MIPMAP_NEAREST 0x2700
316 #define GL_NEVER 0x0200
317 #define GL_NICEST 0x1102
318 #define GL_NONE 0
319 #define GL_NOTEQUAL 0x0205
320 #define GL_NO_ERROR 0
321 #define GL_NUM_COMPRESSED_TEXTURE_FORMATS 0x86A2
322 #define GL_NUM_SHADER_BINARY_FORMATS 0x8DF9
323 #define GL_ONE 1
324 #define GL_ONE_MINUS_CONSTANT_ALPHA 0x8004
325 #define GL_ONE_MINUS_CONSTANT_COLOR 0x8002
326 #define GL_ONE_MINUS_DST_ALPHA 0x0305
327 #define GL_ONE_MINUS_DST_COLOR 0x0307
328 #define GL_ONE_MINUS_SRC_ALPHA 0x0303
329 #define GL_ONE_MINUS_SRC_COLOR 0x0301
330 #define GL_OUT_OF_MEMORY 0x0505
331 #define GL_PACK_ALIGNMENT 0x0D05
332 #define GL_POINTS 0x0000
333 #define GL_POLYGON_OFFSET_FACTOR 0x8038
334 #define GL_POLYGON_OFFSET_FILL 0x8037
335 #define GL_POLYGON_OFFSET_UNITS 0x2A00
336 #define GL_RED_BITS 0x0D52
337 #define GL_RENDERBUFFER 0x8D41
338 #define GL_RENDERBUFFER_ALPHA_SIZE 0x8D53
339 #define GL_RENDERBUFFER_BINDING 0x8CA7
340 #define GL_RENDERBUFFER_BLUE_SIZE 0x8D52
341 #define GL_RENDERBUFFER_DEPTH_SIZE 0x8D54
342 #define GL_RENDERBUFFER_GREEN_SIZE 0x8D51
343 #define GL_RENDERBUFFER_HEIGHT 0x8D43
344 #define GL_RENDERBUFFER_INTERNAL_FORMAT 0x8D44
345 #define GL_RENDERBUFFER_RED_SIZE 0x8D50
346 #define GL_RENDERBUFFER_STENCIL_SIZE 0x8D55
347 #define GL_RENDERBUFFER_WIDTH 0x8D42
348 #define GL_RENDERER 0x1F01
349 #define GL_REPEAT 0x2901
350 #define GL_REPLACE 0x1E01
351 #define GL_RGB 0x1907
352 #define GL_RGB565 0x8D62
353 #define GL_RGB5_A1 0x8057
354 #define GL_RGBA 0x1908
355 #define GL_RGBA4 0x8056
356 #define GL_SAMPLER_2D 0x8B5E
357 #define GL_SAMPLER_CUBE 0x8B60
358 #define GL_SAMPLES 0x80A9
359 #define GL_SAMPLE_ALPHA_TO_COVERAGE 0x809E
360 #define GL_SAMPLE_BUFFERS 0x80A8
361 #define GL_SAMPLE_COVERAGE 0x80A0
362 #define GL_SAMPLE_COVERAGE_INVERT 0x80AB
363 #define GL_SAMPLE_COVERAGE_VALUE 0x80AA
364 #define GL_SCISSOR_BOX 0x0C10
365 #define GL_SCISSOR_TEST 0x0C11
366 #define GL_SHADER_BINARY_FORMATS 0x8DF8
367 #define GL_SHADER_COMPILER 0x8DFA
368 #define GL_SHADER_SOURCE_LENGTH 0x8B88
369 #define GL_SHADER_TYPE 0x8B4F
370 #define GL_SHADING_LANGUAGE_VERSION 0x8B8C
371 #define GL_SHORT 0x1402
372 #define GL_SRC_ALPHA 0x0302
373 #define GL_SRC_ALPHA_SATURATE 0x0308
374 #define GL_SRC_COLOR 0x0300
375 #define GL_STATIC_DRAW 0x88E4
376 #define GL_STENCIL_ATTACHMENT 0x8D20
377 #define GL_STENCIL_BACK_FAIL 0x8801
378 #define GL_STENCIL_BACK_FUNC 0x8800
379 #define GL_STENCIL_BACK_PASS_DEPTH_FAIL 0x8802
380 #define GL_STENCIL_BACK_PASS_DEPTH_PASS 0x8803
381 #define GL_STENCIL_BACK_REF 0x8CA3
382 #define GL_STENCIL_BACK_VALUE_MASK 0x8CA4
383 #define GL_STENCIL_BACK_WRITEMASK 0x8CA5
384 #define GL_STENCIL_BITS 0x0D57
385 #define GL_STENCIL_BUFFER_BIT 0x00000400
386 #define GL_STENCIL_CLEAR_VALUE 0x0B91
387 #define GL_STENCIL_FAIL 0x0B94
388 #define GL_STENCIL_FUNC 0x0B92
389 #define GL_STENCIL_INDEX8 0x8D48
390 #define GL_STENCIL_PASS_DEPTH_FAIL 0x0B95
391 #define GL_STENCIL_PASS_DEPTH_PASS 0x0B96
392 #define GL_STENCIL_REF 0x0B97
```

```
393 #define GL_STENCIL_TEST 0x0B90
394 #define GL_STENCIL_VALUE_MASK 0x0B93
395 #define GL_STENCIL_WRITEMASK 0x0B98
396 #define GL_STREAM_DRAW 0x88E0
397 #define GL_SUBPIXEL_BITS 0x0D50
398 #define GL_TEXTURE 0x1702
399 #define GL_TEXTURE0 0x84C0
400 #define GL_TEXTURE1 0x84C1
401 #define GL_TEXTURE10 0x84CA
402 #define GL_TEXTURE11 0x84CB
403 #define GL_TEXTURE12 0x84CC
404 #define GL_TEXTURE13 0x84CD
405 #define GL_TEXTURE14 0x84CE
406 #define GL_TEXTURE15 0x84CF
407 #define GL_TEXTURE16 0x84D0
408 #define GL_TEXTURE17 0x84D1
409 #define GL_TEXTURE18 0x84D2
410 #define GL_TEXTURE19 0x84D3
411 #define GL_TEXTURE2 0x84C2
412 #define GL_TEXTURE20 0x84D4
413 #define GL_TEXTURE21 0x84D5
414 #define GL_TEXTURE22 0x84D6
415 #define GL_TEXTURE23 0x84D7
416 #define GL_TEXTURE24 0x84D8
417 #define GL_TEXTURE25 0x84D9
418 #define GL_TEXTURE26 0x84DA
419 #define GL_TEXTURE27 0x84DB
420 #define GL_TEXTURE28 0x84DC
421 #define GL_TEXTURE29 0x84DD
422 #define GL_TEXTURE3 0x84C3
423 #define GL_TEXTURE30 0x84DE
424 #define GL_TEXTURE31 0x84DF
425 #define GL_TEXTURE4 0x84C4
426 #define GL_TEXTURE5 0x84C5
427 #define GL_TEXTURE6 0x84C6
428 #define GL_TEXTURE7 0x84C7
429 #define GL_TEXTURE8 0x84C8
430 #define GL_TEXTURE9 0x84C9
431 #define GL_TEXTURE_2D 0x0DE1
432 #define GL_TEXTURE_BINDING_2D 0x8069
433 #define GL_TEXTURE_BINDING_CUBE_MAP 0x8514
434 #define GL_TEXTURE_CUBE_MAP 0x8513
435 #define GL_TEXTURE_CUBE_MAP_NEGATIVE_X 0x8516
436 #define GL_TEXTURE_CUBE_MAP_NEGATIVE_Y 0x8518
437 #define GL_TEXTURE_CUBE_MAP_NEGATIVE_Z 0x851A
438 #define GL_TEXTURE_CUBE_MAP_POSITIVE_X 0x8515
439 #define GL_TEXTURE_CUBE_MAP_POSITIVE_Y 0x8517
440 #define GL_TEXTURE_CUBE_MAP_POSITIVE_Z 0x8519
441 #define GL_TEXTURE_MAG_FILTER 0x2800
442 #define GL_TEXTURE_MIN_FILTER 0x2801
443 #define GL_TEXTURE_WRAP_S 0x2802
444 #define GL_TEXTURE_WRAP_T 0x2803
445 #define GL_TRIANGLES 0x0004
446 #define GL_TRIANGLE_FAN 0x0006
447 #define GL_TRIANGLE_STRIP 0x0005
448 #define GL_TRUE 1
449 #define GL_UNPACK_ALIGNMENT 0x0CF5
450 #define GL_UNSIGNED_BYTE 0x1401
451 #define GL_UNSIGNED_INT 0x1405
452 #define GL_UNSIGNED_SHORT 0x1403
453 #define GL_UNSIGNED_SHORT_4_4_4_4 0x8033
454 #define GL_UNSIGNED_SHORT_5_5_5_1 0x8034
455 #define GL_UNSIGNED_SHORT_5_6_5 0x8363
456 #define GL_VALIDATE_STATUS 0x8B83
457 #define GL_VENDOR 0x1F00
458 #define GL_VERSION 0x1F02
459 #define GL_VERTEX_ATTRIB_ARRAY_BUFFER_BINDING 0x889F
460 #define GL_VERTEX_ATTRIB_ARRAY_ENABLED 0x8622
461 #define GL_VERTEX_ATTRIB_ARRAY_NORMALIZED 0x886A
462 #define GL_VERTEX_ATTRIB_ARRAY_POINTER 0x8645
463 #define GL_VERTEX_ATTRIB_ARRAY_SIZE 0x8623
464 #define GL_VERTEX_ATTRIB_ARRAY_STRIDE 0x8624
465 #define GL_VERTEX_ATTRIB_ARRAY_TYPE 0x8625
466 #define GL_VERTEX_SHADER 0x8B31
467 #define GL_VIEWPORT 0x0BA2
468 #define GL_ZERO 0
469
470
471 #ifndef __khrplatform_h_
472 #define __khrplatform_h_
473
474 /*
475 ** Copyright (c) 2008-2018 The Khronos Group Inc.
476 **
477 ** Permission is hereby granted, free of charge, to any person obtaining a
478 ** copy of this software and/or associated documentation files (the
479 ** "Materials"), to deal in the Materials without restriction, including
```

```
** without limitation the rights to use, copy, modify, merge, publish,
** distribute, sublicense, and/or sell copies of the Materials, and to
** permit persons to whom the Materials are 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 Materials.
**
** THE MATERIALS ARE 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
** MATERIALS OR THE USE OR OTHER DEALINGS IN THE MATERIALS.
*/

/* Khronos platform-specific types and definitions.
 *
 * The master copy of khrplatform.h is maintained in the Khronos EGL
 * Registry repository at https://github.com/KhronosGroup/EGL-Registry
 * The last semantic modification to khrplatform.h was at commit ID:
 *      67a3e0864c2d75ea5287b9f3d2eb74a745936692
 *
 * Adopters may modify this file to suit their platform. Adopters are
 * encouraged to submit platform specific modifications to the Khronos
 * group so that they can be included in future versions of this file.
 * Please submit changes by filing pull requests or issues on
 * the EGL Registry repository linked above.
 *
 *
 * See the Implementer's Guidelines for information about where this file
 * should be located on your system and for more details of its use:
 *    http://www.khronos.org/registry/implementers_guide.pdf
 *
 * This file should be included as
 *        #include <KHR/khrplatform.h>
 * by Khronos client API header files that use its types and defines.
 *
 * The types in khrplatform.h should only be used to define API-specific types.
 *
 * Types defined in khrplatform.h:
 *    khronos_int8_t              signed   8  bit
 *    khronos_uint8_t             unsigned 8  bit
 *    khronos_int16_t             signed   16 bit
 *    khronos_uint16_t            unsigned 16 bit
 *    khronos_int32_t             signed   32 bit
 *    khronos_uint32_t            unsigned 32 bit
 *    khronos_int64_t             signed   64 bit
 *    khronos_uint64_t            unsigned 64 bit
 *    khronos_intptr_t            signed   same number of bits as a pointer
 *    khronos_uintptr_t           unsigned same number of bits as a pointer
 *    khronos_ssize_t             signed   size
 *    khronos_usize_t             unsigned size
 *    khronos_float_t             signed   32 bit floating point
 *    khronos_time_ns_t           unsigned 64 bit time in nanoseconds
 *    khronos_utime_nanoseconds_t unsigned time interval or absolute time in
 *                                         nanoseconds
 *    khronos_stime_nanoseconds_t signed time interval in nanoseconds
 *    khronos_boolean_enum_t      enumerated boolean type. This should
 *      only be used as a base type when a client API's boolean type is
 *      an enum. Client APIs which use an integer or other type for
 *      booleans cannot use this as the base type for their boolean.
 *
 * Tokens defined in khrplatform.h:
 *
 *    KHRONOS_FALSE, KHRONOS_TRUE Enumerated boolean false/true values.
 *
 *    KHRONOS_SUPPORT_INT64 is 1 if 64 bit integers are supported; otherwise 0.
 *    KHRONOS_SUPPORT_FLOAT is 1 if floats are supported; otherwise 0.
 *
 * Calling convention macros defined in this file:
 *    KHRONOS_APICALL
 *    KHRONOS_GLAD_API_PTR
 *    KHRONOS_APIATTRIBUTES
 *
 * These may be used in function prototypes as:
 *
 *      KHRONOS_APICALL void KHRONOS_GLAD_API_PTR funcname(
 *                                  int arg1,
 *                                  int arg2) KHRONOS_APIATTRIBUTES;
 */

#if defined(__SCITECH_SNAP__) && !defined(KHRONOS_STATIC)
#   define KHRONOS_STATIC 1
#endif

```

```
567 /*-------------------------------------------------------------------------
568  * Definition of KHRONOS_APICALL
569  *-------------------------------------------------------------------------
570  * This precedes the return type of the function in the function prototype.
571  */
572 #if defined(KHRONOS_STATIC)
573     /* If the preprocessor constant KHRONOS_STATIC is defined, make the
574      * header compatible with static linking. */
575 #   define KHRONOS_APICALL
576 #elif defined(_WIN32)
577 #   define KHRONOS_APICALL __declspec(dllimport)
578 #elif defined (__SYMBIAN32__)
579 #   define KHRONOS_APICALL IMPORT_C
580 #elif defined(__ANDROID__)
581 #   define KHRONOS_APICALL __attribute__((visibility("default")))
582 #else
583 #   define KHRONOS_APICALL
584 #endif
585
586 /*-------------------------------------------------------------------------
587  * Definition of KHRONOS_GLAD_API_PTR
588  *-------------------------------------------------------------------------
589  * This follows the return type of the function  and precedes the function
590  * name in the function prototype.
591  */
592 #if defined(_WIN32) && !defined(_WIN32_WCE) && !defined(__SCITECH_SNAP__)
593     /* Win32 but not WinCE */
594 #   define KHRONOS_GLAD_API_PTR __stdcall
595 #else
596 #   define KHRONOS_GLAD_API_PTR
597 #endif
598
599 /*-------------------------------------------------------------------------
600  * Definition of KHRONOS_APIATTRIBUTES
601  *-------------------------------------------------------------------------
602  * This follows the closing parenthesis of the function prototype arguments.
603  */
604 #if defined (__ARMCC_2__)
605 #define KHRONOS_APIATTRIBUTES __softfp
606 #else
607 #define KHRONOS_APIATTRIBUTES
608 #endif
609
610 /*-------------------------------------------------------------------------
611  * basic type definitions
612  *-----------------------------------------------------------------------*/
613 #if (defined(__STDC_VERSION__) && __STDC_VERSION__ >= 199901L) || defined(__GNUC__) || defined(__SCO__)
      || defined(__USLC__)
614
615
616 /*
617  * Using <stdint.h>
618  */
619 #include <stdint.h>
620 typedef int32_t                 khronos_int32_t;
621 typedef uint32_t                khronos_uint32_t;
622 typedef int64_t                 khronos_int64_t;
623 typedef uint64_t                khronos_uint64_t;
624 #define KHRONOS_SUPPORT_INT64   1
625 #define KHRONOS_SUPPORT_FLOAT   1
626
627 #elif defined(__VMS ) || defined(__sgi)
628
629 /*
630  * Using <inttypes.h>
631  */
632 #include <inttypes.h>
633 typedef int32_t                 khronos_int32_t;
634 typedef uint32_t                khronos_uint32_t;
635 typedef int64_t                 khronos_int64_t;
636 typedef uint64_t                khronos_uint64_t;
637 #define KHRONOS_SUPPORT_INT64   1
638 #define KHRONOS_SUPPORT_FLOAT   1
639
640 #elif defined(_WIN32) && !defined(__SCITECH_SNAP__)
641
642 /*
643  * Win32
644  */
645 typedef __int32                 khronos_int32_t;
646 typedef unsigned __int32        khronos_uint32_t;
647 typedef __int64                 khronos_int64_t;
648 typedef unsigned __int64        khronos_uint64_t;
649 #define KHRONOS_SUPPORT_INT64   1
650 #define KHRONOS_SUPPORT_FLOAT   1
651
652 #elif defined(__sun__) || defined(__digital__)
```

```
653
654 /*
655  * Sun or Digital
656  */
657 typedef int                     khronos_int32_t;
658 typedef unsigned int            khronos_uint32_t;
659 #if defined(__arch64__) || defined(_LP64)
660 typedef long int                khronos_int64_t;
661 typedef unsigned long int       khronos_uint64_t;
662 #else
663 typedef long long int           khronos_int64_t;
664 typedef unsigned long long int  khronos_uint64_t;
665 #endif /* __arch64__ */
666 #define KHRONOS_SUPPORT_INT64   1
667 #define KHRONOS_SUPPORT_FLOAT   1
668
669 #elif 0
670
671 /*
672  * Hypothetical platform with no float or int64 support
673  */
674 typedef int                     khronos_int32_t;
675 typedef unsigned int            khronos_uint32_t;
676 #define KHRONOS_SUPPORT_INT64   0
677 #define KHRONOS_SUPPORT_FLOAT   0
678
679 #else
680
681 /*
682  * Generic fallback
683  */
684 #include <stdint.h>
685 typedef int32_t                 khronos_int32_t;
686 typedef uint32_t                khronos_uint32_t;
687 typedef int64_t                 khronos_int64_t;
688 typedef uint64_t                khronos_uint64_t;
689 #define KHRONOS_SUPPORT_INT64   1
690 #define KHRONOS_SUPPORT_FLOAT   1
691
692 #endif
693
694
695 /*
696  * Types that are (so far) the same on all platforms
697  */
698 typedef signed   char           khronos_int8_t;
699 typedef unsigned char           khronos_uint8_t;
700 typedef signed   short int      khronos_int16_t;
701 typedef unsigned short int      khronos_uint16_t;
702
703 /*
704  * Types that differ between LLP64 and LP64 architectures - in LLP64,
705  * pointers are 64 bits, but 'long' is still 32 bits. Win64 appears
706  * to be the only LLP64 architecture in current use.
707  */
708 #ifdef _WIN64
709 typedef signed   long long int khronos_intptr_t;
710 typedef unsigned long long int khronos_uintptr_t;
711 typedef signed   long long int khronos_ssize_t;
712 typedef unsigned long long int khronos_usize_t;
713 #else
714 typedef signed   long  int      khronos_intptr_t;
715 typedef unsigned long  int      khronos_uintptr_t;
716 typedef signed   long  int      khronos_ssize_t;
717 typedef unsigned long  int      khronos_usize_t;
718 #endif
719
720 #if KHRONOS_SUPPORT_FLOAT
721 /*
722  * Float type
723  */
724 typedef          float          khronos_float_t;
725 #endif
726
727 #if KHRONOS_SUPPORT_INT64
728 /* Time types
729  *
730  * These types can be used to represent a time interval in nanoseconds or
731  * an absolute Unadjusted System Time.  Unadjusted System Time is the number
732  * of nanoseconds since some arbitrary system event (e.g. since the last
733  * time the system booted).  The Unadjusted System Time is an unsigned
734  * 64 bit value that wraps back to 0 every 584 years.  Time intervals
735  * may be either signed or unsigned.
736  */
737 typedef khronos_uint64_t        khronos_utime_nanoseconds_t;
738 typedef khronos_int64_t         khronos_stime_nanoseconds_t;
739 #endif
```

```
740
741 /*
742  * Dummy value used to pad enum types to 32 bits.
743  */
744 #ifndef KHRONOS_MAX_ENUM
745 #define KHRONOS_MAX_ENUM 0x7FFFFFFF
746 #endif
747
748 /*
749  * Enumerated boolean type
750  *
751  * Values other than zero should be considered to be true.  Therefore
752  * comparisons should not be made against KHRONOS_TRUE.
753  */
754 typedef enum {
755     KHRONOS_FALSE = 0,
756     KHRONOS_TRUE  = 1,
757     KHRONOS_BOOLEAN_ENUM_FORCE_SIZE = KHRONOS_MAX_ENUM
758 } khronos_boolean_enum_t;
759
760 #endif /* __khrplatform_h_ */
761
762 typedef unsigned int GLenum;
763
764 typedef unsigned char GLboolean;
765
766 typedef unsigned int GLbitfield;
767
768 typedef void GLvoid;
769
770 typedef khronos_int8_t GLbyte;
771
772 typedef khronos_uint8_t GLubyte;
773
774 typedef khronos_int16_t GLshort;
775
776 typedef khronos_uint16_t GLushort;
777
778 typedef int GLint;
779
780 typedef unsigned int GLuint;
781
782 typedef khronos_int32_t GLclampx;
783
784 typedef int GLsizei;
785
786 typedef khronos_float_t GLfloat;
787
788 typedef khronos_float_t GLclampf;
789
790 typedef double GLdouble;
791
792 typedef double GLclampd;
793
794 typedef void *GLeglClientBufferEXT;
795
796 typedef void *GLeglImageOES;
797
798 typedef char GLchar;
799
800 typedef char GLcharARB;
801
802 #ifdef __APPLE__
803 typedef void *GLhandleARB;
804 #else
805 typedef unsigned int GLhandleARB;
806 #endif
807
808 typedef khronos_uint16_t GLhalf;
809
810 typedef khronos_uint16_t GLhalfARB;
811
812 typedef khronos_int32_t GLfixed;
813
814 #if defined(__ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__) &&
      (__ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__ > 1060)
815 typedef khronos_intptr_t GLintptr;
816 #else
817 typedef khronos_intptr_t GLintptr;
818 #endif
819
820 #if defined(__ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__) &&
      (__ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__ > 1060)
821 typedef khronos_intptr_t GLintptrARB;
822 #else
823 typedef khronos_intptr_t GLintptrARB;
824 #endif
```

```
825
826 #if defined(__ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__) &&
       (__ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__ > 1060)
827 typedef khronos_ssize_t GLsizeiptr;
828 #else
829 typedef khronos_ssize_t GLsizeiptr;
830 #endif
831
832 #if defined(__ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__) &&
       (__ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__ > 1060)
833 typedef khronos_ssize_t GLsizeiptrARB;
834 #else
835 typedef khronos_ssize_t GLsizeiptrARB;
836 #endif
837
838 typedef khronos_int64_t GLint64;
839
840 typedef khronos_int64_t GLint64EXT;
841
842 typedef khronos_uint64_t GLuint64;
843
844 typedef khronos_uint64_t GLuint64EXT;
845
846 typedef struct __GLsync *GLsync;
847
848 struct _cl_context;
849
850 struct _cl_event;
851
852 typedef void (GLAD_API_PTR *GLDEBUGPROC)(GLenum source,GLenum type,GLuint id,GLenum severity,GLsizei
       length,const GLchar *message,const void *userParam);
853
854 typedef void (GLAD_API_PTR *GLDEBUGPROCARB)(GLenum source,GLenum type,GLuint id,GLenum severity,GLsizei
       length,const GLchar *message,const void *userParam);
855
856 typedef void (GLAD_API_PTR *GLDEBUGPROCKHR)(GLenum source,GLenum type,GLuint id,GLenum severity,GLsizei
       length,const GLchar *message,const void *userParam);
857
858 typedef void (GLAD_API_PTR *GLDEBUGPROCAMD)(GLuint id,GLenum category,GLenum severity,GLsizei
       length,const GLchar *message,void *userParam);
859
860 typedef unsigned short GLhalfNV;
861
862 typedef GLintptr GLvdpauSurfaceNV;
863
864 typedef void (GLAD_API_PTR *GLVULKANPROCNV)(void);
865
866
867
868 #define GL_ES_VERSION_2_0 1
869 GLAD_API_CALL int GLAD_GL_ES_VERSION_2_0;
870
871
872 typedef void (GLAD_API_PTR *PFNGLACTIVETEXTUREPROC)(GLenum texture);
873 typedef void (GLAD_API_PTR *PFNGLATTACHSHADERPROC)(GLuint program, GLuint shader);
874 typedef void (GLAD_API_PTR *PFNGLBINDATTRIBLOCATIONPROC)(GLuint program, GLuint index, const GLchar *
       name);
875 typedef void (GLAD_API_PTR *PFNGLBINDBUFFERPROC)(GLenum target, GLuint buffer);
876 typedef void (GLAD_API_PTR *PFNGLBINDFRAMEBUFFERPROC)(GLenum target, GLuint framebuffer);
877 typedef void (GLAD_API_PTR *PFNGLBINDRENDERBUFFERPROC)(GLenum target, GLuint renderbuffer);
878 typedef void (GLAD_API_PTR *PFNGLBINDTEXTUREPROC)(GLenum target, GLuint texture);
879 typedef void (GLAD_API_PTR *PFNGLBLENDCOLORPROC)(GLfloat red, GLfloat green, GLfloat blue, GLfloat
       alpha);
880 typedef void (GLAD_API_PTR *PFNGLBLENDEQUATIONPROC)(GLenum mode);
881 typedef void (GLAD_API_PTR *PFNGLBLENDEQUATIONSEPARATEPROC)(GLenum modeRGB, GLenum modeAlpha);
882 typedef void (GLAD_API_PTR *PFNGLBLENDFUNCPROC)(GLenum sfactor, GLenum dfactor);
883 typedef void (GLAD_API_PTR *PFNGLBLENDFUNCSEPARATEPROC)(GLenum sfactorRGB, GLenum dfactorRGB, GLenum
       sfactorAlpha, GLenum dfactorAlpha);
884 typedef void (GLAD_API_PTR *PFNGLBUFFERDATAPROC)(GLenum target, GLsizeiptr size, const void * data,
       GLenum usage);
885 typedef void (GLAD_API_PTR *PFNGLBUFFERSUBDATAPROC)(GLenum target, GLintptr offset, GLsizeiptr size,
       const void * data);
886 typedef GLenum (GLAD_API_PTR *PFNGLCHECKFRAMEBUFFERSTATUSPROC)(GLenum target);
887 typedef void (GLAD_API_PTR *PFNGLCLEARPROC)(GLbitfield mask);
888 typedef void (GLAD_API_PTR *PFNGLCLEARCOLORPROC)(GLfloat red, GLfloat green, GLfloat blue, GLfloat
       alpha);
889 typedef void (GLAD_API_PTR *PFNGLCLEARDEPTHFPROC)(GLfloat d);
890 typedef void (GLAD_API_PTR *PFNGLCLEARSTENCILPROC)(GLint s);
891 typedef void (GLAD_API_PTR *PFNGLCOLORMASKPROC)(GLboolean red, GLboolean green, GLboolean blue,
       GLboolean alpha);
892 typedef void (GLAD_API_PTR *PFNGLCOMPILESHADERPROC)(GLuint shader);
893 typedef void (GLAD_API_PTR *PFNGLCOMPRESSEDTEXIMAGE2DPROC)(GLenum target, GLint level, GLenum
       internalformat, GLsizei width, GLsizei height, GLint border, GLsizei imageSize, const void * data);
894 typedef void (GLAD_API_PTR *PFNGLCOMPRESSEDTEXSUBIMAGE2DPROC)(GLenum target, GLint level, GLint xoffset,
       GLint yoffset, GLsizei width, GLsizei height, GLenum format, GLsizei imageSize, const void * data);
895 typedef void (GLAD_API_PTR *PFNGLCOPYTEXIMAGE2DPROC)(GLenum target, GLint level, GLenum internalformat,
       GLint x, GLint y, GLsizei width, GLsizei height, GLint border);
```

```
896 typedef void (GLAD_API_PTR *PFNGLCOPYTEXSUBIMAGE2DPROC)(GLenum target, GLint level, GLint xoffset, GLint
      yoffset, GLint x, GLint y, GLsizei width, GLsizei height);
897 typedef GLuint (GLAD_API_PTR *PFNGLCREATEPROGRAMPROC)(void);
898 typedef GLuint (GLAD_API_PTR *PFNGLCREATESHADERPROC)(GLenum type);
899 typedef void (GLAD_API_PTR *PFNGLCULLFACEPROC)(GLenum mode);
900 typedef void (GLAD_API_PTR *PFNGLDELETEBUFFERSPROC)(GLsizei n, const GLuint * buffers);
901 typedef void (GLAD_API_PTR *PFNGLDELETEFRAMEBUFFERSPROC)(GLsizei n, const GLuint * framebuffers);
902 typedef void (GLAD_API_PTR *PFNGLDELETEPROGRAMPROC)(GLuint program);
903 typedef void (GLAD_API_PTR *PFNGLDELETERENDERBUFFERSPROC)(GLsizei n, const GLuint * renderbuffers);
904 typedef void (GLAD_API_PTR *PFNGLDELETESHADERPROC)(GLuint shader);
905 typedef void (GLAD_API_PTR *PFNGLDELETETEXTURESPROC)(GLsizei n, const GLuint * textures);
906 typedef void (GLAD_API_PTR *PFNGLDEPTHFUNCPROC)(GLenum func);
907 typedef void (GLAD_API_PTR *PFNGLDEPTHMASKPROC)(GLboolean flag);
908 typedef void (GLAD_API_PTR *PFNGLDEPTHRANGEFPROC)(GLfloat n, GLfloat f);
909 typedef void (GLAD_API_PTR *PFNGLDETACHSHADERPROC)(GLuint program, GLuint shader);
910 typedef void (GLAD_API_PTR *PFNGLDISABLEPROC)(GLenum cap);
911 typedef void (GLAD_API_PTR *PFNGLDISABLEVERTEXATTRIBARRAYPROC)(GLuint index);
912 typedef void (GLAD_API_PTR *PFNGLDRAWARRAYSPROC)(GLenum mode, GLint first, GLsizei count);
913 typedef void (GLAD_API_PTR *PFNGLDRAWELEMENTSPROC)(GLenum mode, GLsizei count, GLenum type, const void *
      indices);
914 typedef void (GLAD_API_PTR *PFNGLENABLEPROC)(GLenum cap);
915 typedef void (GLAD_API_PTR *PFNGLENABLEVERTEXATTRIBARRAYPROC)(GLuint index);
916 typedef void (GLAD_API_PTR *PFNGLFINISHPROC)(void);
917 typedef void (GLAD_API_PTR *PFNGLFLUSHPROC)(void);
918 typedef void (GLAD_API_PTR *PFNGLFRAMEBUFFERRENDERBUFFERPROC)(GLenum target, GLenum attachment, GLenum
      renderbuffertarget, GLuint renderbuffer);
919 typedef void (GLAD_API_PTR *PFNGLFRAMEBUFFERTEXTURE2DPROC)(GLenum target, GLenum attachment, GLenum
      textarget, GLuint texture, GLint level);
920 typedef void (GLAD_API_PTR *PFNGLFRONTFACEPROC)(GLenum mode);
921 typedef void (GLAD_API_PTR *PFNGLGENBUFFERSPROC)(GLsizei n, GLuint * buffers);
922 typedef void (GLAD_API_PTR *PFNGLGENFRAMEBUFFERSPROC)(GLsizei n, GLuint * framebuffers);
923 typedef void (GLAD_API_PTR *PFNGLGENRENDERBUFFERSPROC)(GLsizei n, GLuint * renderbuffers);
924 typedef void (GLAD_API_PTR *PFNGLGENTEXTURESPROC)(GLsizei n, GLuint * textures);
925 typedef void (GLAD_API_PTR *PFNGLGENERATEMIPMAPPROC)(GLenum target);
926 typedef void (GLAD_API_PTR *PFNGLGETACTIVEATTRIBPROC)(GLuint program, GLuint index, GLsizei bufSize,
      GLsizei * length, GLint * size, GLenum * type, GLchar * name);
927 typedef void (GLAD_API_PTR *PFNGLGETACTIVEUNIFORMPROC)(GLuint program, GLuint index, GLsizei bufSize,
      GLsizei * length, GLint * size, GLenum * type, GLchar * name);
928 typedef void (GLAD_API_PTR *PFNGLGETATTACHEDSHADERSPROC)(GLuint program, GLsizei maxCount, GLsizei *
      count, GLuint * shaders);
929 typedef GLint (GLAD_API_PTR *PFNGLGETATTRIBLOCATIONPROC)(GLuint program, const GLchar * name);
930 typedef void (GLAD_API_PTR *PFNGLGETBOOLEANVPROC)(GLenum pname, GLboolean * data);
931 typedef void (GLAD_API_PTR *PFNGLGETBUFFERPARAMETERIVPROC)(GLenum target, GLenum pname, GLint * params);
932 typedef GLenum (GLAD_API_PTR *PFNGLGETERRORPROC)(void);
933 typedef void (GLAD_API_PTR *PFNGLGETFLOATVPROC)(GLenum pname, GLfloat * data);
934 typedef void (GLAD_API_PTR *PFNGLGETFRAMEBUFFERATTACHMENTPARAMETERIVPROC)(GLenum target, GLenum
      attachment, GLenum pname, GLint * params);
935 typedef void (GLAD_API_PTR *PFNGLGETINTEGERVPROC)(GLenum pname, GLint * data);
936 typedef void (GLAD_API_PTR *PFNGLGETPROGRAMINFOLOGPROC)(GLuint program, GLsizei bufSize, GLsizei *
      length, GLchar * infoLog);
937 typedef void (GLAD_API_PTR *PFNGLGETPROGRAMIVPROC)(GLuint program, GLenum pname, GLint * params);
938 typedef void (GLAD_API_PTR *PFNGLGETRENDERBUFFERPARAMETERIVPROC)(GLenum target, GLenum pname, GLint *
      params);
939 typedef void (GLAD_API_PTR *PFNGLGETSHADERINFOLOGPROC)(GLuint shader, GLsizei bufSize, GLsizei * length,
      GLchar * infoLog);
940 typedef void (GLAD_API_PTR *PFNGLGETSHADERPRECISIONFORMATPROC)(GLenum shadertype, GLenum precisiontype,
      GLint * range, GLint * precision);
941 typedef void (GLAD_API_PTR *PFNGLGETSHADERSOURCEPROC)(GLuint shader, GLsizei bufSize, GLsizei * length,
      GLchar * source);
942 typedef void (GLAD_API_PTR *PFNGLGETSHADERIVPROC)(GLuint shader, GLenum pname, GLint * params);
943 typedef const GLubyte * (GLAD_API_PTR *PFNGLGETSTRINGPROC)(GLenum name);
944 typedef void (GLAD_API_PTR *PFNGLGETTEXPARAMETERFVPROC)(GLenum target, GLenum pname, GLfloat * params);
945 typedef void (GLAD_API_PTR *PFNGLGETTEXPARAMETERIVPROC)(GLenum target, GLenum pname, GLint * params);
946 typedef GLint (GLAD_API_PTR *PFNGLGETUNIFORMLOCATIONPROC)(GLuint program, const GLchar * name);
947 typedef void (GLAD_API_PTR *PFNGLGETUNIFORMFVPROC)(GLuint program, GLint location, GLfloat * params);
948 typedef void (GLAD_API_PTR *PFNGLGETUNIFORMIVPROC)(GLuint program, GLint location, GLint * params);
949 typedef void (GLAD_API_PTR *PFNGLGETVERTEXATTRIBPOINTERVPROC)(GLuint index, GLenum pname, void **
      pointer);
950 typedef void (GLAD_API_PTR *PFNGLGETVERTEXATTRIBFVPROC)(GLuint index, GLenum pname, GLfloat * params);
951 typedef void (GLAD_API_PTR *PFNGLGETVERTEXATTRIBIVPROC)(GLuint index, GLenum pname, GLint * params);
952 typedef void (GLAD_API_PTR *PFNGLHINTPROC)(GLenum target, GLenum mode);
953 typedef GLboolean (GLAD_API_PTR *PFNGLISBUFFERPROC)(GLuint buffer);
954 typedef GLboolean (GLAD_API_PTR *PFNGLISENABLEDPROC)(GLenum cap);
955 typedef GLboolean (GLAD_API_PTR *PFNGLISFRAMEBUFFERPROC)(GLuint framebuffer);
956 typedef GLboolean (GLAD_API_PTR *PFNGLISPROGRAMPROC)(GLuint program);
957 typedef GLboolean (GLAD_API_PTR *PFNGLISRENDERBUFFERPROC)(GLuint renderbuffer);
958 typedef GLboolean (GLAD_API_PTR *PFNGLISSHADERPROC)(GLuint shader);
959 typedef GLboolean (GLAD_API_PTR *PFNGLISTEXTUREPROC)(GLuint texture);
960 typedef void (GLAD_API_PTR *PFNGLLINEWIDTHPROC)(GLfloat width);
961 typedef void (GLAD_API_PTR *PFNGLLINKPROGRAMPROC)(GLuint program);
962 typedef void (GLAD_API_PTR *PFNGLPIXELSTOREIPROC)(GLenum pname, GLint param);
963 typedef void (GLAD_API_PTR *PFNGLPOLYGONOFFSETPROC)(GLfloat factor, GLfloat units);
964 typedef void (GLAD_API_PTR *PFNGLREADPIXELSPROC)(GLint x, GLint y, GLsizei width, GLsizei height, GLenum
      format, GLenum type, void * pixels);
965 typedef void (GLAD_API_PTR *PFNGLRELEASESHADERCOMPILERPROC)(void);
966 typedef void (GLAD_API_PTR *PFNGLRENDERBUFFERSTORAGEPROC)(GLenum target, GLenum internalformat, GLsizei
      width, GLsizei height);
```

```
967 typedef void (GLAD_API_PTR *PFNGLSAMPLECOVERAGEPROC)(GLfloat value, GLboolean invert);
968 typedef void (GLAD_API_PTR *PFNGLSCISSORPROC)(GLint x, GLint y, GLsizei width, GLsizei height);
969 typedef void (GLAD_API_PTR *PFNGLSHADERBINARYPROC)(GLsizei count, const GLuint * shaders, GLenum
      binaryFormat, const void * binary, GLsizei length);
970 typedef void (GLAD_API_PTR *PFNGLSHADERSOURCEPROC)(GLuint shader, GLsizei count, const GLchar *const*
      string, const GLint * length);
971 typedef void (GLAD_API_PTR *PFNGLSTENCILFUNCPROC)(GLenum func, GLint ref, GLuint mask);
972 typedef void (GLAD_API_PTR *PFNGLSTENCILFUNCSEPARATEPROC)(GLenum face, GLenum func, GLint ref, GLuint
      mask);
973 typedef void (GLAD_API_PTR *PFNGLSTENCILMASKPROC)(GLuint mask);
974 typedef void (GLAD_API_PTR *PFNGLSTENCILMASKSEPARATEPROC)(GLenum face, GLuint mask);
975 typedef void (GLAD_API_PTR *PFNGLSTENCILOPPROC)(GLenum fail, GLenum zfail, GLenum zpass);
976 typedef void (GLAD_API_PTR *PFNGLSTENCILOPSEPARATEPROC)(GLenum face, GLenum sfail, GLenum dpfail, GLenum
      dppass);
977 typedef void (GLAD_API_PTR *PFNGLTEXIMAGE2DPROC)(GLenum target, GLint level, GLint internalformat,
      GLsizei width, GLsizei height, GLint border, GLenum format, GLenum type, const void * pixels);
978 typedef void (GLAD_API_PTR *PFNGLTEXPARAMETERFPROC)(GLenum target, GLenum pname, GLfloat param);
979 typedef void (GLAD_API_PTR *PFNGLTEXPARAMETERFVPROC)(GLenum target, GLenum pname, const GLfloat *
      params);
980 typedef void (GLAD_API_PTR *PFNGLTEXPARAMETERIPROC)(GLenum target, GLenum pname, GLint param);
981 typedef void (GLAD_API_PTR *PFNGLTEXPARAMETERIVPROC)(GLenum target, GLenum pname, const GLint * params);
982 typedef void (GLAD_API_PTR *PFNGLTEXSUBIMAGE2DPROC)(GLenum target, GLint level, GLint xoffset, GLint
      yoffset, GLsizei width, GLsizei height, GLenum format, GLenum type, const void * pixels);
983 typedef void (GLAD_API_PTR *PFNGLUNIFORM1FPROC)(GLint location, GLfloat v0);
984 typedef void (GLAD_API_PTR *PFNGLUNIFORM1FVPROC)(GLint location, GLsizei count, const GLfloat * value);
985 typedef void (GLAD_API_PTR *PFNGLUNIFORM1IPROC)(GLint location, GLint v0);
986 typedef void (GLAD_API_PTR *PFNGLUNIFORM1IVPROC)(GLint location, GLsizei count, const GLint * value);
987 typedef void (GLAD_API_PTR *PFNGLUNIFORM2FPROC)(GLint location, GLfloat v0, GLfloat v1);
988 typedef void (GLAD_API_PTR *PFNGLUNIFORM2FVPROC)(GLint location, GLsizei count, const GLfloat * value);
989 typedef void (GLAD_API_PTR *PFNGLUNIFORM2IPROC)(GLint location, GLint v0, GLint v1);
990 typedef void (GLAD_API_PTR *PFNGLUNIFORM2IVPROC)(GLint location, GLsizei count, const GLint * value);
991 typedef void (GLAD_API_PTR *PFNGLUNIFORM3FPROC)(GLint location, GLfloat v0, GLfloat v1, GLfloat v2);
992 typedef void (GLAD_API_PTR *PFNGLUNIFORM3FVPROC)(GLint location, GLsizei count, const GLfloat * value);
993 typedef void (GLAD_API_PTR *PFNGLUNIFORM3IPROC)(GLint location, GLint v0, GLint v1, GLint v2);
994 typedef void (GLAD_API_PTR *PFNGLUNIFORM3IVPROC)(GLint location, GLsizei count, const GLint * value);
995 typedef void (GLAD_API_PTR *PFNGLUNIFORM4FPROC)(GLint location, GLfloat v0, GLfloat v1, GLfloat v2,
      GLfloat v3);
996 typedef void (GLAD_API_PTR *PFNGLUNIFORM4FVPROC)(GLint location, GLsizei count, const GLfloat * value);
997 typedef void (GLAD_API_PTR *PFNGLUNIFORM4IPROC)(GLint location, GLint v0, GLint v1, GLint v2, GLint v3);
998 typedef void (GLAD_API_PTR *PFNGLUNIFORM4IVPROC)(GLint location, GLsizei count, const GLint * value);
999 typedef void (GLAD_API_PTR *PFNGLUNIFORMMATRIX2FVPROC)(GLint location, GLsizei count, GLboolean
      transpose, const GLfloat * value);
1000 typedef void (GLAD_API_PTR *PFNGLUNIFORMMATRIX3FVPROC)(GLint location, GLsizei count, GLboolean
      transpose, const GLfloat * value);
1001 typedef void (GLAD_API_PTR *PFNGLUNIFORMMATRIX4FVPROC)(GLint location, GLsizei count, GLboolean
      transpose, const GLfloat * value);
1002 typedef void (GLAD_API_PTR *PFNGLUSEPROGRAMPROC)(GLuint program);
1003 typedef void (GLAD_API_PTR *PFNGLVALIDATEPROGRAMPROC)(GLuint program);
1004 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB1FPROC)(GLuint index, GLfloat x);
1005 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB1FVPROC)(GLuint index, const GLfloat * v);
1006 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB2FPROC)(GLuint index, GLfloat x, GLfloat y);
1007 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB2FVPROC)(GLuint index, const GLfloat * v);
1008 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB3FPROC)(GLuint index, GLfloat x, GLfloat y, GLfloat z);
1009 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB3FVPROC)(GLuint index, const GLfloat * v);
1010 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4FPROC)(GLuint index, GLfloat x, GLfloat y, GLfloat z,
      GLfloat w);
1011 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIB4FVPROC)(GLuint index, const GLfloat * v);
1012 typedef void (GLAD_API_PTR *PFNGLVERTEXATTRIBPOINTERPROC)(GLuint index, GLint size, GLenum type,
      GLboolean normalized, GLsizei stride, const void * pointer);
1013 typedef void (GLAD_API_PTR *PFNGLVIEWPORTPROC)(GLint x, GLint y, GLsizei width, GLsizei height);
1014
1015 GLAD_API_CALL PFNGLACTIVETEXTUREPROC glad_glActiveTexture;
1016 #define glActiveTexture glad_glActiveTexture
1017 GLAD_API_CALL PFNGLATTACHSHADERPROC glad_glAttachShader;
1018 #define glAttachShader glad_glAttachShader
1019 GLAD_API_CALL PFNGLBINDATTRIBLOCATIONPROC glad_glBindAttribLocation;
1020 #define glBindAttribLocation glad_glBindAttribLocation
1021 GLAD_API_CALL PFNGLBINDBUFFERPROC glad_glBindBuffer;
1022 #define glBindBuffer glad_glBindBuffer
1023 GLAD_API_CALL PFNGLBINDFRAMEBUFFERPROC glad_glBindFramebuffer;
1024 #define glBindFramebuffer glad_glBindFramebuffer
1025 GLAD_API_CALL PFNGLBINDRENDERBUFFERPROC glad_glBindRenderbuffer;
1026 #define glBindRenderbuffer glad_glBindRenderbuffer
1027 GLAD_API_CALL PFNGLBINDTEXTUREPROC glad_glBindTexture;
1028 #define glBindTexture glad_glBindTexture
1029 GLAD_API_CALL PFNGLBLENDCOLORPROC glad_glBlendColor;
1030 #define glBlendColor glad_glBlendColor
1031 GLAD_API_CALL PFNGLBLENDEQUATIONPROC glad_glBlendEquation;
1032 #define glBlendEquation glad_glBlendEquation
1033 GLAD_API_CALL PFNGLBLENDEQUATIONSEPARATEPROC glad_glBlendEquationSeparate;
1034 #define glBlendEquationSeparate glad_glBlendEquationSeparate
1035 GLAD_API_CALL PFNGLBLENDFUNCPROC glad_glBlendFunc;
1036 #define glBlendFunc glad_glBlendFunc
1037 GLAD_API_CALL PFNGLBLENDFUNCSEPARATEPROC glad_glBlendFuncSeparate;
1038 #define glBlendFuncSeparate glad_glBlendFuncSeparate
1039 GLAD_API_CALL PFNGLBUFFERDATAPROC glad_glBufferData;
1040 #define glBufferData glad_glBufferData
```

```
1041 GLAD_API_CALL PFNGLBUFFERSUBDATAPROC glad_glBufferSubData;
1042 #define glBufferSubData glad_glBufferSubData
1043 GLAD_API_CALL PFNGLCHECKFRAMEBUFFERSTATUSPROC glad_glCheckFramebufferStatus;
1044 #define glCheckFramebufferStatus glad_glCheckFramebufferStatus
1045 GLAD_API_CALL PFNGLCLEARPROC glad_glClear;
1046 #define glClear glad_glClear
1047 GLAD_API_CALL PFNGLCLEARCOLORPROC glad_glClearColor;
1048 #define glClearColor glad_glClearColor
1049 GLAD_API_CALL PFNGLCLEARDEPTHFPROC glad_glClearDepthf;
1050 #define glClearDepthf glad_glClearDepthf
1051 GLAD_API_CALL PFNGLCLEARSTENCILPROC glad_glClearStencil;
1052 #define glClearStencil glad_glClearStencil
1053 GLAD_API_CALL PFNGLCOLORMASKPROC glad_glColorMask;
1054 #define glColorMask glad_glColorMask
1055 GLAD_API_CALL PFNGLCOMPILESHADERPROC glad_glCompileShader;
1056 #define glCompileShader glad_glCompileShader
1057 GLAD_API_CALL PFNGLCOMPRESSEDTEXIMAGE2DPROC glad_glCompressedTexImage2D;
1058 #define glCompressedTexImage2D glad_glCompressedTexImage2D
1059 GLAD_API_CALL PFNGLCOMPRESSEDTEXSUBIMAGE2DPROC glad_glCompressedTexSubImage2D;
1060 #define glCompressedTexSubImage2D glad_glCompressedTexSubImage2D
1061 GLAD_API_CALL PFNGLCOPYTEXIMAGE2DPROC glad_glCopyTexImage2D;
1062 #define glCopyTexImage2D glad_glCopyTexImage2D
1063 GLAD_API_CALL PFNGLCOPYTEXSUBIMAGE2DPROC glad_glCopyTexSubImage2D;
1064 #define glCopyTexSubImage2D glad_glCopyTexSubImage2D
1065 GLAD_API_CALL PFNGLCREATEPROGRAMPROC glad_glCreateProgram;
1066 #define glCreateProgram glad_glCreateProgram
1067 GLAD_API_CALL PFNGLCREATESHADERPROC glad_glCreateShader;
1068 #define glCreateShader glad_glCreateShader
1069 GLAD_API_CALL PFNGLCULLFACEPROC glad_glCullFace;
1070 #define glCullFace glad_glCullFace
1071 GLAD_API_CALL PFNGLDELETEBUFFERSPROC glad_glDeleteBuffers;
1072 #define glDeleteBuffers glad_glDeleteBuffers
1073 GLAD_API_CALL PFNGLDELETEFRAMEBUFFERSPROC glad_glDeleteFramebuffers;
1074 #define glDeleteFramebuffers glad_glDeleteFramebuffers
1075 GLAD_API_CALL PFNGLDELETEPROGRAMPROC glad_glDeleteProgram;
1076 #define glDeleteProgram glad_glDeleteProgram
1077 GLAD_API_CALL PFNGLDELETERENDERBUFFERSPROC glad_glDeleteRenderbuffers;
1078 #define glDeleteRenderbuffers glad_glDeleteRenderbuffers
1079 GLAD_API_CALL PFNGLDELETESHADERPROC glad_glDeleteShader;
1080 #define glDeleteShader glad_glDeleteShader
1081 GLAD_API_CALL PFNGLDELETETEXTURESPROC glad_glDeleteTextures;
1082 #define glDeleteTextures glad_glDeleteTextures
1083 GLAD_API_CALL PFNGLDEPTHFUNCPROC glad_glDepthFunc;
1084 #define glDepthFunc glad_glDepthFunc
1085 GLAD_API_CALL PFNGLDEPTHMASKPROC glad_glDepthMask;
1086 #define glDepthMask glad_glDepthMask
1087 GLAD_API_CALL PFNGLDEPTHRANGEFPROC glad_glDepthRangef;
1088 #define glDepthRangef glad_glDepthRangef
1089 GLAD_API_CALL PFNGLDETACHSHADERPROC glad_glDetachShader;
1090 #define glDetachShader glad_glDetachShader
1091 GLAD_API_CALL PFNGLDISABLEPROC glad_glDisable;
1092 #define glDisable glad_glDisable
1093 GLAD_API_CALL PFNGLDISABLEVERTEXATTRIBARRAYPROC glad_glDisableVertexAttribArray;
1094 #define glDisableVertexAttribArray glad_glDisableVertexAttribArray
1095 GLAD_API_CALL PFNGLDRAWARRAYSPROC glad_glDrawArrays;
1096 #define glDrawArrays glad_glDrawArrays
1097 GLAD_API_CALL PFNGLDRAWELEMENTSPROC glad_glDrawElements;
1098 #define glDrawElements glad_glDrawElements
1099 GLAD_API_CALL PFNGLENABLEPROC glad_glEnable;
1100 #define glEnable glad_glEnable
1101 GLAD_API_CALL PFNGLENABLEVERTEXATTRIBARRAYPROC glad_glEnableVertexAttribArray;
1102 #define glEnableVertexAttribArray glad_glEnableVertexAttribArray
1103 GLAD_API_CALL PFNGLFINISHPROC glad_glFinish;
1104 #define glFinish glad_glFinish
1105 GLAD_API_CALL PFNGLFLUSHPROC glad_glFlush;
1106 #define glFlush glad_glFlush
1107 GLAD_API_CALL PFNGLFRAMEBUFFERRENDERBUFFERPROC glad_glFramebufferRenderbuffer;
1108 #define glFramebufferRenderbuffer glad_glFramebufferRenderbuffer
1109 GLAD_API_CALL PFNGLFRAMEBUFFERTEXTURE2DPROC glad_glFramebufferTexture2D;
1110 #define glFramebufferTexture2D glad_glFramebufferTexture2D
1111 GLAD_API_CALL PFNGLFRONTFACEPROC glad_glFrontFace;
1112 #define glFrontFace glad_glFrontFace
1113 GLAD_API_CALL PFNGLGENBUFFERSPROC glad_glGenBuffers;
1114 #define glGenBuffers glad_glGenBuffers
1115 GLAD_API_CALL PFNGLGENFRAMEBUFFERSPROC glad_glGenFramebuffers;
1116 #define glGenFramebuffers glad_glGenFramebuffers
1117 GLAD_API_CALL PFNGLGENRENDERBUFFERSPROC glad_glGenRenderbuffers;
1118 #define glGenRenderbuffers glad_glGenRenderbuffers
1119 GLAD_API_CALL PFNGLGENTEXTURESPROC glad_glGenTextures;
1120 #define glGenTextures glad_glGenTextures
1121 GLAD_API_CALL PFNGLGENERATEMIPMAPPROC glad_glGenerateMipmap;
1122 #define glGenerateMipmap glad_glGenerateMipmap
1123 GLAD_API_CALL PFNGLGETACTIVEATTRIBPROC glad_glGetActiveAttrib;
1124 #define glGetActiveAttrib glad_glGetActiveAttrib
1125 GLAD_API_CALL PFNGLGETACTIVEUNIFORMPROC glad_glGetActiveUniform;
1126 #define glGetActiveUniform glad_glGetActiveUniform
1127 GLAD_API_CALL PFNGLGETATTACHEDSHADERSPROC glad_glGetAttachedShaders;
```

```
1128 #define glGetAttachedShaders glad_glGetAttachedShaders
1129 GLAD_API_CALL PFNGLGETATTRIBLOCATIONPROC glad_glGetAttribLocation;
1130 #define glGetAttribLocation glad_glGetAttribLocation
1131 GLAD_API_CALL PFNGLGETBOOLEANVPROC glad_glGetBooleanv;
1132 #define glGetBooleanv glad_glGetBooleanv
1133 GLAD_API_CALL PFNGLGETBUFFERPARAMETERIVPROC glad_glGetBufferParameteriv;
1134 #define glGetBufferParameteriv glad_glGetBufferParameteriv
1135 GLAD_API_CALL PFNGLGETERRORPROC glad_glGetError;
1136 #define glGetError glad_glGetError
1137 GLAD_API_CALL PFNGLGETFLOATVPROC glad_glGetFloatv;
1138 #define glGetFloatv glad_glGetFloatv
1139 GLAD_API_CALL PFNGLGETFRAMEBUFFERATTACHMENTPARAMETERIVPROC glad_glGetFramebufferAttachmentParameteriv;
1140 #define glGetFramebufferAttachmentParameteriv glad_glGetFramebufferAttachmentParameteriv
1141 GLAD_API_CALL PFNGLGETINTEGERVPROC glad_glGetIntegerv;
1142 #define glGetIntegerv glad_glGetIntegerv
1143 GLAD_API_CALL PFNGLGETPROGRAMINFOLOGPROC glad_glGetProgramInfoLog;
1144 #define glGetProgramInfoLog glad_glGetProgramInfoLog
1145 GLAD_API_CALL PFNGLGETPROGRAMIVPROC glad_glGetProgramiv;
1146 #define glGetProgramiv glad_glGetProgramiv
1147 GLAD_API_CALL PFNGLGETRENDERBUFFERPARAMETERIVPROC glad_glGetRenderbufferParameteriv;
1148 #define glGetRenderbufferParameteriv glad_glGetRenderbufferParameteriv
1149 GLAD_API_CALL PFNGLGETSHADERINFOLOGPROC glad_glGetShaderInfoLog;
1150 #define glGetShaderInfoLog glad_glGetShaderInfoLog
1151 GLAD_API_CALL PFNGLGETSHADERPRECISIONFORMATPROC glad_glGetShaderPrecisionFormat;
1152 #define glGetShaderPrecisionFormat glad_glGetShaderPrecisionFormat
1153 GLAD_API_CALL PFNGLGETSHADERSOURCEPROC glad_glGetShaderSource;
1154 #define glGetShaderSource glad_glGetShaderSource
1155 GLAD_API_CALL PFNGLGETSHADERIVPROC glad_glGetShaderiv;
1156 #define glGetShaderiv glad_glGetShaderiv
1157 GLAD_API_CALL PFNGLGETSTRINGPROC glad_glGetString;
1158 #define glGetString glad_glGetString
1159 GLAD_API_CALL PFNGLGETTEXPARAMETERFVPROC glad_glGetTexParameterfv;
1160 #define glGetTexParameterfv glad_glGetTexParameterfv
1161 GLAD_API_CALL PFNGLGETTEXPARAMETERIVPROC glad_glGetTexParameteriv;
1162 #define glGetTexParameteriv glad_glGetTexParameteriv
1163 GLAD_API_CALL PFNGLGETUNIFORMLOCATIONPROC glad_glGetUniformLocation;
1164 #define glGetUniformLocation glad_glGetUniformLocation
1165 GLAD_API_CALL PFNGLGETUNIFORMFVPROC glad_glGetUniformfv;
1166 #define glGetUniformfv glad_glGetUniformfv
1167 GLAD_API_CALL PFNGLGETUNIFORMIVPROC glad_glGetUniformiv;
1168 #define glGetUniformiv glad_glGetUniformiv
1169 GLAD_API_CALL PFNGLGETVERTEXATTRIBPOINTERVPROC glad_glGetVertexAttribPointerv;
1170 #define glGetVertexAttribPointerv glad_glGetVertexAttribPointerv
1171 GLAD_API_CALL PFNGLGETVERTEXATTRIBFVPROC glad_glGetVertexAttribfv;
1172 #define glGetVertexAttribfv glad_glGetVertexAttribfv
1173 GLAD_API_CALL PFNGLGETVERTEXATTRIBIVPROC glad_glGetVertexAttribiv;
1174 #define glGetVertexAttribiv glad_glGetVertexAttribiv
1175 GLAD_API_CALL PFNGLHINTPROC glad_glHint;
1176 #define glHint glad_glHint
1177 GLAD_API_CALL PFNGLISBUFFERPROC glad_glIsBuffer;
1178 #define glIsBuffer glad_glIsBuffer
1179 GLAD_API_CALL PFNGLISENABLEDPROC glad_glIsEnabled;
1180 #define glIsEnabled glad_glIsEnabled
1181 GLAD_API_CALL PFNGLISFRAMEBUFFERPROC glad_glIsFramebuffer;
1182 #define glIsFramebuffer glad_glIsFramebuffer
1183 GLAD_API_CALL PFNGLISPROGRAMPROC glad_glIsProgram;
1184 #define glIsProgram glad_glIsProgram
1185 GLAD_API_CALL PFNGLISRENDERBUFFERPROC glad_glIsRenderbuffer;
1186 #define glIsRenderbuffer glad_glIsRenderbuffer
1187 GLAD_API_CALL PFNGLISSHADERPROC glad_glIsShader;
1188 #define glIsShader glad_glIsShader
1189 GLAD_API_CALL PFNGLISTEXTUREPROC glad_glIsTexture;
1190 #define glIsTexture glad_glIsTexture
1191 GLAD_API_CALL PFNGLLINEWIDTHPROC glad_glLineWidth;
1192 #define glLineWidth glad_glLineWidth
1193 GLAD_API_CALL PFNGLLINKPROGRAMPROC glad_glLinkProgram;
1194 #define glLinkProgram glad_glLinkProgram
1195 GLAD_API_CALL PFNGLPIXELSTOREIPROC glad_glPixelStorei;
1196 #define glPixelStorei glad_glPixelStorei
1197 GLAD_API_CALL PFNGLPOLYGONOFFSETPROC glad_glPolygonOffset;
1198 #define glPolygonOffset glad_glPolygonOffset
1199 GLAD_API_CALL PFNGLREADPIXELSPROC glad_glReadPixels;
1200 #define glReadPixels glad_glReadPixels
1201 GLAD_API_CALL PFNGLRELEASESHADERCOMPILERPROC glad_glReleaseShaderCompiler;
1202 #define glReleaseShaderCompiler glad_glReleaseShaderCompiler
1203 GLAD_API_CALL PFNGLRENDERBUFFERSTORAGEPROC glad_glRenderbufferStorage;
1204 #define glRenderbufferStorage glad_glRenderbufferStorage
1205 GLAD_API_CALL PFNGLSAMPLECOVERAGEPROC glad_glSampleCoverage;
1206 #define glSampleCoverage glad_glSampleCoverage
1207 GLAD_API_CALL PFNGLSCISSORPROC glad_glScissor;
1208 #define glScissor glad_glScissor
1209 GLAD_API_CALL PFNGLSHADERBINARYPROC glad_glShaderBinary;
1210 #define glShaderBinary glad_glShaderBinary
1211 GLAD_API_CALL PFNGLSHADERSOURCEPROC glad_glShaderSource;
1212 #define glShaderSource glad_glShaderSource
1213 GLAD_API_CALL PFNGLSTENCILFUNCPROC glad_glStencilFunc;
1214 #define glStencilFunc glad_glStencilFunc
```

```
1215 GLAD_API_CALL PFNGLSTENCILFUNCSEPARATEPROC glad_glStencilFuncSeparate;
1216 #define glStencilFuncSeparate glad_glStencilFuncSeparate
1217 GLAD_API_CALL PFNGLSTENCILMASKPROC glad_glStencilMask;
1218 #define glStencilMask glad_glStencilMask
1219 GLAD_API_CALL PFNGLSTENCILMASKSEPARATEPROC glad_glStencilMaskSeparate;
1220 #define glStencilMaskSeparate glad_glStencilMaskSeparate
1221 GLAD_API_CALL PFNGLSTENCILOPPROC glad_glStencilOp;
1222 #define glStencilOp glad_glStencilOp
1223 GLAD_API_CALL PFNGLSTENCILOPSEPARATEPROC glad_glStencilOpSeparate;
1224 #define glStencilOpSeparate glad_glStencilOpSeparate
1225 GLAD_API_CALL PFNGLTEXIMAGE2DPROC glad_glTexImage2D;
1226 #define glTexImage2D glad_glTexImage2D
1227 GLAD_API_CALL PFNGLTEXPARAMETERFPROC glad_glTexParameterf;
1228 #define glTexParameterf glad_glTexParameterf
1229 GLAD_API_CALL PFNGLTEXPARAMETERFVPROC glad_glTexParameterfv;
1230 #define glTexParameterfv glad_glTexParameterfv
1231 GLAD_API_CALL PFNGLTEXPARAMETERIPROC glad_glTexParameteri;
1232 #define glTexParameteri glad_glTexParameteri
1233 GLAD_API_CALL PFNGLTEXPARAMETERIVPROC glad_glTexParameteriv;
1234 #define glTexParameteriv glad_glTexParameteriv
1235 GLAD_API_CALL PFNGLTEXSUBIMAGE2DPROC glad_glTexSubImage2D;
1236 #define glTexSubImage2D glad_glTexSubImage2D
1237 GLAD_API_CALL PFNGLUNIFORM1FPROC glad_glUniform1f;
1238 #define glUniform1f glad_glUniform1f
1239 GLAD_API_CALL PFNGLUNIFORM1FVPROC glad_glUniform1fv;
1240 #define glUniform1fv glad_glUniform1fv
1241 GLAD_API_CALL PFNGLUNIFORM1IPROC glad_glUniform1i;
1242 #define glUniform1i glad_glUniform1i
1243 GLAD_API_CALL PFNGLUNIFORM1IVPROC glad_glUniform1iv;
1244 #define glUniform1iv glad_glUniform1iv
1245 GLAD_API_CALL PFNGLUNIFORM2FPROC glad_glUniform2f;
1246 #define glUniform2f glad_glUniform2f
1247 GLAD_API_CALL PFNGLUNIFORM2FVPROC glad_glUniform2fv;
1248 #define glUniform2fv glad_glUniform2fv
1249 GLAD_API_CALL PFNGLUNIFORM2IPROC glad_glUniform2i;
1250 #define glUniform2i glad_glUniform2i
1251 GLAD_API_CALL PFNGLUNIFORM2IVPROC glad_glUniform2iv;
1252 #define glUniform2iv glad_glUniform2iv
1253 GLAD_API_CALL PFNGLUNIFORM3FPROC glad_glUniform3f;
1254 #define glUniform3f glad_glUniform3f
1255 GLAD_API_CALL PFNGLUNIFORM3FVPROC glad_glUniform3fv;
1256 #define glUniform3fv glad_glUniform3fv
1257 GLAD_API_CALL PFNGLUNIFORM3IPROC glad_glUniform3i;
1258 #define glUniform3i glad_glUniform3i
1259 GLAD_API_CALL PFNGLUNIFORM3IVPROC glad_glUniform3iv;
1260 #define glUniform3iv glad_glUniform3iv
1261 GLAD_API_CALL PFNGLUNIFORM4FPROC glad_glUniform4f;
1262 #define glUniform4f glad_glUniform4f
1263 GLAD_API_CALL PFNGLUNIFORM4FVPROC glad_glUniform4fv;
1264 #define glUniform4fv glad_glUniform4fv
1265 GLAD_API_CALL PFNGLUNIFORM4IPROC glad_glUniform4i;
1266 #define glUniform4i glad_glUniform4i
1267 GLAD_API_CALL PFNGLUNIFORM4IVPROC glad_glUniform4iv;
1268 #define glUniform4iv glad_glUniform4iv
1269 GLAD_API_CALL PFNGLUNIFORMMATRIX2FVPROC glad_glUniformMatrix2fv;
1270 #define glUniformMatrix2fv glad_glUniformMatrix2fv
1271 GLAD_API_CALL PFNGLUNIFORMMATRIX3FVPROC glad_glUniformMatrix3fv;
1272 #define glUniformMatrix3fv glad_glUniformMatrix3fv
1273 GLAD_API_CALL PFNGLUNIFORMMATRIX4FVPROC glad_glUniformMatrix4fv;
1274 #define glUniformMatrix4fv glad_glUniformMatrix4fv
1275 GLAD_API_CALL PFNGLUSEPROGRAMPROC glad_glUseProgram;
1276 #define glUseProgram glad_glUseProgram
1277 GLAD_API_CALL PFNGLVALIDATEPROGRAMPROC glad_glValidateProgram;
1278 #define glValidateProgram glad_glValidateProgram
1279 GLAD_API_CALL PFNGLVERTEXATTRIB1FPROC glad_glVertexAttrib1f;
1280 #define glVertexAttrib1f glad_glVertexAttrib1f
1281 GLAD_API_CALL PFNGLVERTEXATTRIB1FVPROC glad_glVertexAttrib1fv;
1282 #define glVertexAttrib1fv glad_glVertexAttrib1fv
1283 GLAD_API_CALL PFNGLVERTEXATTRIB2FPROC glad_glVertexAttrib2f;
1284 #define glVertexAttrib2f glad_glVertexAttrib2f
1285 GLAD_API_CALL PFNGLVERTEXATTRIB2FVPROC glad_glVertexAttrib2fv;
1286 #define glVertexAttrib2fv glad_glVertexAttrib2fv
1287 GLAD_API_CALL PFNGLVERTEXATTRIB3FPROC glad_glVertexAttrib3f;
1288 #define glVertexAttrib3f glad_glVertexAttrib3f
1289 GLAD_API_CALL PFNGLVERTEXATTRIB3FVPROC glad_glVertexAttrib3fv;
1290 #define glVertexAttrib3fv glad_glVertexAttrib3fv
1291 GLAD_API_CALL PFNGLVERTEXATTRIB4FPROC glad_glVertexAttrib4f;
1292 #define glVertexAttrib4f glad_glVertexAttrib4f
1293 GLAD_API_CALL PFNGLVERTEXATTRIB4FVPROC glad_glVertexAttrib4fv;
1294 #define glVertexAttrib4fv glad_glVertexAttrib4fv
1295 GLAD_API_CALL PFNGLVERTEXATTRIBPOINTERPROC glad_glVertexAttribPointer;
1296 #define glVertexAttribPointer glad_glVertexAttribPointer
1297 GLAD_API_CALL PFNGLVIEWPORTPROC glad_glViewport;
1298 #define glViewport glad_glViewport
1299
1300
1301
```

```
1302
1303
1304 GLAD_API_CALL int gladLoadGLES2UserPtr( GLADuserptrloadfunc load, void *userptr);
1305 GLAD_API_CALL int gladLoadGLES2( GLADloadfunc load);
1306
1307
1308
1309 #ifdef __cplusplus
1310 }
1311 #endif
1312 #endif
1313
1314 /* Source */
1315 #ifdef GLAD_GLES2_IMPLEMENTATION
1316 #include <stdio.h>
1317 #include <stdlib.h>
1318 #include <string.h>
1319
1320 #ifndef GLAD_IMPL_UTIL_C_
1321 #define GLAD_IMPL_UTIL_C_
1322
1323 #ifdef _MSC_VER
1324 #define GLAD_IMPL_UTIL_SSCANF sscanf_s
1325 #else
1326 #define GLAD_IMPL_UTIL_SSCANF sscanf
1327 #endif
1328
1329 #endif /* GLAD_IMPL_UTIL_C_ */
1330
1331 #ifdef __cplusplus
1332 extern "C" {
1333 #endif
1334
1335
1336
1337 int GLAD_GL_ES_VERSION_2_0 = 0;
1338
1339
1340
1341 PFNGLACTIVETEXTUREPROC glad_glActiveTexture = NULL;
1342 PFNGLATTACHSHADERPROC glad_glAttachShader = NULL;
1343 PFNGLBINDATTRIBLOCATIONPROC glad_glBindAttribLocation = NULL;
1344 PFNGLBINDBUFFERPROC glad_glBindBuffer = NULL;
1345 PFNGLBINDFRAMEBUFFERPROC glad_glBindFramebuffer = NULL;
1346 PFNGLBINDRENDERBUFFERPROC glad_glBindRenderbuffer = NULL;
1347 PFNGLBINDTEXTUREPROC glad_glBindTexture = NULL;
1348 PFNGLBLENDCOLORPROC glad_glBlendColor = NULL;
1349 PFNGLBLENDEQUATIONPROC glad_glBlendEquation = NULL;
1350 PFNGLBLENDEQUATIONSEPARATEPROC glad_glBlendEquationSeparate = NULL;
1351 PFNGLBLENDFUNCPROC glad_glBlendFunc = NULL;
1352 PFNGLBLENDFUNCSEPARATEPROC glad_glBlendFuncSeparate = NULL;
1353 PFNGLBUFFERDATAPROC glad_glBufferData = NULL;
1354 PFNGLBUFFERSUBDATAPROC glad_glBufferSubData = NULL;
1355 PFNGLCHECKFRAMEBUFFERSTATUSPROC glad_glCheckFramebufferStatus = NULL;
1356 PFNGLCLEARPROC glad_glClear = NULL;
1357 PFNGLCLEARCOLORPROC glad_glClearColor = NULL;
1358 PFNGLCLEARDEPTHFPROC glad_glClearDepthf = NULL;
1359 PFNGLCLEARSTENCILPROC glad_glClearStencil = NULL;
1360 PFNGLCOLORMASKPROC glad_glColorMask = NULL;
1361 PFNGLCOMPILESHADERPROC glad_glCompileShader = NULL;
1362 PFNGLCOMPRESSEDTEXIMAGE2DPROC glad_glCompressedTexImage2D = NULL;
1363 PFNGLCOMPRESSEDTEXSUBIMAGE2DPROC glad_glCompressedTexSubImage2D = NULL;
1364 PFNGLCOPYTEXIMAGE2DPROC glad_glCopyTexImage2D = NULL;
1365 PFNGLCOPYTEXSUBIMAGE2DPROC glad_glCopyTexSubImage2D = NULL;
1366 PFNGLCREATEPROGRAMPROC glad_glCreateProgram = NULL;
1367 PFNGLCREATESHADERPROC glad_glCreateShader = NULL;
1368 PFNGLCULLFACEPROC glad_glCullFace = NULL;
1369 PFNGLDELETEBUFFERSPROC glad_glDeleteBuffers = NULL;
1370 PFNGLDELETEFRAMEBUFFERSPROC glad_glDeleteFramebuffers = NULL;
1371 PFNGLDELETEPROGRAMPROC glad_glDeleteProgram = NULL;
1372 PFNGLDELETERENDERBUFFERSPROC glad_glDeleteRenderbuffers = NULL;
1373 PFNGLDELETESHADERPROC glad_glDeleteShader = NULL;
1374 PFNGLDELETETEXTURESPROC glad_glDeleteTextures = NULL;
1375 PFNGLDEPTHFUNCPROC glad_glDepthFunc = NULL;
1376 PFNGLDEPTHMASKPROC glad_glDepthMask = NULL;
1377 PFNGLDEPTHRANGEFPROC glad_glDepthRangef = NULL;
1378 PFNGLDETACHSHADERPROC glad_glDetachShader = NULL;
1379 PFNGLDISABLEPROC glad_glDisable = NULL;
1380 PFNGLDISABLEVERTEXATTRIBARRAYPROC glad_glDisableVertexAttribArray = NULL;
1381 PFNGLDRAWARRAYSPROC glad_glDrawArrays = NULL;
1382 PFNGLDRAWELEMENTSPROC glad_glDrawElements = NULL;
1383 PFNGLENABLEPROC glad_glEnable = NULL;
1384 PFNGLENABLEVERTEXATTRIBARRAYPROC glad_glEnableVertexAttribArray = NULL;
1385 PFNGLFINISHPROC glad_glFinish = NULL;
1386 PFNGLFLUSHPROC glad_glFlush = NULL;
1387 PFNGLFRAMEBUFFERRENDERBUFFERPROC glad_glFramebufferRenderbuffer = NULL;
1388 PFNGLFRAMEBUFFERTEXTURE2DPROC glad_glFramebufferTexture2D = NULL;
```

```
1389 PFNGLFRONTFACEPROC glad_glFrontFace = NULL;
1390 PFNGLGENBUFFERSPROC glad_glGenBuffers = NULL;
1391 PFNGLGENFRAMEBUFFERSPROC glad_glGenFramebuffers = NULL;
1392 PFNGLGENRENDERBUFFERSPROC glad_glGenRenderbuffers = NULL;
1393 PFNGLGENTEXTURESPROC glad_glGenTextures = NULL;
1394 PFNGLGENERATEMIPMAPPROC glad_glGenerateMipmap = NULL;
1395 PFNGLGETACTIVEATTRIBPROC glad_glGetActiveAttrib = NULL;
1396 PFNGLGETACTIVEUNIFORMPROC glad_glGetActiveUniform = NULL;
1397 PFNGLGETATTACHEDSHADERSPROC glad_glGetAttachedShaders = NULL;
1398 PFNGLGETATTRIBLOCATIONPROC glad_glGetAttribLocation = NULL;
1399 PFNGLGETBOOLEANVPROC glad_glGetBooleanv = NULL;
1400 PFNGLGETBUFFERPARAMETERIVPROC glad_glGetBufferParameteriv = NULL;
1401 PFNGLGETERRORPROC glad_glGetError = NULL;
1402 PFNGLGETFLOATVPROC glad_glGetFloatv = NULL;
1403 PFNGLGETFRAMEBUFFERATTACHMENTPARAMETERIVPROC glad_glGetFramebufferAttachmentParameteriv = NULL;
1404 PFNGLGETINTEGERVPROC glad_glGetIntegerv = NULL;
1405 PFNGLGETPROGRAMINFOLOGPROC glad_glGetProgramInfoLog = NULL;
1406 PFNGLGETPROGRAMIVPROC glad_glGetProgramiv = NULL;
1407 PFNGLGETRENDERBUFFERPARAMETERIVPROC glad_glGetRenderbufferParameteriv = NULL;
1408 PFNGLGETSHADERINFOLOGPROC glad_glGetShaderInfoLog = NULL;
1409 PFNGLGETSHADERPRECISIONFORMATPROC glad_glGetShaderPrecisionFormat = NULL;
1410 PFNGLGETSHADERSOURCEPROC glad_glGetShaderSource = NULL;
1411 PFNGLGETSHADERIVPROC glad_glGetShaderiv = NULL;
1412 PFNGLGETSTRINGPROC glad_glGetString = NULL;
1413 PFNGLGETTEXPARAMETERFVPROC glad_glGetTexParameterfv = NULL;
1414 PFNGLGETTEXPARAMETERIVPROC glad_glGetTexParameteriv = NULL;
1415 PFNGLGETUNIFORMLOCATIONPROC glad_glGetUniformLocation = NULL;
1416 PFNGLGETUNIFORMFVPROC glad_glGetUniformfv = NULL;
1417 PFNGLGETUNIFORMIVPROC glad_glGetUniformiv = NULL;
1418 PFNGLGETVERTEXATTRIBPOINTERVPROC glad_glGetVertexAttribPointerv = NULL;
1419 PFNGLGETVERTEXATTRIBFVPROC glad_glGetVertexAttribfv = NULL;
1420 PFNGLGETVERTEXATTRIBIVPROC glad_glGetVertexAttribiv = NULL;
1421 PFNGLHINTPROC glad_glHint = NULL;
1422 PFNGLISBUFFERPROC glad_glIsBuffer = NULL;
1423 PFNGLISENABLEDPROC glad_glIsEnabled = NULL;
1424 PFNGLISFRAMEBUFFERPROC glad_glIsFramebuffer = NULL;
1425 PFNGLISPROGRAMPROC glad_glIsProgram = NULL;
1426 PFNGLISRENDERBUFFERPROC glad_glIsRenderbuffer = NULL;
1427 PFNGLISSHADERPROC glad_glIsShader = NULL;
1428 PFNGLISTEXTUREPROC glad_glIsTexture = NULL;
1429 PFNGLLINEWIDTHPROC glad_glLineWidth = NULL;
1430 PFNGLLINKPROGRAMPROC glad_glLinkProgram = NULL;
1431 PFNGLPIXELSTOREIPROC glad_glPixelStorei = NULL;
1432 PFNGLPOLYGONOFFSETPROC glad_glPolygonOffset = NULL;
1433 PFNGLREADPIXELSPROC glad_glReadPixels = NULL;
1434 PFNGLRELEASESHADERCOMPILERPROC glad_glReleaseShaderCompiler = NULL;
1435 PFNGLRENDERBUFFERSTORAGEPROC glad_glRenderbufferStorage = NULL;
1436 PFNGLSAMPLECOVERAGEPROC glad_glSampleCoverage = NULL;
1437 PFNGLSCISSORPROC glad_glScissor = NULL;
1438 PFNGLSHADERBINARYPROC glad_glShaderBinary = NULL;
1439 PFNGLSHADERSOURCEPROC glad_glShaderSource = NULL;
1440 PFNGLSTENCILFUNCPROC glad_glStencilFunc = NULL;
1441 PFNGLSTENCILFUNCSEPARATEPROC glad_glStencilFuncSeparate = NULL;
1442 PFNGLSTENCILMASKPROC glad_glStencilMask = NULL;
1443 PFNGLSTENCILMASKSEPARATEPROC glad_glStencilMaskSeparate = NULL;
1444 PFNGLSTENCILOPPROC glad_glStencilOp = NULL;
1445 PFNGLSTENCILOPSEPARATEPROC glad_glStencilOpSeparate = NULL;
1446 PFNGLTEXIMAGE2DPROC glad_glTexImage2D = NULL;
1447 PFNGLTEXPARAMETERFPROC glad_glTexParameterf = NULL;
1448 PFNGLTEXPARAMETERFVPROC glad_glTexParameterfv = NULL;
1449 PFNGLTEXPARAMETERIPROC glad_glTexParameteri = NULL;
1450 PFNGLTEXPARAMETERIVPROC glad_glTexParameteriv = NULL;
1451 PFNGLTEXSUBIMAGE2DPROC glad_glTexSubImage2D = NULL;
1452 PFNGLUNIFORM1FPROC glad_glUniform1f = NULL;
1453 PFNGLUNIFORM1FVPROC glad_glUniform1fv = NULL;
1454 PFNGLUNIFORM1IPROC glad_glUniform1i = NULL;
1455 PFNGLUNIFORM1IVPROC glad_glUniform1iv = NULL;
1456 PFNGLUNIFORM2FPROC glad_glUniform2f = NULL;
1457 PFNGLUNIFORM2FVPROC glad_glUniform2fv = NULL;
1458 PFNGLUNIFORM2IPROC glad_glUniform2i = NULL;
1459 PFNGLUNIFORM2IVPROC glad_glUniform2iv = NULL;
1460 PFNGLUNIFORM3FPROC glad_glUniform3f = NULL;
1461 PFNGLUNIFORM3FVPROC glad_glUniform3fv = NULL;
1462 PFNGLUNIFORM3IPROC glad_glUniform3i = NULL;
1463 PFNGLUNIFORM3IVPROC glad_glUniform3iv = NULL;
1464 PFNGLUNIFORM4FPROC glad_glUniform4f = NULL;
1465 PFNGLUNIFORM4FVPROC glad_glUniform4fv = NULL;
1466 PFNGLUNIFORM4IPROC glad_glUniform4i = NULL;
1467 PFNGLUNIFORM4IVPROC glad_glUniform4iv = NULL;
1468 PFNGLUNIFORMMATRIX2FVPROC glad_glUniformMatrix2fv = NULL;
1469 PFNGLUNIFORMMATRIX3FVPROC glad_glUniformMatrix3fv = NULL;
1470 PFNGLUNIFORMMATRIX4FVPROC glad_glUniformMatrix4fv = NULL;
1471 PFNGLUSEPROGRAMPROC glad_glUseProgram = NULL;
1472 PFNGLVALIDATEPROGRAMPROC glad_glValidateProgram = NULL;
1473 PFNGLVERTEXATTRIB1FPROC glad_glVertexAttrib1f = NULL;
1474 PFNGLVERTEXATTRIB1FVPROC glad_glVertexAttrib1fv = NULL;
1475 PFNGLVERTEXATTRIB2FPROC glad_glVertexAttrib2f = NULL;
```

```
1476 PFNGLVERTEXATTRIB2FVPROC glad_glVertexAttrib2fv = NULL;
1477 PFNGLVERTEXATTRIB3FPROC glad_glVertexAttrib3f = NULL;
1478 PFNGLVERTEXATTRIB3FVPROC glad_glVertexAttrib3fv = NULL;
1479 PFNGLVERTEXATTRIB4FPROC glad_glVertexAttrib4f = NULL;
1480 PFNGLVERTEXATTRIB4FVPROC glad_glVertexAttrib4fv = NULL;
1481 PFNGLVERTEXATTRIBPOINTERPROC glad_glVertexAttribPointer = NULL;
1482 PFNGLVIEWPORTPROC glad_glViewport = NULL;
1483
1484
1485 static void glad_gl_load_GL_ES_VERSION_2_0( GLADuserptrloadfunc load, void* userptr) {
1486     if(!GLAD_GL_ES_VERSION_2_0) return;
1487     glad_glActiveTexture = (PFNGLACTIVETEXTUREPROC) load(userptr, "glActiveTexture");
1488     glad_glAttachShader = (PFNGLATTACHSHADERPROC) load(userptr, "glAttachShader");
1489     glad_glBindAttribLocation = (PFNGLBINDATTRIBLOCATIONPROC) load(userptr, "glBindAttribLocation");
1490     glad_glBindBuffer = (PFNGLBINDBUFFERPROC) load(userptr, "glBindBuffer");
1491     glad_glBindFramebuffer = (PFNGLBINDFRAMEBUFFERPROC) load(userptr, "glBindFramebuffer");
1492     glad_glBindRenderbuffer = (PFNGLBINDRENDERBUFFERPROC) load(userptr, "glBindRenderbuffer");
1493     glad_glBindTexture = (PFNGLBINDTEXTUREPROC) load(userptr, "glBindTexture");
1494     glad_glBlendColor = (PFNGLBLENDCOLORPROC) load(userptr, "glBlendColor");
1495     glad_glBlendEquation = (PFNGLBLENDEQUATIONPROC) load(userptr, "glBlendEquation");
1496     glad_glBlendEquationSeparate = (PFNGLBLENDEQUATIONSEPARATEPROC) load(userptr,
    "glBlendEquationSeparate");
1497     glad_glBlendFunc = (PFNGLBLENDFUNCPROC) load(userptr, "glBlendFunc");
1498     glad_glBlendFuncSeparate = (PFNGLBLENDFUNCSEPARATEPROC) load(userptr, "glBlendFuncSeparate");
1499     glad_glBufferData = (PFNGLBUFFERDATAPROC) load(userptr, "glBufferData");
1500     glad_glBufferSubData = (PFNGLBUFFERSUBDATAPROC) load(userptr, "glBufferSubData");
1501     glad_glCheckFramebufferStatus = (PFNGLCHECKFRAMEBUFFERSTATUSPROC) load(userptr,
    "glCheckFramebufferStatus");
1502     glad_glClear = (PFNGLCLEARPROC) load(userptr, "glClear");
1503     glad_glClearColor = (PFNGLCLEARCOLORPROC) load(userptr, "glClearColor");
1504     glad_glClearDepthf = (PFNGLCLEARDEPTHFPROC) load(userptr, "glClearDepthf");
1505     glad_glClearStencil = (PFNGLCLEARSTENCILPROC) load(userptr, "glClearStencil");
1506     glad_glColorMask = (PFNGLCOLORMASKPROC) load(userptr, "glColorMask");
1507     glad_glCompileShader = (PFNGLCOMPILESHADERPROC) load(userptr, "glCompileShader");
1508     glad_glCompressedTexImage2D = (PFNGLCOMPRESSEDTEXIMAGE2DPROC) load(userptr,
    "glCompressedTexImage2D");
1509     glad_glCompressedTexSubImage2D = (PFNGLCOMPRESSEDTEXSUBIMAGE2DPROC) load(userptr,
    "glCompressedTexSubImage2D");
1510     glad_glCopyTexImage2D = (PFNGLCOPYTEXIMAGE2DPROC) load(userptr, "glCopyTexImage2D");
1511     glad_glCopyTexSubImage2D = (PFNGLCOPYTEXSUBIMAGE2DPROC) load(userptr, "glCopyTexSubImage2D");
1512     glad_glCreateProgram = (PFNGLCREATEPROGRAMPROC) load(userptr, "glCreateProgram");
1513     glad_glCreateShader = (PFNGLCREATESHADERPROC) load(userptr, "glCreateShader");
1514     glad_glCullFace = (PFNGLCULLFACEPROC) load(userptr, "glCullFace");
1515     glad_glDeleteBuffers = (PFNGLDELETEBUFFERSPROC) load(userptr, "glDeleteBuffers");
1516     glad_glDeleteFramebuffers = (PFNGLDELETEFRAMEBUFFERSPROC) load(userptr, "glDeleteFramebuffers");
1517     glad_glDeleteProgram = (PFNGLDELETEPROGRAMPROC) load(userptr, "glDeleteProgram");
1518     glad_glDeleteRenderbuffers = (PFNGLDELETERENDERBUFFERSPROC) load(userptr, "glDeleteRenderbuffers");
1519     glad_glDeleteShader = (PFNGLDELETESHADERPROC) load(userptr, "glDeleteShader");
1520     glad_glDeleteTextures = (PFNGLDELETETEXTURESPROC) load(userptr, "glDeleteTextures");
1521     glad_glDepthFunc = (PFNGLDEPTHFUNCPROC) load(userptr, "glDepthFunc");
1522     glad_glDepthMask = (PFNGLDEPTHMASKPROC) load(userptr, "glDepthMask");
1523     glad_glDepthRangef = (PFNGLDEPTHRANGEFPROC) load(userptr, "glDepthRangef");
1524     glad_glDetachShader = (PFNGLDETACHSHADERPROC) load(userptr, "glDetachShader");
1525     glad_glDisable = (PFNGLDISABLEPROC) load(userptr, "glDisable");
1526     glad_glDisableVertexAttribArray = (PFNGLDISABLEVERTEXATTRIBARRAYPROC) load(userptr,
    "glDisableVertexAttribArray");
1527     glad_glDrawArrays = (PFNGLDRAWARRAYSPROC) load(userptr, "glDrawArrays");
1528     glad_glDrawElements = (PFNGLDRAWELEMENTSPROC) load(userptr, "glDrawElements");
1529     glad_glEnable = (PFNGLENABLEPROC) load(userptr, "glEnable");
1530     glad_glEnableVertexAttribArray = (PFNGLENABLEVERTEXATTRIBARRAYPROC) load(userptr,
    "glEnableVertexAttribArray");
1531     glad_glFinish = (PFNGLFINISHPROC) load(userptr, "glFinish");
1532     glad_glFlush = (PFNGLFLUSHPROC) load(userptr, "glFlush");
1533     glad_glFramebufferRenderbuffer = (PFNGLFRAMEBUFFERRENDERBUFFERPROC) load(userptr,
    "glFramebufferRenderbuffer");
1534     glad_glFramebufferTexture2D = (PFNGLFRAMEBUFFERTEXTURE2DPROC) load(userptr,
    "glFramebufferTexture2D");
1535     glad_glFrontFace = (PFNGLFRONTFACEPROC) load(userptr, "glFrontFace");
1536     glad_glGenBuffers = (PFNGLGENBUFFERSPROC) load(userptr, "glGenBuffers");
1537     glad_glGenFramebuffers = (PFNGLGENFRAMEBUFFERSPROC) load(userptr, "glGenFramebuffers");
1538     glad_glGenRenderbuffers = (PFNGLGENRENDERBUFFERSPROC) load(userptr, "glGenRenderbuffers");
1539     glad_glGenTextures = (PFNGLGENTEXTURESPROC) load(userptr, "glGenTextures");
1540     glad_glGenerateMipmap = (PFNGLGENERATEMIPMAPPROC) load(userptr, "glGenerateMipmap");
1541     glad_glGetActiveAttrib = (PFNGLGETACTIVEATTRIBPROC) load(userptr, "glGetActiveAttrib");
1542     glad_glGetActiveUniform = (PFNGLGETACTIVEUNIFORMPROC) load(userptr, "glGetActiveUniform");
1543     glad_glGetAttachedShaders = (PFNGLGETATTACHEDSHADERSPROC) load(userptr, "glGetAttachedShaders");
1544     glad_glGetAttribLocation = (PFNGLGETATTRIBLOCATIONPROC) load(userptr, "glGetAttribLocation");
1545     glad_glGetBooleanv = (PFNGLGETBOOLEANVPROC) load(userptr, "glGetBooleanv");
1546     glad_glGetBufferParameteriv = (PFNGLGETBUFFERPARAMETERIVPROC) load(userptr,
    "glGetBufferParameteriv");
1547     glad_glGetError = (PFNGLGETERRORPROC) load(userptr, "glGetError");
1548     glad_glGetFloatv = (PFNGLGETFLOATVPROC) load(userptr, "glGetFloatv");
1549     glad_glGetFramebufferAttachmentParameteriv = (PFNGLGETFRAMEBUFFERATTACHMENTPARAMETERIVPROC)
    load(userptr, "glGetFramebufferAttachmentParameteriv");
1550     glad_glGetIntegerv = (PFNGLGETINTEGERVPROC) load(userptr, "glGetIntegerv");
1551     glad_glGetProgramInfoLog = (PFNGLGETPROGRAMINFOLOGPROC) load(userptr, "glGetProgramInfoLog");
1552     glad_glGetProgramiv = (PFNGLGETPROGRAMIVPROC) load(userptr, "glGetProgramiv");
```

```
1553     glad_glGetRenderbufferParameteriv = (PFNGLGETRENDERBUFFERPARAMETERIVPROC) load(userptr,
    "glGetRenderbufferParameteriv");
1554     glad_glGetShaderInfoLog = (PFNGLGETSHADERINFOLOGPROC) load(userptr, "glGetShaderInfoLog");
1555     glad_glGetShaderPrecisionFormat = (PFNGLGETSHADERPRECISIONFORMATPROC) load(userptr,
    "glGetShaderPrecisionFormat");
1556     glad_glGetShaderSource = (PFNGLGETSHADERSOURCEPROC) load(userptr, "glGetShaderSource");
1557     glad_glGetShaderiv = (PFNGLGETSHADERIVPROC) load(userptr, "glGetShaderiv");
1558     glad_glGetString = (PFNGLGETSTRINGPROC) load(userptr, "glGetString");
1559     glad_glGetTexParameterfv = (PFNGLGETTEXPARAMETERFVPROC) load(userptr, "glGetTexParameterfv");
1560     glad_glGetTexParameteriv = (PFNGLGETTEXPARAMETERIVPROC) load(userptr, "glGetTexParameteriv");
1561     glad_glGetUniformLocation = (PFNGLGETUNIFORMLOCATIONPROC) load(userptr, "glGetUniformLocation");
1562     glad_glGetUniformfv = (PFNGLGETUNIFORMFVPROC) load(userptr, "glGetUniformfv");
1563     glad_glGetUniformiv = (PFNGLGETUNIFORMIVPROC) load(userptr, "glGetUniformiv");
1564     glad_glGetVertexAttribPointerv = (PFNGLGETVERTEXATTRIBPOINTERVPROC) load(userptr,
    "glGetVertexAttribPointerv");
1565     glad_glGetVertexAttribfv = (PFNGLGETVERTEXATTRIBFVPROC) load(userptr, "glGetVertexAttribfv");
1566     glad_glGetVertexAttribiv = (PFNGLGETVERTEXATTRIBIVPROC) load(userptr, "glGetVertexAttribiv");
1567     glad_glHint = (PFNGLHINTPROC) load(userptr, "glHint");
1568     glad_glIsBuffer = (PFNGLISBUFFERPROC) load(userptr, "glIsBuffer");
1569     glad_glIsEnabled = (PFNGLISENABLEDPROC) load(userptr, "glIsEnabled");
1570     glad_glIsFramebuffer = (PFNGLISFRAMEBUFFERPROC) load(userptr, "glIsFramebuffer");
1571     glad_glIsProgram = (PFNGLISPROGRAMPROC) load(userptr, "glIsProgram");
1572     glad_glIsRenderbuffer = (PFNGLISRENDERBUFFERPROC) load(userptr, "glIsRenderbuffer");
1573     glad_glIsShader = (PFNGLISSHADERPROC) load(userptr, "glIsShader");
1574     glad_glIsTexture = (PFNGLISTEXTUREPROC) load(userptr, "glIsTexture");
1575     glad_glLineWidth = (PFNGLLINEWIDTHPROC) load(userptr, "glLineWidth");
1576     glad_glLinkProgram = (PFNGLLINKPROGRAMPROC) load(userptr, "glLinkProgram");
1577     glad_glPixelStorei = (PFNGLPIXELSTOREIPROC) load(userptr, "glPixelStorei");
1578     glad_glPolygonOffset = (PFNGLPOLYGONOFFSETPROC) load(userptr, "glPolygonOffset");
1579     glad_glReadPixels = (PFNGLREADPIXELSPROC) load(userptr, "glReadPixels");
1580     glad_glReleaseShaderCompiler = (PFNGLRELEASESHADERCOMPILERPROC) load(userptr,
    "glReleaseShaderCompiler");
1581     glad_glRenderbufferStorage = (PFNGLRENDERBUFFERSTORAGEPROC) load(userptr, "glRenderbufferStorage");
1582     glad_glSampleCoverage = (PFNGLSAMPLECOVERAGEPROC) load(userptr, "glSampleCoverage");
1583     glad_glScissor = (PFNGLSCISSORPROC) load(userptr, "glScissor");
1584     glad_glShaderBinary = (PFNGLSHADERBINARYPROC) load(userptr, "glShaderBinary");
1585     glad_glShaderSource = (PFNGLSHADERSOURCEPROC) load(userptr, "glShaderSource");
1586     glad_glStencilFunc = (PFNGLSTENCILFUNCPROC) load(userptr, "glStencilFunc");
1587     glad_glStencilFuncSeparate = (PFNGLSTENCILFUNCSEPARATEPROC) load(userptr, "glStencilFuncSeparate");
1588     glad_glStencilMask = (PFNGLSTENCILMASKPROC) load(userptr, "glStencilMask");
1589     glad_glStencilMaskSeparate = (PFNGLSTENCILMASKSEPARATEPROC) load(userptr, "glStencilMaskSeparate");
1590     glad_glStencilOp = (PFNGLSTENCILOPPROC) load(userptr, "glStencilOp");
1591     glad_glStencilOpSeparate = (PFNGLSTENCILOPSEPARATEPROC) load(userptr, "glStencilOpSeparate");
1592     glad_glTexImage2D = (PFNGLTEXIMAGE2DPROC) load(userptr, "glTexImage2D");
1593     glad_glTexParameterf = (PFNGLTEXPARAMETERFPROC) load(userptr, "glTexParameterf");
1594     glad_glTexParameterfv = (PFNGLTEXPARAMETERFVPROC) load(userptr, "glTexParameterfv");
1595     glad_glTexParameteri = (PFNGLTEXPARAMETERIPROC) load(userptr, "glTexParameteri");
1596     glad_glTexParameteriv = (PFNGLTEXPARAMETERIVPROC) load(userptr, "glTexParameteriv");
1597     glad_glTexSubImage2D = (PFNGLTEXSUBIMAGE2DPROC) load(userptr, "glTexSubImage2D");
1598     glad_glUniform1f = (PFNGLUNIFORM1FPROC) load(userptr, "glUniform1f");
1599     glad_glUniform1fv = (PFNGLUNIFORM1FVPROC) load(userptr, "glUniform1fv");
1600     glad_glUniform1i = (PFNGLUNIFORM1IPROC) load(userptr, "glUniform1i");
1601     glad_glUniform1iv = (PFNGLUNIFORM1IVPROC) load(userptr, "glUniform1iv");
1602     glad_glUniform2f = (PFNGLUNIFORM2FPROC) load(userptr, "glUniform2f");
1603     glad_glUniform2fv = (PFNGLUNIFORM2FVPROC) load(userptr, "glUniform2fv");
1604     glad_glUniform2i = (PFNGLUNIFORM2IPROC) load(userptr, "glUniform2i");
1605     glad_glUniform2iv = (PFNGLUNIFORM2IVPROC) load(userptr, "glUniform2iv");
1606     glad_glUniform3f = (PFNGLUNIFORM3FPROC) load(userptr, "glUniform3f");
1607     glad_glUniform3fv = (PFNGLUNIFORM3FVPROC) load(userptr, "glUniform3fv");
1608     glad_glUniform3i = (PFNGLUNIFORM3IPROC) load(userptr, "glUniform3i");
1609     glad_glUniform3iv = (PFNGLUNIFORM3IVPROC) load(userptr, "glUniform3iv");
1610     glad_glUniform4f = (PFNGLUNIFORM4FPROC) load(userptr, "glUniform4f");
1611     glad_glUniform4fv = (PFNGLUNIFORM4FVPROC) load(userptr, "glUniform4fv");
1612     glad_glUniform4i = (PFNGLUNIFORM4IPROC) load(userptr, "glUniform4i");
1613     glad_glUniform4iv = (PFNGLUNIFORM4IVPROC) load(userptr, "glUniform4iv");
1614     glad_glUniformMatrix2fv = (PFNGLUNIFORMMATRIX2FVPROC) load(userptr, "glUniformMatrix2fv");
1615     glad_glUniformMatrix3fv = (PFNGLUNIFORMMATRIX3FVPROC) load(userptr, "glUniformMatrix3fv");
1616     glad_glUniformMatrix4fv = (PFNGLUNIFORMMATRIX4FVPROC) load(userptr, "glUniformMatrix4fv");
1617     glad_glUseProgram = (PFNGLUSEPROGRAMPROC) load(userptr, "glUseProgram");
1618     glad_glValidateProgram = (PFNGLVALIDATEPROGRAMPROC) load(userptr, "glValidateProgram");
1619     glad_glVertexAttrib1f = (PFNGLVERTEXATTRIB1FPROC) load(userptr, "glVertexAttrib1f");
1620     glad_glVertexAttrib1fv = (PFNGLVERTEXATTRIB1FVPROC) load(userptr, "glVertexAttrib1fv");
1621     glad_glVertexAttrib2f = (PFNGLVERTEXATTRIB2FPROC) load(userptr, "glVertexAttrib2f");
1622     glad_glVertexAttrib2fv = (PFNGLVERTEXATTRIB2FVPROC) load(userptr, "glVertexAttrib2fv");
1623     glad_glVertexAttrib3f = (PFNGLVERTEXATTRIB3FPROC) load(userptr, "glVertexAttrib3f");
1624     glad_glVertexAttrib3fv = (PFNGLVERTEXATTRIB3FVPROC) load(userptr, "glVertexAttrib3fv");
1625     glad_glVertexAttrib4f = (PFNGLVERTEXATTRIB4FPROC) load(userptr, "glVertexAttrib4f");
1626     glad_glVertexAttrib4fv = (PFNGLVERTEXATTRIB4FVPROC) load(userptr, "glVertexAttrib4fv");
1627     glad_glVertexAttribPointer = (PFNGLVERTEXATTRIBPOINTERPROC) load(userptr, "glVertexAttribPointer");
1628     glad_glViewport = (PFNGLVIEWPORTPROC) load(userptr, "glViewport");
1629 }
1630
1631
1632
1633 #if defined(GL_ES_VERSION_3_0) || defined(GL_VERSION_3_0)
1634 #define GLAD_GL_IS_SOME_NEW_VERSION 1
1635 #else
```

```
1636 #define GLAD_GL_IS_SOME_NEW_VERSION 0
1637 #endif
1638
1639 static int glad_gl_get_extensions( int version, const char **out_exts, unsigned int *out_num_exts_i,
       char ***out_exts_i) {
1640 #if GLAD_GL_IS_SOME_NEW_VERSION
1641     if(GLAD_VERSION_MAJOR(version) < 3) {
1642 #else
1643     (void) version;
1644     (void) out_num_exts_i;
1645     (void) out_exts_i;
1646 #endif
1647         if (glad_glGetString == NULL) {
1648             return 0;
1649         }
1650         *out_exts = (const char *)glad_glGetString(GL_EXTENSIONS);
1651 #if GLAD_GL_IS_SOME_NEW_VERSION
1652     } else {
1653         unsigned int index = 0;
1654         unsigned int num_exts_i = 0;
1655         char **exts_i = NULL;
1656         if (glad_glGetStringi == NULL || glad_glGetIntegerv == NULL) {
1657             return 0;
1658         }
1659         glad_glGetIntegerv(GL_NUM_EXTENSIONS, (int*) &num_exts_i);
1660         if (num_exts_i > 0) {
1661             exts_i = (char **) malloc(num_exts_i * (sizeof *exts_i));
1662         }
1663         if (exts_i == NULL) {
1664             return 0;
1665         }
1666         for(index = 0; index < num_exts_i; index++) {
1667             const char *gl_str_tmp = (const char*) glad_glGetStringi(GL_EXTENSIONS, index);
1668             size_t len = strlen(gl_str_tmp) + 1;
1669
1670             char *local_str = (char*) malloc(len * sizeof(char));
1671             if(local_str != NULL) {
1672                 memcpy(local_str, gl_str_tmp, len * sizeof(char));
1673             }
1674
1675             exts_i[index] = local_str;
1676         }
1677
1678         *out_num_exts_i = num_exts_i;
1679         *out_exts_i = exts_i;
1680     }
1681 #endif
1682     return 1;
1683 }
1684 static void glad_gl_free_extensions(char **exts_i, unsigned int num_exts_i) {
1685     if (exts_i != NULL) {
1686         unsigned int index;
1687         for(index = 0; index < num_exts_i; index++) {
1688             free((void *) (exts_i[index]));
1689         }
1690         free((void *)exts_i);
1691         exts_i = NULL;
1692     }
1693 }
1694 static int glad_gl_has_extension(int version, const char *exts, unsigned int num_exts_i, char **exts_i,
       const char *ext) {
1695     if(GLAD_VERSION_MAJOR(version) < 3 || !GLAD_GL_IS_SOME_NEW_VERSION) {
1696         const char *extensions;
1697         const char *loc;
1698         const char *terminator;
1699         extensions = exts;
1700         if(extensions == NULL || ext == NULL) {
1701             return 0;
1702         }
1703         while(1) {
1704             loc = strstr(extensions, ext);
1705             if(loc == NULL) {
1706                 return 0;
1707             }
1708             terminator = loc + strlen(ext);
1709             if((loc == extensions || *(loc - 1) == ' ') &&
1710                 (*terminator == ' ' || *terminator == '\0')) {
1711                 return 1;
1712             }
1713             extensions = terminator;
1714         }
1715     } else {
1716         unsigned int index;
1717         for(index = 0; index < num_exts_i; index++) {
1718             const char *e = exts_i[index];
1719             if(strcmp(e, ext) == 0) {
1720                 return 1;
```

```
1721             }
1722         }
1723     }
1724     return 0;
1725 }
1726
1727 static GLADapiproc glad_gl_get_proc_from_userptr(void *userptr, const char* name) {
1728     return (GLAD_GNUC_EXTENSION (GLADapiproc (*)(const char *name)) userptr)(name);
1729 }
1730
1731 static int glad_gl_find_extensions_gles2( int version) {
1732     const char *exts = NULL;
1733     unsigned int num_exts_i = 0;
1734     char **exts_i = NULL;
1735     if (!glad_gl_get_extensions(version, &exts, &num_exts_i, &exts_i)) return 0;
1736
1737     (void) glad_gl_has_extension;
1738
1739     glad_gl_free_extensions(exts_i, num_exts_i);
1740
1741     return 1;
1742 }
1743
1744 static int glad_gl_find_core_gles2(void) {
1745     int i;
1746     const char* version;
1747     const char* prefixes[] = {
1748         "OpenGL ES-CM ",
1749         "OpenGL ES-CL ",
1750         "OpenGL ES ",
1751         "OpenGL SC ",
1752         NULL
1753     };
1754     int major = 0;
1755     int minor = 0;
1756     version = (const char*) glad_glGetString(GL_VERSION);
1757     if (!version) return 0;
1758     for (i = 0;  prefixes[i];  i++) {
1759         const size_t length = strlen(prefixes[i]);
1760         if (strncmp(version, prefixes[i], length) == 0) {
1761             version += length;
1762             break;
1763         }
1764     }
1765
1766     GLAD_IMPL_UTIL_SSCANF(version, "%d.%d", &major, &minor);
1767
1768     GLAD_GL_ES_VERSION_2_0 = (major == 2 && minor >= 0) || major > 2;
1769
1770     return GLAD_MAKE_VERSION(major, minor);
1771 }
1772
1773 int gladLoadGLES2UserPtr( GLADuserptrloadfunc load, void *userptr) {
1774     int version;
1775
1776     glad_glGetString = (PFNGLGETSTRINGPROC) load(userptr, "glGetString");
1777     if(glad_glGetString == NULL) return 0;
1778     if(glad_glGetString(GL_VERSION) == NULL) return 0;
1779     version = glad_gl_find_core_gles2();
1780
1781     glad_gl_load_GL_ES_VERSION_2_0(load, userptr);
1782
1783     if (!glad_gl_find_extensions_gles2(version)) return 0;
1784
1785
1786
1787     return version;
1788 }
1789
1790
1791 int gladLoadGLES2( GLADloadfunc load) {
1792     return gladLoadGLES2UserPtr( glad_gl_get_proc_from_userptr, GLAD_GNUC_EXTENSION (void*) load);
1793 }
1794
1795
1796
1797
1798
1799
1800 #ifdef __cplusplus
1801 }
1802 #endif
1803
1804 #endif /* GLAD_GLES2_IMPLEMENTATION */
1805
```

27.4 vulkan.h

```
1
28 #ifndef GLAD_VULKAN_H_
29 #define GLAD_VULKAN_H_
30
31 #ifdef VULKAN_H_
32   #error  header already included (API: vulkan), remove previous include!
33 #endif
34 #define VULKAN_H_ 1
35
36 #ifdef VULKAN_CORE_H_
37   #error  header already included (API: vulkan), remove previous include!
38 #endif
39 #define VULKAN_CORE_H_ 1
40
41
42 #define GLAD_VULKAN
43 #define GLAD_OPTION_VULKAN_HEADER_ONLY
44
45 #ifdef __cplusplus
46 extern "C" {
47 #endif
48
49 #ifndef GLAD_PLATFORM_H_
50 #define GLAD_PLATFORM_H_
51
52 #ifndef GLAD_PLATFORM_WIN32
53   #if defined(_WIN32) || defined(__WIN32__) || defined(WIN32) || defined(__MINGW32__)
54     #define GLAD_PLATFORM_WIN32 1
55   #else
56     #define GLAD_PLATFORM_WIN32 0
57   #endif
58 #endif
59
60 #ifndef GLAD_PLATFORM_APPLE
61   #ifdef __APPLE__
62     #define GLAD_PLATFORM_APPLE 1
63   #else
64     #define GLAD_PLATFORM_APPLE 0
65   #endif
66 #endif
67
68 #ifndef GLAD_PLATFORM_EMSCRIPTEN
69   #ifdef __EMSCRIPTEN__
70     #define GLAD_PLATFORM_EMSCRIPTEN 1
71   #else
72     #define GLAD_PLATFORM_EMSCRIPTEN 0
73   #endif
74 #endif
75
76 #ifndef GLAD_PLATFORM_UWP
77   #if defined(_MSC_VER) && !defined(GLAD_INTERNAL_HAVE_WINAPIFAMILY)
78     #ifdef __has_include
79       #if __has_include(<winapifamily.h>)
80         #define GLAD_INTERNAL_HAVE_WINAPIFAMILY 1
81       #endif
82     #elif _MSC_VER >= 1700 && !_USING_V110_SDK71_
83       #define GLAD_INTERNAL_HAVE_WINAPIFAMILY 1
84     #endif
85   #endif
86
87   #ifdef GLAD_INTERNAL_HAVE_WINAPIFAMILY
88     #include <winapifamily.h>
89     #if !WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_DESKTOP) &&
       WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_APP)
90       #define GLAD_PLATFORM_UWP 1
91     #endif
92   #endif
93
94   #ifndef GLAD_PLATFORM_UWP
95     #define GLAD_PLATFORM_UWP 0
96   #endif
97 #endif
98
99 #ifdef __GNUC__
100   #define GLAD_GNUC_EXTENSION __extension__
101 #else
102   #define GLAD_GNUC_EXTENSION
103 #endif
104
105 #ifndef GLAD_API_CALL
106   #if defined(GLAD_API_CALL_EXPORT)
107     #if GLAD_PLATFORM_WIN32 || defined(__CYGWIN__)
108       #if defined(GLAD_API_CALL_EXPORT_BUILD)
109         #if defined(__GNUC__)
110           #define GLAD_API_CALL __attribute__ ((dllexport)) extern
```

```
111         #else
112           #define GLAD_API_CALL __declspec(dllexport) extern
113         #endif
114       #else
115         #if defined(__GNUC__)
116           #define GLAD_API_CALL __attribute__ ((dllimport)) extern
117         #else
118           #define GLAD_API_CALL __declspec(dllimport) extern
119         #endif
120       #endif
121     #elif defined(__GNUC__) && defined(GLAD_API_CALL_EXPORT_BUILD)
122       #define GLAD_API_CALL __attribute__ ((visibility ("default"))) extern
123     #else
124       #define GLAD_API_CALL extern
125     #endif
126   #else
127     #define GLAD_API_CALL extern
128   #endif
129 #endif
130
131 #ifdef APIENTRY
132   #define GLAD_API_PTR APIENTRY
133 #elif GLAD_PLATFORM_WIN32
134   #define GLAD_API_PTR __stdcall
135 #else
136   #define GLAD_API_PTR
137 #endif
138
139 #ifndef GLAPI
140 #define GLAPI GLAD_API_CALL
141 #endif
142
143 #ifndef GLAPIENTRY
144 #define GLAPIENTRY GLAD_API_PTR
145 #endif
146
147 #define GLAD_MAKE_VERSION(major, minor) (major * 10000 + minor)
148 #define GLAD_VERSION_MAJOR(version) (version / 10000)
149 #define GLAD_VERSION_MINOR(version) (version % 10000)
150
151 #define GLAD_GENERATOR_VERSION "2.0.0-beta"
152
153 typedef void (*GLADapiproc)(void);
154
155 typedef GLADapiproc (*GLADloadfunc)(const char *name);
156 typedef GLADapiproc (*GLADuserptrloadfunc)(void *userptr, const char *name);
157
158 typedef void (*GLADprecallback)(const char *name, GLADapiproc apiproc, int len_args, ...);
159 typedef void (*GLADpostcallback)(void *ret, const char *name, GLADapiproc apiproc, int len_args, ...);
160
161 #endif /* GLAD_PLATFORM_H_ */
162
163 #define VK_ATTACHMENT_UNUSED (~0U)
164 #define VK_EXT_DEBUG_REPORT_EXTENSION_NAME "VK_EXT_debug_report"
165 #define VK_EXT_DEBUG_REPORT_SPEC_VERSION 9
166 #define VK_FALSE 0
167 #define VK_KHR_SURFACE_EXTENSION_NAME "VK_KHR_surface"
168 #define VK_KHR_SURFACE_SPEC_VERSION 25
169 #define VK_KHR_SWAPCHAIN_EXTENSION_NAME "VK_KHR_swapchain"
170 #define VK_KHR_SWAPCHAIN_SPEC_VERSION 70
171 #define VK_LOD_CLAMP_NONE 1000.0f
172 #define VK_LUID_SIZE 8
173 #define VK_MAX_DESCRIPTION_SIZE 256
174 #define VK_MAX_DEVICE_GROUP_SIZE 32
175 #define VK_MAX_EXTENSION_NAME_SIZE 256
176 #define VK_MAX_MEMORY_HEAPS 16
177 #define VK_MAX_MEMORY_TYPES 32
178 #define VK_MAX_PHYSICAL_DEVICE_NAME_SIZE 256
179 #define VK_QUEUE_FAMILY_EXTERNAL (~0U-1)
180 #define VK_QUEUE_FAMILY_IGNORED (~0U)
181 #define VK_REMAINING_ARRAY_LAYERS (~0U)
182 #define VK_REMAINING_MIP_LEVELS (~0U)
183 #define VK_SUBPASS_EXTERNAL (~0U)
184 #define VK_TRUE 1
185 #define VK_UUID_SIZE 16
186 #define VK_WHOLE_SIZE (~0ULL)
187
188
189 /* */
190 /* File: vk_platform.h */
191 /* */
192 /*
193 ** Copyright (c) 2014-2020 The Khronos Group Inc.
194 **
195 ** SPDX-License-Identifier: Apache-2.0
196 */
197
```

```
198
199 #ifndef VK_PLATFORM_H_
200 #define VK_PLATFORM_H_
201
202 #ifdef __cplusplus
203 extern "C"
204 {
205 #endif /* __cplusplus */
206
207 /*
208 ***************************************************************************************************
209 *   Platform-specific directives and type declarations
210 ***************************************************************************************************
211 */
212
213 /* Platform-specific calling convention macros.
214  *
215  * Platforms should define these so that Vulkan clients call Vulkan commands
216  * with the same calling conventions that the Vulkan implementation expects.
217  *
218  * VKAPI_ATTR - Placed before the return type in function declarations.
219  *              Useful for C++11 and GCC/Clang-style function attribute syntax.
220  * VKAPI_CALL - Placed after the return type in function declarations.
221  *              Useful for MSVC-style calling convention syntax.
222  * VKAPI_PTR  - Placed between the '(' and '*' in function pointer types.
223  *
224  * Function declaration:  VKAPI_ATTR void VKAPI_CALL vkCommand(void);
225  * Function pointer type: typedef void (VKAPI_PTR *PFN_vkCommand)(void);
226  */
227 #if defined(_WIN32)
228     /* On Windows, Vulkan commands use the stdcall convention */
229     #define VKAPI_ATTR
230     #define VKAPI_CALL __stdcall
231     #define VKAPI_PTR  VKAPI_CALL
232 #elif defined(__ANDROID__) && defined(__ARM_ARCH) && __ARM_ARCH < 7
233     #error "Vulkan isn't supported for the 'armeabi' NDK ABI"
234 #elif defined(__ANDROID__) && defined(__ARM_ARCH) && __ARM_ARCH >= 7 && defined(__ARM_32BIT_STATE)
235     /* On Android 32-bit ARM targets, Vulkan functions use the "hardfloat" */
236     /* calling convention, i.e. float parameters are passed in registers. This */
237     /* is true even if the rest of the application passes floats on the stack, */
238     /* as it does by default when compiling for the armeabi-v7a NDK ABI. */
239     #define VKAPI_ATTR __attribute__((pcs("aapcs-vfp")))
240     #define VKAPI_CALL
241     #define VKAPI_PTR  VKAPI_ATTR
242 #else
243     /* On other platforms, use the default calling convention */
244     #define VKAPI_ATTR
245     #define VKAPI_CALL
246     #define VKAPI_PTR
247 #endif
248
249 #include <stddef.h>
250
251 #if !defined(VK_NO_STDINT_H)
252     #if defined(_MSC_VER) && (_MSC_VER < 1600)
253         typedef signed   __int8  int8_t;
254         typedef unsigned __int8  uint8_t;
255         typedef signed   __int16 int16_t;
256         typedef unsigned __int16 uint16_t;
257         typedef signed   __int32 int32_t;
258         typedef unsigned __int32 uint32_t;
259         typedef signed   __int64 int64_t;
260         typedef unsigned __int64 uint64_t;
261     #else
262         #include <stdint.h>
263     #endif
264 #endif /* !defined(VK_NO_STDINT_H) */
265
266 #ifdef __cplusplus
267 } /* extern "C" */
268 #endif /* __cplusplus */
269
270 #endif
271
272 #define VK_MAKE_VERSION(major, minor, patch) \
273     ((((uint32_t)(major)) << 22) | (((uint32_t)(minor)) << 12) | ((uint32_t)(patch)))
274
275 #define VK_VERSION_MAJOR(version) ((uint32_t)(version) >> 22)
276
277 #define VK_VERSION_MINOR(version) (((uint32_t)(version) >> 12) & 0x3ff)
278
279 #define VK_VERSION_PATCH(version) ((uint32_t)(version) & 0xfff)
280
281 /* DEPRECATED: This define has been removed. Specific version defines (e.g. VK_API_VERSION_1_0), or the
       VK_MAKE_VERSION macro, should be used instead. */
282 /*#define VK_API_VERSION VK_MAKE_VERSION(1, 0, 0) // Patch version should always be set to 0 */
283
```

```
284 /* Vulkan 1.0 version number */
285 #define VK_API_VERSION_1_0 VK_MAKE_VERSION(1, 0, 0)/* Patch version should always be set to 0 */
286
287 /* Vulkan 1.1 version number */
288 #define VK_API_VERSION_1_1 VK_MAKE_VERSION(1, 1, 0)/* Patch version should always be set to 0 */
289
290 /* Version of this file */
291 #define VK_HEADER_VERSION 152
292
293 /* Complete version of this file */
294 #define VK_HEADER_VERSION_COMPLETE VK_MAKE_VERSION(1, 2, VK_HEADER_VERSION)
295
296 #define VK_DEFINE_HANDLE(object) typedef struct object##_T* object;
297
298 #if !defined(VK_DEFINE_NON_DISPATCHABLE_HANDLE)
299 #if defined(__LP64__) || defined(_WIN64) || (defined(__x86_64__) && !defined(__ILP32__) ) ||
       defined(_M_X64) || defined(__ia64) || defined (_M_IA64) || defined(__aarch64__) ||
       defined(__powerpc64__)
300         #define VK_DEFINE_NON_DISPATCHABLE_HANDLE(object) typedef struct object##_T *object;
301 #else
302         #define VK_DEFINE_NON_DISPATCHABLE_HANDLE(object) typedef uint64_t object;
303 #endif
304 #endif
305
306 #define VK_NULL_HANDLE 0
307
308
309
310
311
312
313
314
315
316 VK_DEFINE_HANDLE(VkInstance)
317
318 VK_DEFINE_HANDLE(VkPhysicalDevice)
319
320 VK_DEFINE_HANDLE(VkDevice)
321
322 VK_DEFINE_HANDLE(VkQueue)
323
324 VK_DEFINE_HANDLE(VkCommandBuffer)
325
326 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkDeviceMemory)
327
328 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkCommandPool)
329
330 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkBuffer)
331
332 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkBufferView)
333
334 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkImage)
335
336 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkImageView)
337
338 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkShaderModule)
339
340 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkPipeline)
341
342 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkPipelineLayout)
343
344 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkSampler)
345
346 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkDescriptorSet)
347
348 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkDescriptorSetLayout)
349
350 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkDescriptorPool)
351
352 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkFence)
353
354 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkSemaphore)
355
356 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkEvent)
357
358 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkQueryPool)
359
360 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkFramebuffer)
361
362 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkRenderPass)
363
364 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkPipelineCache)
365
366 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkDescriptorUpdateTemplate)
367
368 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkSamplerYcbcrConversion)
```

```
369
370 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkSurfaceKHR)
371
372 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkSwapchainKHR)
373
374 VK_DEFINE_NON_DISPATCHABLE_HANDLE(VkDebugReportCallbackEXT)
375
376 typedef enum VkAttachmentLoadOp {
377     VK_ATTACHMENT_LOAD_OP_LOAD = 0,
378     VK_ATTACHMENT_LOAD_OP_CLEAR = 1,
379     VK_ATTACHMENT_LOAD_OP_DONT_CARE = 2
380 } VkAttachmentLoadOp;
381
382 typedef enum VkAttachmentStoreOp {
383     VK_ATTACHMENT_STORE_OP_STORE = 0,
384     VK_ATTACHMENT_STORE_OP_DONT_CARE = 1
385 } VkAttachmentStoreOp;
386
387 typedef enum VkBlendFactor {
388     VK_BLEND_FACTOR_ZERO = 0,
389     VK_BLEND_FACTOR_ONE = 1,
390     VK_BLEND_FACTOR_SRC_COLOR = 2,
391     VK_BLEND_FACTOR_ONE_MINUS_SRC_COLOR = 3,
392     VK_BLEND_FACTOR_DST_COLOR = 4,
393     VK_BLEND_FACTOR_ONE_MINUS_DST_COLOR = 5,
394     VK_BLEND_FACTOR_SRC_ALPHA = 6,
395     VK_BLEND_FACTOR_ONE_MINUS_SRC_ALPHA = 7,
396     VK_BLEND_FACTOR_DST_ALPHA = 8,
397     VK_BLEND_FACTOR_ONE_MINUS_DST_ALPHA = 9,
398     VK_BLEND_FACTOR_CONSTANT_COLOR = 10,
399     VK_BLEND_FACTOR_ONE_MINUS_CONSTANT_COLOR = 11,
400     VK_BLEND_FACTOR_CONSTANT_ALPHA = 12,
401     VK_BLEND_FACTOR_ONE_MINUS_CONSTANT_ALPHA = 13,
402     VK_BLEND_FACTOR_SRC_ALPHA_SATURATE = 14,
403     VK_BLEND_FACTOR_SRC1_COLOR = 15,
404     VK_BLEND_FACTOR_ONE_MINUS_SRC1_COLOR = 16,
405     VK_BLEND_FACTOR_SRC1_ALPHA = 17,
406     VK_BLEND_FACTOR_ONE_MINUS_SRC1_ALPHA = 18
407 } VkBlendFactor;
408
409 typedef enum VkBlendOp {
410     VK_BLEND_OP_ADD = 0,
411     VK_BLEND_OP_SUBTRACT = 1,
412     VK_BLEND_OP_REVERSE_SUBTRACT = 2,
413     VK_BLEND_OP_MIN = 3,
414     VK_BLEND_OP_MAX = 4
415 } VkBlendOp;
416
417 typedef enum VkBorderColor {
418     VK_BORDER_COLOR_FLOAT_TRANSPARENT_BLACK = 0,
419     VK_BORDER_COLOR_INT_TRANSPARENT_BLACK = 1,
420     VK_BORDER_COLOR_FLOAT_OPAQUE_BLACK = 2,
421     VK_BORDER_COLOR_INT_OPAQUE_BLACK = 3,
422     VK_BORDER_COLOR_FLOAT_OPAQUE_WHITE = 4,
423     VK_BORDER_COLOR_INT_OPAQUE_WHITE = 5
424 } VkBorderColor;
425
426
427
428
429 typedef enum VkPipelineCacheHeaderVersion {
430     VK_PIPELINE_CACHE_HEADER_VERSION_ONE = 1
431 } VkPipelineCacheHeaderVersion;
432
433
434
435
436 typedef enum VkDeviceQueueCreateFlagBits {
437     VK_DEVICE_QUEUE_CREATE_PROTECTED_BIT = 1
438 } VkDeviceQueueCreateFlagBits;
439
440 typedef enum VkBufferCreateFlagBits {
441     VK_BUFFER_CREATE_SPARSE_BINDING_BIT = 1,
442     VK_BUFFER_CREATE_SPARSE_RESIDENCY_BIT = 2,
443     VK_BUFFER_CREATE_SPARSE_ALIASED_BIT = 4,
444     VK_BUFFER_CREATE_PROTECTED_BIT = 8
445 } VkBufferCreateFlagBits;
446
447 typedef enum VkBufferUsageFlagBits {
448     VK_BUFFER_USAGE_TRANSFER_SRC_BIT = 1,
449     VK_BUFFER_USAGE_TRANSFER_DST_BIT = 2,
450     VK_BUFFER_USAGE_UNIFORM_TEXEL_BUFFER_BIT = 4,
451     VK_BUFFER_USAGE_STORAGE_TEXEL_BUFFER_BIT = 8,
452     VK_BUFFER_USAGE_UNIFORM_BUFFER_BIT = 16,
453     VK_BUFFER_USAGE_STORAGE_BUFFER_BIT = 32,
454     VK_BUFFER_USAGE_INDEX_BUFFER_BIT = 64,
455     VK_BUFFER_USAGE_VERTEX_BUFFER_BIT = 128,
```

```
456     VK_BUFFER_USAGE_INDIRECT_BUFFER_BIT = 256
457 } VkBufferUsageFlagBits;
458
459 typedef enum VkColorComponentFlagBits {
460     VK_COLOR_COMPONENT_R_BIT = 1,
461     VK_COLOR_COMPONENT_G_BIT = 2,
462     VK_COLOR_COMPONENT_B_BIT = 4,
463     VK_COLOR_COMPONENT_A_BIT = 8
464 } VkColorComponentFlagBits;
465
466 typedef enum VkComponentSwizzle {
467     VK_COMPONENT_SWIZZLE_IDENTITY = 0,
468     VK_COMPONENT_SWIZZLE_ZERO = 1,
469     VK_COMPONENT_SWIZZLE_ONE = 2,
470     VK_COMPONENT_SWIZZLE_R = 3,
471     VK_COMPONENT_SWIZZLE_G = 4,
472     VK_COMPONENT_SWIZZLE_B = 5,
473     VK_COMPONENT_SWIZZLE_A = 6
474 } VkComponentSwizzle;
475
476 typedef enum VkCommandPoolCreateFlagBits {
477     VK_COMMAND_POOL_CREATE_TRANSIENT_BIT = 1,
478     VK_COMMAND_POOL_CREATE_RESET_COMMAND_BUFFER_BIT = 2,
479     VK_COMMAND_POOL_CREATE_PROTECTED_BIT = 4
480 } VkCommandPoolCreateFlagBits;
481
482 typedef enum VkCommandPoolResetFlagBits {
483     VK_COMMAND_POOL_RESET_RELEASE_RESOURCES_BIT = 1
484 } VkCommandPoolResetFlagBits;
485
486 typedef enum VkCommandBufferResetFlagBits {
487     VK_COMMAND_BUFFER_RESET_RELEASE_RESOURCES_BIT = 1
488 } VkCommandBufferResetFlagBits;
489
490 typedef enum VkCommandBufferLevel {
491     VK_COMMAND_BUFFER_LEVEL_PRIMARY = 0,
492     VK_COMMAND_BUFFER_LEVEL_SECONDARY = 1
493 } VkCommandBufferLevel;
494
495 typedef enum VkCommandBufferUsageFlagBits {
496     VK_COMMAND_BUFFER_USAGE_ONE_TIME_SUBMIT_BIT = 1,
497     VK_COMMAND_BUFFER_USAGE_RENDER_PASS_CONTINUE_BIT = 2,
498     VK_COMMAND_BUFFER_USAGE_SIMULTANEOUS_USE_BIT = 4
499 } VkCommandBufferUsageFlagBits;
500
501 typedef enum VkCompareOp {
502     VK_COMPARE_OP_NEVER = 0,
503     VK_COMPARE_OP_LESS = 1,
504     VK_COMPARE_OP_EQUAL = 2,
505     VK_COMPARE_OP_LESS_OR_EQUAL = 3,
506     VK_COMPARE_OP_GREATER = 4,
507     VK_COMPARE_OP_NOT_EQUAL = 5,
508     VK_COMPARE_OP_GREATER_OR_EQUAL = 6,
509     VK_COMPARE_OP_ALWAYS = 7
510 } VkCompareOp;
511
512 typedef enum VkCullModeFlagBits {
513     VK_CULL_MODE_NONE = 0,
514     VK_CULL_MODE_FRONT_BIT = 1,
515     VK_CULL_MODE_BACK_BIT = 2,
516     VK_CULL_MODE_FRONT_AND_BACK = 0x00000003
517 } VkCullModeFlagBits;
518
519 typedef enum VkDescriptorType {
520     VK_DESCRIPTOR_TYPE_SAMPLER = 0,
521     VK_DESCRIPTOR_TYPE_COMBINED_IMAGE_SAMPLER = 1,
522     VK_DESCRIPTOR_TYPE_SAMPLED_IMAGE = 2,
523     VK_DESCRIPTOR_TYPE_STORAGE_IMAGE = 3,
524     VK_DESCRIPTOR_TYPE_UNIFORM_TEXEL_BUFFER = 4,
525     VK_DESCRIPTOR_TYPE_STORAGE_TEXEL_BUFFER = 5,
526     VK_DESCRIPTOR_TYPE_UNIFORM_BUFFER = 6,
527     VK_DESCRIPTOR_TYPE_STORAGE_BUFFER = 7,
528     VK_DESCRIPTOR_TYPE_UNIFORM_BUFFER_DYNAMIC = 8,
529     VK_DESCRIPTOR_TYPE_STORAGE_BUFFER_DYNAMIC = 9,
530     VK_DESCRIPTOR_TYPE_INPUT_ATTACHMENT = 10
531 } VkDescriptorType;
532
533 typedef enum VkDynamicState {
534     VK_DYNAMIC_STATE_VIEWPORT = 0,
535     VK_DYNAMIC_STATE_SCISSOR = 1,
536     VK_DYNAMIC_STATE_LINE_WIDTH = 2,
537     VK_DYNAMIC_STATE_DEPTH_BIAS = 3,
538     VK_DYNAMIC_STATE_BLEND_CONSTANTS = 4,
539     VK_DYNAMIC_STATE_DEPTH_BOUNDS = 5,
540     VK_DYNAMIC_STATE_STENCIL_COMPARE_MASK = 6,
541     VK_DYNAMIC_STATE_STENCIL_WRITE_MASK = 7,
542     VK_DYNAMIC_STATE_STENCIL_REFERENCE = 8
```

```
543 } VkDynamicState;
544
545 typedef enum VkFenceCreateFlagBits {
546     VK_FENCE_CREATE_SIGNALED_BIT = 1
547 } VkFenceCreateFlagBits;
548
549 typedef enum VkPolygonMode {
550     VK_POLYGON_MODE_FILL = 0,
551     VK_POLYGON_MODE_LINE = 1,
552     VK_POLYGON_MODE_POINT = 2
553 } VkPolygonMode;
554
555 typedef enum VkFormat {
556     VK_FORMAT_UNDEFINED = 0,
557     VK_FORMAT_R4G4_UNORM_PACK8 = 1,
558     VK_FORMAT_R4G4B4A4_UNORM_PACK16 = 2,
559     VK_FORMAT_B4G4R4A4_UNORM_PACK16 = 3,
560     VK_FORMAT_R5G6B5_UNORM_PACK16 = 4,
561     VK_FORMAT_B5G6R5_UNORM_PACK16 = 5,
562     VK_FORMAT_R5G5B5A1_UNORM_PACK16 = 6,
563     VK_FORMAT_B5G5R5A1_UNORM_PACK16 = 7,
564     VK_FORMAT_A1R5G5B5_UNORM_PACK16 = 8,
565     VK_FORMAT_R8_UNORM = 9,
566     VK_FORMAT_R8_SNORM = 10,
567     VK_FORMAT_R8_USCALED = 11,
568     VK_FORMAT_R8_SSCALED = 12,
569     VK_FORMAT_R8_UINT = 13,
570     VK_FORMAT_R8_SINT = 14,
571     VK_FORMAT_R8_SRGB = 15,
572     VK_FORMAT_R8G8_UNORM = 16,
573     VK_FORMAT_R8G8_SNORM = 17,
574     VK_FORMAT_R8G8_USCALED = 18,
575     VK_FORMAT_R8G8_SSCALED = 19,
576     VK_FORMAT_R8G8_UINT = 20,
577     VK_FORMAT_R8G8_SINT = 21,
578     VK_FORMAT_R8G8_SRGB = 22,
579     VK_FORMAT_R8G8B8_UNORM = 23,
580     VK_FORMAT_R8G8B8_SNORM = 24,
581     VK_FORMAT_R8G8B8_USCALED = 25,
582     VK_FORMAT_R8G8B8_SSCALED = 26,
583     VK_FORMAT_R8G8B8_UINT = 27,
584     VK_FORMAT_R8G8B8_SINT = 28,
585     VK_FORMAT_R8G8B8_SRGB = 29,
586     VK_FORMAT_B8G8R8_UNORM = 30,
587     VK_FORMAT_B8G8R8_SNORM = 31,
588     VK_FORMAT_B8G8R8_USCALED = 32,
589     VK_FORMAT_B8G8R8_SSCALED = 33,
590     VK_FORMAT_B8G8R8_UINT = 34,
591     VK_FORMAT_B8G8R8_SINT = 35,
592     VK_FORMAT_B8G8R8_SRGB = 36,
593     VK_FORMAT_R8G8B8A8_UNORM = 37,
594     VK_FORMAT_R8G8B8A8_SNORM = 38,
595     VK_FORMAT_R8G8B8A8_USCALED = 39,
596     VK_FORMAT_R8G8B8A8_SSCALED = 40,
597     VK_FORMAT_R8G8B8A8_UINT = 41,
598     VK_FORMAT_R8G8B8A8_SINT = 42,
599     VK_FORMAT_R8G8B8A8_SRGB = 43,
600     VK_FORMAT_B8G8R8A8_UNORM = 44,
601     VK_FORMAT_B8G8R8A8_SNORM = 45,
602     VK_FORMAT_B8G8R8A8_USCALED = 46,
603     VK_FORMAT_B8G8R8A8_SSCALED = 47,
604     VK_FORMAT_B8G8R8A8_UINT = 48,
605     VK_FORMAT_B8G8R8A8_SINT = 49,
606     VK_FORMAT_B8G8R8A8_SRGB = 50,
607     VK_FORMAT_A8B8G8R8_UNORM_PACK32 = 51,
608     VK_FORMAT_A8B8G8R8_SNORM_PACK32 = 52,
609     VK_FORMAT_A8B8G8R8_USCALED_PACK32 = 53,
610     VK_FORMAT_A8B8G8R8_SSCALED_PACK32 = 54,
611     VK_FORMAT_A8B8G8R8_UINT_PACK32 = 55,
612     VK_FORMAT_A8B8G8R8_SINT_PACK32 = 56,
613     VK_FORMAT_A8B8G8R8_SRGB_PACK32 = 57,
614     VK_FORMAT_A2R10G10B10_UNORM_PACK32 = 58,
615     VK_FORMAT_A2R10G10B10_SNORM_PACK32 = 59,
616     VK_FORMAT_A2R10G10B10_USCALED_PACK32 = 60,
617     VK_FORMAT_A2R10G10B10_SSCALED_PACK32 = 61,
618     VK_FORMAT_A2R10G10B10_UINT_PACK32 = 62,
619     VK_FORMAT_A2R10G10B10_SINT_PACK32 = 63,
620     VK_FORMAT_A2B10G10R10_UNORM_PACK32 = 64,
621     VK_FORMAT_A2B10G10R10_SNORM_PACK32 = 65,
622     VK_FORMAT_A2B10G10R10_USCALED_PACK32 = 66,
623     VK_FORMAT_A2B10G10R10_SSCALED_PACK32 = 67,
624     VK_FORMAT_A2B10G10R10_UINT_PACK32 = 68,
625     VK_FORMAT_A2B10G10R10_SINT_PACK32 = 69,
626     VK_FORMAT_R16_UNORM = 70,
627     VK_FORMAT_R16_SNORM = 71,
628     VK_FORMAT_R16_USCALED = 72,
629     VK_FORMAT_R16_SSCALED = 73,
```

```
630     VK_FORMAT_R16_UINT = 74,
631     VK_FORMAT_R16_SINT = 75,
632     VK_FORMAT_R16_SFLOAT = 76,
633     VK_FORMAT_R16G16_UNORM = 77,
634     VK_FORMAT_R16G16_SNORM = 78,
635     VK_FORMAT_R16G16_USCALED = 79,
636     VK_FORMAT_R16G16_SSCALED = 80,
637     VK_FORMAT_R16G16_UINT = 81,
638     VK_FORMAT_R16G16_SINT = 82,
639     VK_FORMAT_R16G16_SFLOAT = 83,
640     VK_FORMAT_R16G16B16_UNORM = 84,
641     VK_FORMAT_R16G16B16_SNORM = 85,
642     VK_FORMAT_R16G16B16_USCALED = 86,
643     VK_FORMAT_R16G16B16_SSCALED = 87,
644     VK_FORMAT_R16G16B16_UINT = 88,
645     VK_FORMAT_R16G16B16_SINT = 89,
646     VK_FORMAT_R16G16B16_SFLOAT = 90,
647     VK_FORMAT_R16G16B16A16_UNORM = 91,
648     VK_FORMAT_R16G16B16A16_SNORM = 92,
649     VK_FORMAT_R16G16B16A16_USCALED = 93,
650     VK_FORMAT_R16G16B16A16_SSCALED = 94,
651     VK_FORMAT_R16G16B16A16_UINT = 95,
652     VK_FORMAT_R16G16B16A16_SINT = 96,
653     VK_FORMAT_R16G16B16A16_SFLOAT = 97,
654     VK_FORMAT_R32_UINT = 98,
655     VK_FORMAT_R32_SINT = 99,
656     VK_FORMAT_R32_SFLOAT = 100,
657     VK_FORMAT_R32G32_UINT = 101,
658     VK_FORMAT_R32G32_SINT = 102,
659     VK_FORMAT_R32G32_SFLOAT = 103,
660     VK_FORMAT_R32G32B32_UINT = 104,
661     VK_FORMAT_R32G32B32_SINT = 105,
662     VK_FORMAT_R32G32B32_SFLOAT = 106,
663     VK_FORMAT_R32G32B32A32_UINT = 107,
664     VK_FORMAT_R32G32B32A32_SINT = 108,
665     VK_FORMAT_R32G32B32A32_SFLOAT = 109,
666     VK_FORMAT_R64_UINT = 110,
667     VK_FORMAT_R64_SINT = 111,
668     VK_FORMAT_R64_SFLOAT = 112,
669     VK_FORMAT_R64G64_UINT = 113,
670     VK_FORMAT_R64G64_SINT = 114,
671     VK_FORMAT_R64G64_SFLOAT = 115,
672     VK_FORMAT_R64G64B64_UINT = 116,
673     VK_FORMAT_R64G64B64_SINT = 117,
674     VK_FORMAT_R64G64B64_SFLOAT = 118,
675     VK_FORMAT_R64G64B64A64_UINT = 119,
676     VK_FORMAT_R64G64B64A64_SINT = 120,
677     VK_FORMAT_R64G64B64A64_SFLOAT = 121,
678     VK_FORMAT_B10G11R11_UFLOAT_PACK32 = 122,
679     VK_FORMAT_E5B9G9R9_UFLOAT_PACK32 = 123,
680     VK_FORMAT_D16_UNORM = 124,
681     VK_FORMAT_X8_D24_UNORM_PACK32 = 125,
682     VK_FORMAT_D32_SFLOAT = 126,
683     VK_FORMAT_S8_UINT = 127,
684     VK_FORMAT_D16_UNORM_S8_UINT = 128,
685     VK_FORMAT_D24_UNORM_S8_UINT = 129,
686     VK_FORMAT_D32_SFLOAT_S8_UINT = 130,
687     VK_FORMAT_BC1_RGB_UNORM_BLOCK = 131,
688     VK_FORMAT_BC1_RGB_SRGB_BLOCK = 132,
689     VK_FORMAT_BC1_RGBA_UNORM_BLOCK = 133,
690     VK_FORMAT_BC1_RGBA_SRGB_BLOCK = 134,
691     VK_FORMAT_BC2_UNORM_BLOCK = 135,
692     VK_FORMAT_BC2_SRGB_BLOCK = 136,
693     VK_FORMAT_BC3_UNORM_BLOCK = 137,
694     VK_FORMAT_BC3_SRGB_BLOCK = 138,
695     VK_FORMAT_BC4_UNORM_BLOCK = 139,
696     VK_FORMAT_BC4_SNORM_BLOCK = 140,
697     VK_FORMAT_BC5_UNORM_BLOCK = 141,
698     VK_FORMAT_BC5_SNORM_BLOCK = 142,
699     VK_FORMAT_BC6H_UFLOAT_BLOCK = 143,
700     VK_FORMAT_BC6H_SFLOAT_BLOCK = 144,
701     VK_FORMAT_BC7_UNORM_BLOCK = 145,
702     VK_FORMAT_BC7_SRGB_BLOCK = 146,
703     VK_FORMAT_ETC2_R8G8B8_UNORM_BLOCK = 147,
704     VK_FORMAT_ETC2_R8G8B8_SRGB_BLOCK = 148,
705     VK_FORMAT_ETC2_R8G8B8A1_UNORM_BLOCK = 149,
706     VK_FORMAT_ETC2_R8G8B8A1_SRGB_BLOCK = 150,
707     VK_FORMAT_ETC2_R8G8B8A8_UNORM_BLOCK = 151,
708     VK_FORMAT_ETC2_R8G8B8A8_SRGB_BLOCK = 152,
709     VK_FORMAT_EAC_R11_UNORM_BLOCK = 153,
710     VK_FORMAT_EAC_R11_SNORM_BLOCK = 154,
711     VK_FORMAT_EAC_R11G11_UNORM_BLOCK = 155,
712     VK_FORMAT_EAC_R11G11_SNORM_BLOCK = 156,
713     VK_FORMAT_ASTC_4x4_UNORM_BLOCK = 157,
714     VK_FORMAT_ASTC_4x4_SRGB_BLOCK = 158,
715     VK_FORMAT_ASTC_5x4_UNORM_BLOCK = 159,
716     VK_FORMAT_ASTC_5x4_SRGB_BLOCK = 160,
```

```
717     VK_FORMAT_ASTC_5x5_UNORM_BLOCK = 161,
718     VK_FORMAT_ASTC_5x5_SRGB_BLOCK = 162,
719     VK_FORMAT_ASTC_6x5_UNORM_BLOCK = 163,
720     VK_FORMAT_ASTC_6x5_SRGB_BLOCK = 164,
721     VK_FORMAT_ASTC_6x6_UNORM_BLOCK = 165,
722     VK_FORMAT_ASTC_6x6_SRGB_BLOCK = 166,
723     VK_FORMAT_ASTC_8x5_UNORM_BLOCK = 167,
724     VK_FORMAT_ASTC_8x5_SRGB_BLOCK = 168,
725     VK_FORMAT_ASTC_8x6_UNORM_BLOCK = 169,
726     VK_FORMAT_ASTC_8x6_SRGB_BLOCK = 170,
727     VK_FORMAT_ASTC_8x8_UNORM_BLOCK = 171,
728     VK_FORMAT_ASTC_8x8_SRGB_BLOCK = 172,
729     VK_FORMAT_ASTC_10x5_UNORM_BLOCK = 173,
730     VK_FORMAT_ASTC_10x5_SRGB_BLOCK = 174,
731     VK_FORMAT_ASTC_10x6_UNORM_BLOCK = 175,
732     VK_FORMAT_ASTC_10x6_SRGB_BLOCK = 176,
733     VK_FORMAT_ASTC_10x8_UNORM_BLOCK = 177,
734     VK_FORMAT_ASTC_10x8_SRGB_BLOCK = 178,
735     VK_FORMAT_ASTC_10x10_UNORM_BLOCK = 179,
736     VK_FORMAT_ASTC_10x10_SRGB_BLOCK = 180,
737     VK_FORMAT_ASTC_12x10_UNORM_BLOCK = 181,
738     VK_FORMAT_ASTC_12x10_SRGB_BLOCK = 182,
739     VK_FORMAT_ASTC_12x12_UNORM_BLOCK = 183,
740     VK_FORMAT_ASTC_12x12_SRGB_BLOCK = 184,
741     VK_FORMAT_G8B8G8R8_422_UNORM = 1000156000,
742     VK_FORMAT_B8G8R8G8_422_UNORM = 1000156001,
743     VK_FORMAT_G8_B8_R8_3PLANE_420_UNORM = 1000156002,
744     VK_FORMAT_G8_B8R8_2PLANE_420_UNORM = 1000156003,
745     VK_FORMAT_G8_B8_R8_3PLANE_422_UNORM = 1000156004,
746     VK_FORMAT_G8_B8R8_2PLANE_422_UNORM = 1000156005,
747     VK_FORMAT_G8_B8_R8_3PLANE_444_UNORM = 1000156006,
748     VK_FORMAT_R10X6_UNORM_PACK16 = 1000156007,
749     VK_FORMAT_R10X6G10X6_UNORM_2PACK16 = 1000156008,
750     VK_FORMAT_R10X6G10X6B10X6A10X6_UNORM_4PACK16 = 1000156009,
751     VK_FORMAT_G10X6B10X6G10X6R10X6_422_UNORM_4PACK16 = 1000156010,
752     VK_FORMAT_B10X6G10X6R10X6G10X6_422_UNORM_4PACK16 = 1000156011,
753     VK_FORMAT_G10X6_B10X6_R10X6_3PLANE_420_UNORM_3PACK16 = 1000156012,
754     VK_FORMAT_G10X6_B10X6R10X6_2PLANE_420_UNORM_3PACK16 = 1000156013,
755     VK_FORMAT_G10X6_B10X6_R10X6_3PLANE_422_UNORM_3PACK16 = 1000156014,
756     VK_FORMAT_G10X6_B10X6R10X6_2PLANE_422_UNORM_3PACK16 = 1000156015,
757     VK_FORMAT_G10X6_B10X6_R10X6_3PLANE_444_UNORM_3PACK16 = 1000156016,
758     VK_FORMAT_R12X4_UNORM_PACK16 = 1000156017,
759     VK_FORMAT_R12X4G12X4_UNORM_2PACK16 = 1000156018,
760     VK_FORMAT_R12X4G12X4B12X4A12X4_UNORM_4PACK16 = 1000156019,
761     VK_FORMAT_G12X4B12X4G12X4R12X4_422_UNORM_4PACK16 = 1000156020,
762     VK_FORMAT_B12X4G12X4R12X4G12X4_422_UNORM_4PACK16 = 1000156021,
763     VK_FORMAT_G12X4_B12X4_R12X4_3PLANE_420_UNORM_3PACK16 = 1000156022,
764     VK_FORMAT_G12X4_B12X4R12X4_2PLANE_420_UNORM_3PACK16 = 1000156023,
765     VK_FORMAT_G12X4_B12X4_R12X4_3PLANE_422_UNORM_3PACK16 = 1000156024,
766     VK_FORMAT_G12X4_B12X4R12X4_2PLANE_422_UNORM_3PACK16 = 1000156025,
767     VK_FORMAT_G12X4_B12X4_R12X4_3PLANE_444_UNORM_3PACK16 = 1000156026,
768     VK_FORMAT_G16B16G16R16_422_UNORM = 1000156027,
769     VK_FORMAT_B16G16R16G16_422_UNORM = 1000156028,
770     VK_FORMAT_G16_B16_R16_3PLANE_420_UNORM = 1000156029,
771     VK_FORMAT_G16_B16R16_2PLANE_420_UNORM = 1000156030,
772     VK_FORMAT_G16_B16_R16_3PLANE_422_UNORM = 1000156031,
773     VK_FORMAT_G16_B16R16_2PLANE_422_UNORM = 1000156032,
774     VK_FORMAT_G16_B16_R16_3PLANE_444_UNORM = 1000156033
775 } VkFormat;
776
777 typedef enum VkFormatFeatureFlagBits {
778     VK_FORMAT_FEATURE_SAMPLED_IMAGE_BIT = 1,
779     VK_FORMAT_FEATURE_STORAGE_IMAGE_BIT = 2,
780     VK_FORMAT_FEATURE_STORAGE_IMAGE_ATOMIC_BIT = 4,
781     VK_FORMAT_FEATURE_UNIFORM_TEXEL_BUFFER_BIT = 8,
782     VK_FORMAT_FEATURE_STORAGE_TEXEL_BUFFER_BIT = 16,
783     VK_FORMAT_FEATURE_STORAGE_TEXEL_BUFFER_ATOMIC_BIT = 32,
784     VK_FORMAT_FEATURE_VERTEX_BUFFER_BIT = 64,
785     VK_FORMAT_FEATURE_COLOR_ATTACHMENT_BIT = 128,
786     VK_FORMAT_FEATURE_COLOR_ATTACHMENT_BLEND_BIT = 256,
787     VK_FORMAT_FEATURE_DEPTH_STENCIL_ATTACHMENT_BIT = 512,
788     VK_FORMAT_FEATURE_BLIT_SRC_BIT = 1024,
789     VK_FORMAT_FEATURE_BLIT_DST_BIT = 2048,
790     VK_FORMAT_FEATURE_SAMPLED_IMAGE_FILTER_LINEAR_BIT = 4096,
791     VK_FORMAT_FEATURE_TRANSFER_SRC_BIT = 16384,
792     VK_FORMAT_FEATURE_TRANSFER_DST_BIT = 32768,
793     VK_FORMAT_FEATURE_MIDPOINT_CHROMA_SAMPLES_BIT = 131072,
794     VK_FORMAT_FEATURE_SAMPLED_IMAGE_YCBCR_CONVERSION_LINEAR_FILTER_BIT = 262144,
795     VK_FORMAT_FEATURE_SAMPLED_IMAGE_YCBCR_CONVERSION_SEPARATE_RECONSTRUCTION_FILTER_BIT = 524288,
796     VK_FORMAT_FEATURE_SAMPLED_IMAGE_YCBCR_CONVERSION_CHROMA_RECONSTRUCTION_EXPLICIT_BIT = 1048576,
797     VK_FORMAT_FEATURE_SAMPLED_IMAGE_YCBCR_CONVERSION_CHROMA_RECONSTRUCTION_EXPLICIT_FORCEABLE_BIT =
      2097152,
798     VK_FORMAT_FEATURE_DISJOINT_BIT = 4194304,
799     VK_FORMAT_FEATURE_COSITED_CHROMA_SAMPLES_BIT = 8388608
800 } VkFormatFeatureFlagBits;
801
802 typedef enum VkFrontFace {
```

```
803     VK_FRONT_FACE_COUNTER_CLOCKWISE = 0,
804     VK_FRONT_FACE_CLOCKWISE = 1
805 } VkFrontFace;
806
807 typedef enum VkImageAspectFlagBits {
808     VK_IMAGE_ASPECT_COLOR_BIT = 1,
809     VK_IMAGE_ASPECT_DEPTH_BIT = 2,
810     VK_IMAGE_ASPECT_STENCIL_BIT = 4,
811     VK_IMAGE_ASPECT_METADATA_BIT = 8,
812     VK_IMAGE_ASPECT_PLANE_0_BIT = 16,
813     VK_IMAGE_ASPECT_PLANE_1_BIT = 32,
814     VK_IMAGE_ASPECT_PLANE_2_BIT = 64
815 } VkImageAspectFlagBits;
816
817 typedef enum VkImageCreateFlagBits {
818     VK_IMAGE_CREATE_SPARSE_BINDING_BIT = 1,
819     VK_IMAGE_CREATE_SPARSE_RESIDENCY_BIT = 2,
820     VK_IMAGE_CREATE_SPARSE_ALIASED_BIT = 4,
821     VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT = 8,
822     VK_IMAGE_CREATE_CUBE_COMPATIBLE_BIT = 16,
823     VK_IMAGE_CREATE_ALIAS_BIT = 1024,
824     VK_IMAGE_CREATE_SPLIT_INSTANCE_BIND_REGIONS_BIT = 64,
825     VK_IMAGE_CREATE_2D_ARRAY_COMPATIBLE_BIT = 32,
826     VK_IMAGE_CREATE_BLOCK_TEXEL_VIEW_COMPATIBLE_BIT = 128,
827     VK_IMAGE_CREATE_EXTENDED_USAGE_BIT = 256,
828     VK_IMAGE_CREATE_PROTECTED_BIT = 2048,
829     VK_IMAGE_CREATE_DISJOINT_BIT = 512
830 } VkImageCreateFlagBits;
831
832 typedef enum VkImageLayout {
833     VK_IMAGE_LAYOUT_UNDEFINED = 0,
834     VK_IMAGE_LAYOUT_GENERAL = 1,
835     VK_IMAGE_LAYOUT_COLOR_ATTACHMENT_OPTIMAL = 2,
836     VK_IMAGE_LAYOUT_DEPTH_STENCIL_ATTACHMENT_OPTIMAL = 3,
837     VK_IMAGE_LAYOUT_DEPTH_STENCIL_READ_ONLY_OPTIMAL = 4,
838     VK_IMAGE_LAYOUT_SHADER_READ_ONLY_OPTIMAL = 5,
839     VK_IMAGE_LAYOUT_TRANSFER_SRC_OPTIMAL = 6,
840     VK_IMAGE_LAYOUT_TRANSFER_DST_OPTIMAL = 7,
841     VK_IMAGE_LAYOUT_PREINITIALIZED = 8,
842     VK_IMAGE_LAYOUT_DEPTH_READ_ONLY_STENCIL_ATTACHMENT_OPTIMAL = 1000117000,
843     VK_IMAGE_LAYOUT_DEPTH_ATTACHMENT_STENCIL_READ_ONLY_OPTIMAL = 1000117001,
844     VK_IMAGE_LAYOUT_PRESENT_SRC_KHR = 1000001002
845 } VkImageLayout;
846
847 typedef enum VkImageTiling {
848     VK_IMAGE_TILING_OPTIMAL = 0,
849     VK_IMAGE_TILING_LINEAR = 1
850 } VkImageTiling;
851
852 typedef enum VkImageType {
853     VK_IMAGE_TYPE_1D = 0,
854     VK_IMAGE_TYPE_2D = 1,
855     VK_IMAGE_TYPE_3D = 2
856 } VkImageType;
857
858 typedef enum VkImageUsageFlagBits {
859     VK_IMAGE_USAGE_TRANSFER_SRC_BIT = 1,
860     VK_IMAGE_USAGE_TRANSFER_DST_BIT = 2,
861     VK_IMAGE_USAGE_SAMPLED_BIT = 4,
862     VK_IMAGE_USAGE_STORAGE_BIT = 8,
863     VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT = 16,
864     VK_IMAGE_USAGE_DEPTH_STENCIL_ATTACHMENT_BIT = 32,
865     VK_IMAGE_USAGE_TRANSIENT_ATTACHMENT_BIT = 64,
866     VK_IMAGE_USAGE_INPUT_ATTACHMENT_BIT = 128
867 } VkImageUsageFlagBits;
868
869
870 typedef enum VkImageViewType {
871     VK_IMAGE_VIEW_TYPE_1D = 0,
872     VK_IMAGE_VIEW_TYPE_2D = 1,
873     VK_IMAGE_VIEW_TYPE_3D = 2,
874     VK_IMAGE_VIEW_TYPE_CUBE = 3,
875     VK_IMAGE_VIEW_TYPE_1D_ARRAY = 4,
876     VK_IMAGE_VIEW_TYPE_2D_ARRAY = 5,
877     VK_IMAGE_VIEW_TYPE_CUBE_ARRAY = 6
878 } VkImageViewType;
879
880 typedef enum VkSharingMode {
881     VK_SHARING_MODE_EXCLUSIVE = 0,
882     VK_SHARING_MODE_CONCURRENT = 1
883 } VkSharingMode;
884
885 typedef enum VkIndexType {
886     VK_INDEX_TYPE_UINT16 = 0,
887     VK_INDEX_TYPE_UINT32 = 1
888 } VkIndexType;
889
```

```
890 typedef enum VkLogicOp {
891     VK_LOGIC_OP_CLEAR = 0,
892     VK_LOGIC_OP_AND = 1,
893     VK_LOGIC_OP_AND_REVERSE = 2,
894     VK_LOGIC_OP_COPY = 3,
895     VK_LOGIC_OP_AND_INVERTED = 4,
896     VK_LOGIC_OP_NO_OP = 5,
897     VK_LOGIC_OP_XOR = 6,
898     VK_LOGIC_OP_OR = 7,
899     VK_LOGIC_OP_NOR = 8,
900     VK_LOGIC_OP_EQUIVALENT = 9,
901     VK_LOGIC_OP_INVERT = 10,
902     VK_LOGIC_OP_OR_REVERSE = 11,
903     VK_LOGIC_OP_COPY_INVERTED = 12,
904     VK_LOGIC_OP_OR_INVERTED = 13,
905     VK_LOGIC_OP_NAND = 14,
906     VK_LOGIC_OP_SET = 15
907 } VkLogicOp;
908
909 typedef enum VkMemoryHeapFlagBits {
910     VK_MEMORY_HEAP_DEVICE_LOCAL_BIT = 1,
911     VK_MEMORY_HEAP_MULTI_INSTANCE_BIT = 2
912 } VkMemoryHeapFlagBits;
913
914 typedef enum VkAccessFlagBits {
915     VK_ACCESS_INDIRECT_COMMAND_READ_BIT = 1,
916     VK_ACCESS_INDEX_READ_BIT = 2,
917     VK_ACCESS_VERTEX_ATTRIBUTE_READ_BIT = 4,
918     VK_ACCESS_UNIFORM_READ_BIT = 8,
919     VK_ACCESS_INPUT_ATTACHMENT_READ_BIT = 16,
920     VK_ACCESS_SHADER_READ_BIT = 32,
921     VK_ACCESS_SHADER_WRITE_BIT = 64,
922     VK_ACCESS_COLOR_ATTACHMENT_READ_BIT = 128,
923     VK_ACCESS_COLOR_ATTACHMENT_WRITE_BIT = 256,
924     VK_ACCESS_DEPTH_STENCIL_ATTACHMENT_READ_BIT = 512,
925     VK_ACCESS_DEPTH_STENCIL_ATTACHMENT_WRITE_BIT = 1024,
926     VK_ACCESS_TRANSFER_READ_BIT = 2048,
927     VK_ACCESS_TRANSFER_WRITE_BIT = 4096,
928     VK_ACCESS_HOST_READ_BIT = 8192,
929     VK_ACCESS_HOST_WRITE_BIT = 16384,
930     VK_ACCESS_MEMORY_READ_BIT = 32768,
931     VK_ACCESS_MEMORY_WRITE_BIT = 65536
932 } VkAccessFlagBits;
933
934 typedef enum VkMemoryPropertyFlagBits {
935     VK_MEMORY_PROPERTY_DEVICE_LOCAL_BIT = 1,
936     VK_MEMORY_PROPERTY_HOST_VISIBLE_BIT = 2,
937     VK_MEMORY_PROPERTY_HOST_COHERENT_BIT = 4,
938     VK_MEMORY_PROPERTY_HOST_CACHED_BIT = 8,
939     VK_MEMORY_PROPERTY_LAZILY_ALLOCATED_BIT = 16,
940     VK_MEMORY_PROPERTY_PROTECTED_BIT = 32
941 } VkMemoryPropertyFlagBits;
942
943 typedef enum VkPhysicalDeviceType {
944     VK_PHYSICAL_DEVICE_TYPE_OTHER = 0,
945     VK_PHYSICAL_DEVICE_TYPE_INTEGRATED_GPU = 1,
946     VK_PHYSICAL_DEVICE_TYPE_DISCRETE_GPU = 2,
947     VK_PHYSICAL_DEVICE_TYPE_VIRTUAL_GPU = 3,
948     VK_PHYSICAL_DEVICE_TYPE_CPU = 4
949 } VkPhysicalDeviceType;
950
951 typedef enum VkPipelineBindPoint {
952     VK_PIPELINE_BIND_POINT_GRAPHICS = 0,
953     VK_PIPELINE_BIND_POINT_COMPUTE = 1
954 } VkPipelineBindPoint;
955
956 typedef enum VkPipelineCreateFlagBits {
957     VK_PIPELINE_CREATE_DISABLE_OPTIMIZATION_BIT = 1,
958     VK_PIPELINE_CREATE_ALLOW_DERIVATIVES_BIT = 2,
959     VK_PIPELINE_CREATE_DERIVATIVE_BIT = 4,
960     VK_PIPELINE_CREATE_VIEW_INDEX_FROM_DEVICE_INDEX_BIT = 8,
961     VK_PIPELINE_CREATE_DISPATCH_BASE_BIT = 16,
962     VK_PIPELINE_CREATE_DISPATCH_BASE = VK_PIPELINE_CREATE_DISPATCH_BASE_BIT
963 } VkPipelineCreateFlagBits;
964
965 typedef enum VkPrimitiveTopology {
966     VK_PRIMITIVE_TOPOLOGY_POINT_LIST = 0,
967     VK_PRIMITIVE_TOPOLOGY_LINE_LIST = 1,
968     VK_PRIMITIVE_TOPOLOGY_LINE_STRIP = 2,
969     VK_PRIMITIVE_TOPOLOGY_TRIANGLE_LIST = 3,
970     VK_PRIMITIVE_TOPOLOGY_TRIANGLE_STRIP = 4,
971     VK_PRIMITIVE_TOPOLOGY_TRIANGLE_FAN = 5,
972     VK_PRIMITIVE_TOPOLOGY_LINE_LIST_WITH_ADJACENCY = 6,
973     VK_PRIMITIVE_TOPOLOGY_LINE_STRIP_WITH_ADJACENCY = 7,
974     VK_PRIMITIVE_TOPOLOGY_TRIANGLE_LIST_WITH_ADJACENCY = 8,
975     VK_PRIMITIVE_TOPOLOGY_TRIANGLE_STRIP_WITH_ADJACENCY = 9,
976     VK_PRIMITIVE_TOPOLOGY_PATCH_LIST = 10
```

```
977 } VkPrimitiveTopology;
978
979 typedef enum VkQueryControlFlagBits {
980     VK_QUERY_CONTROL_PRECISE_BIT = 1
981 } VkQueryControlFlagBits;
982
983 typedef enum VkQueryPipelineStatisticFlagBits {
984     VK_QUERY_PIPELINE_STATISTIC_INPUT_ASSEMBLY_VERTICES_BIT = 1,
985     VK_QUERY_PIPELINE_STATISTIC_INPUT_ASSEMBLY_PRIMITIVES_BIT = 2,
986     VK_QUERY_PIPELINE_STATISTIC_VERTEX_SHADER_INVOCATIONS_BIT = 4,
987     VK_QUERY_PIPELINE_STATISTIC_GEOMETRY_SHADER_INVOCATIONS_BIT = 8,
988     VK_QUERY_PIPELINE_STATISTIC_GEOMETRY_SHADER_PRIMITIVES_BIT = 16,
989     VK_QUERY_PIPELINE_STATISTIC_CLIPPING_INVOCATIONS_BIT = 32,
990     VK_QUERY_PIPELINE_STATISTIC_CLIPPING_PRIMITIVES_BIT = 64,
991     VK_QUERY_PIPELINE_STATISTIC_FRAGMENT_SHADER_INVOCATIONS_BIT = 128,
992     VK_QUERY_PIPELINE_STATISTIC_TESSELLATION_CONTROL_SHADER_PATCHES_BIT = 256,
993     VK_QUERY_PIPELINE_STATISTIC_TESSELLATION_EVALUATION_SHADER_INVOCATIONS_BIT = 512,
994     VK_QUERY_PIPELINE_STATISTIC_COMPUTE_SHADER_INVOCATIONS_BIT = 1024
995 } VkQueryPipelineStatisticFlagBits;
996
997 typedef enum VkQueryResultFlagBits {
998     VK_QUERY_RESULT_64_BIT = 1,
999     VK_QUERY_RESULT_WAIT_BIT = 2,
1000     VK_QUERY_RESULT_WITH_AVAILABILITY_BIT = 4,
1001     VK_QUERY_RESULT_PARTIAL_BIT = 8
1002 } VkQueryResultFlagBits;
1003
1004 typedef enum VkQueryType {
1005     VK_QUERY_TYPE_OCCLUSION = 0,
1006     VK_QUERY_TYPE_PIPELINE_STATISTICS = 1,
1007     VK_QUERY_TYPE_TIMESTAMP = 2
1008 } VkQueryType;
1009
1010 typedef enum VkQueueFlagBits {
1011     VK_QUEUE_GRAPHICS_BIT = 1,
1012     VK_QUEUE_COMPUTE_BIT = 2,
1013     VK_QUEUE_TRANSFER_BIT = 4,
1014     VK_QUEUE_SPARSE_BINDING_BIT = 8,
1015     VK_QUEUE_PROTECTED_BIT = 16
1016 } VkQueueFlagBits;
1017
1018 typedef enum VkSubpassContents {
1019     VK_SUBPASS_CONTENTS_INLINE = 0,
1020     VK_SUBPASS_CONTENTS_SECONDARY_COMMAND_BUFFERS = 1
1021 } VkSubpassContents;
1022
1023 typedef enum VkResult {
1024     VK_SUCCESS = 0,
1025     VK_NOT_READY = 1,
1026     VK_TIMEOUT = 2,
1027     VK_EVENT_SET = 3,
1028     VK_EVENT_RESET = 4,
1029     VK_INCOMPLETE = 5,
1030     VK_ERROR_OUT_OF_HOST_MEMORY = -1,
1031     VK_ERROR_OUT_OF_DEVICE_MEMORY = -2,
1032     VK_ERROR_INITIALIZATION_FAILED = -3,
1033     VK_ERROR_DEVICE_LOST = -4,
1034     VK_ERROR_MEMORY_MAP_FAILED = -5,
1035     VK_ERROR_LAYER_NOT_PRESENT = -6,
1036     VK_ERROR_EXTENSION_NOT_PRESENT = -7,
1037     VK_ERROR_FEATURE_NOT_PRESENT = -8,
1038     VK_ERROR_INCOMPATIBLE_DRIVER = -9,
1039     VK_ERROR_TOO_MANY_OBJECTS = -10,
1040     VK_ERROR_FORMAT_NOT_SUPPORTED = -11,
1041     VK_ERROR_FRAGMENTED_POOL = -12,
1042     VK_ERROR_UNKNOWN = -13,
1043     VK_ERROR_OUT_OF_POOL_MEMORY = -1000069000,
1044     VK_ERROR_INVALID_EXTERNAL_HANDLE = -1000072003,
1045     VK_ERROR_SURFACE_LOST_KHR = -1000000000,
1046     VK_ERROR_NATIVE_WINDOW_IN_USE_KHR = -1000000001,
1047     VK_SUBOPTIMAL_KHR = 1000001003,
1048     VK_ERROR_OUT_OF_DATE_KHR = -1000001004,
1049     VK_ERROR_VALIDATION_FAILED_EXT = -1000011001
1050 } VkResult;
1051
1052 typedef enum VkShaderStageFlagBits {
1053     VK_SHADER_STAGE_VERTEX_BIT = 1,
1054     VK_SHADER_STAGE_TESSELLATION_CONTROL_BIT = 2,
1055     VK_SHADER_STAGE_TESSELLATION_EVALUATION_BIT = 4,
1056     VK_SHADER_STAGE_GEOMETRY_BIT = 8,
1057     VK_SHADER_STAGE_FRAGMENT_BIT = 16,
1058     VK_SHADER_STAGE_COMPUTE_BIT = 32,
1059     VK_SHADER_STAGE_ALL_GRAPHICS = 0x0000001F,
1060     VK_SHADER_STAGE_ALL = 0x7FFFFFFF
1061 } VkShaderStageFlagBits;
1062
1063 typedef enum VkSparseMemoryBindFlagBits {
```

```
1064     VK_SPARSE_MEMORY_BIND_METADATA_BIT = 1
1065 } VkSparseMemoryBindFlagBits;
1066
1067 typedef enum VkStencilFaceFlagBits {
1068     VK_STENCIL_FACE_FRONT_BIT = 1,
1069     VK_STENCIL_FACE_BACK_BIT = 2,
1070     VK_STENCIL_FACE_FRONT_AND_BACK = 0x00000003,
1071     VK_STENCIL_FRONT_AND_BACK = VK_STENCIL_FACE_FRONT_AND_BACK
1072 } VkStencilFaceFlagBits;
1073
1074 typedef enum VkStencilOp {
1075     VK_STENCIL_OP_KEEP = 0,
1076     VK_STENCIL_OP_ZERO = 1,
1077     VK_STENCIL_OP_REPLACE = 2,
1078     VK_STENCIL_OP_INCREMENT_AND_CLAMP = 3,
1079     VK_STENCIL_OP_DECREMENT_AND_CLAMP = 4,
1080     VK_STENCIL_OP_INVERT = 5,
1081     VK_STENCIL_OP_INCREMENT_AND_WRAP = 6,
1082     VK_STENCIL_OP_DECREMENT_AND_WRAP = 7
1083 } VkStencilOp;
1084
1085 typedef enum VkStructureType {
1086     VK_STRUCTURE_TYPE_APPLICATION_INFO = 0,
1087     VK_STRUCTURE_TYPE_INSTANCE_CREATE_INFO = 1,
1088     VK_STRUCTURE_TYPE_DEVICE_QUEUE_CREATE_INFO = 2,
1089     VK_STRUCTURE_TYPE_DEVICE_CREATE_INFO = 3,
1090     VK_STRUCTURE_TYPE_SUBMIT_INFO = 4,
1091     VK_STRUCTURE_TYPE_MEMORY_ALLOCATE_INFO = 5,
1092     VK_STRUCTURE_TYPE_MAPPED_MEMORY_RANGE = 6,
1093     VK_STRUCTURE_TYPE_BIND_SPARSE_INFO = 7,
1094     VK_STRUCTURE_TYPE_FENCE_CREATE_INFO = 8,
1095     VK_STRUCTURE_TYPE_SEMAPHORE_CREATE_INFO = 9,
1096     VK_STRUCTURE_TYPE_EVENT_CREATE_INFO = 10,
1097     VK_STRUCTURE_TYPE_QUERY_POOL_CREATE_INFO = 11,
1098     VK_STRUCTURE_TYPE_BUFFER_CREATE_INFO = 12,
1099     VK_STRUCTURE_TYPE_BUFFER_VIEW_CREATE_INFO = 13,
1100     VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO = 14,
1101     VK_STRUCTURE_TYPE_IMAGE_VIEW_CREATE_INFO = 15,
1102     VK_STRUCTURE_TYPE_SHADER_MODULE_CREATE_INFO = 16,
1103     VK_STRUCTURE_TYPE_PIPELINE_CACHE_CREATE_INFO = 17,
1104     VK_STRUCTURE_TYPE_PIPELINE_SHADER_STAGE_CREATE_INFO = 18,
1105     VK_STRUCTURE_TYPE_PIPELINE_VERTEX_INPUT_STATE_CREATE_INFO = 19,
1106     VK_STRUCTURE_TYPE_PIPELINE_INPUT_ASSEMBLY_STATE_CREATE_INFO = 20,
1107     VK_STRUCTURE_TYPE_PIPELINE_TESSELLATION_STATE_CREATE_INFO = 21,
1108     VK_STRUCTURE_TYPE_PIPELINE_VIEWPORT_STATE_CREATE_INFO = 22,
1109     VK_STRUCTURE_TYPE_PIPELINE_RASTERIZATION_STATE_CREATE_INFO = 23,
1110     VK_STRUCTURE_TYPE_PIPELINE_MULTISAMPLE_STATE_CREATE_INFO = 24,
1111     VK_STRUCTURE_TYPE_PIPELINE_DEPTH_STENCIL_STATE_CREATE_INFO = 25,
1112     VK_STRUCTURE_TYPE_PIPELINE_COLOR_BLEND_STATE_CREATE_INFO = 26,
1113     VK_STRUCTURE_TYPE_PIPELINE_DYNAMIC_STATE_CREATE_INFO = 27,
1114     VK_STRUCTURE_TYPE_GRAPHICS_PIPELINE_CREATE_INFO = 28,
1115     VK_STRUCTURE_TYPE_COMPUTE_PIPELINE_CREATE_INFO = 29,
1116     VK_STRUCTURE_TYPE_PIPELINE_LAYOUT_CREATE_INFO = 30,
1117     VK_STRUCTURE_TYPE_SAMPLER_CREATE_INFO = 31,
1118     VK_STRUCTURE_TYPE_DESCRIPTOR_SET_LAYOUT_CREATE_INFO = 32,
1119     VK_STRUCTURE_TYPE_DESCRIPTOR_POOL_CREATE_INFO = 33,
1120     VK_STRUCTURE_TYPE_DESCRIPTOR_SET_ALLOCATE_INFO = 34,
1121     VK_STRUCTURE_TYPE_WRITE_DESCRIPTOR_SET = 35,
1122     VK_STRUCTURE_TYPE_COPY_DESCRIPTOR_SET = 36,
1123     VK_STRUCTURE_TYPE_FRAMEBUFFER_CREATE_INFO = 37,
1124     VK_STRUCTURE_TYPE_RENDER_PASS_CREATE_INFO = 38,
1125     VK_STRUCTURE_TYPE_COMMAND_POOL_CREATE_INFO = 39,
1126     VK_STRUCTURE_TYPE_COMMAND_BUFFER_ALLOCATE_INFO = 40,
1127     VK_STRUCTURE_TYPE_COMMAND_BUFFER_INHERITANCE_INFO = 41,
1128     VK_STRUCTURE_TYPE_COMMAND_BUFFER_BEGIN_INFO = 42,
1129     VK_STRUCTURE_TYPE_RENDER_PASS_BEGIN_INFO = 43,
1130     VK_STRUCTURE_TYPE_BUFFER_MEMORY_BARRIER = 44,
1131     VK_STRUCTURE_TYPE_IMAGE_MEMORY_BARRIER = 45,
1132     VK_STRUCTURE_TYPE_MEMORY_BARRIER = 46,
1133     VK_STRUCTURE_TYPE_LOADER_INSTANCE_CREATE_INFO = 47,
1134     VK_STRUCTURE_TYPE_LOADER_DEVICE_CREATE_INFO = 48,
1135     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_SUBGROUP_PROPERTIES = 1000094000,
1136     VK_STRUCTURE_TYPE_BIND_BUFFER_MEMORY_INFO = 1000157000,
1137     VK_STRUCTURE_TYPE_BIND_IMAGE_MEMORY_INFO = 1000157001,
1138     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_16BIT_STORAGE_FEATURES = 1000083000,
1139     VK_STRUCTURE_TYPE_MEMORY_DEDICATED_REQUIREMENTS = 1000127000,
1140     VK_STRUCTURE_TYPE_MEMORY_DEDICATED_ALLOCATE_INFO = 1000127001,
1141     VK_STRUCTURE_TYPE_MEMORY_ALLOCATE_FLAGS_INFO = 1000060000,
1142     VK_STRUCTURE_TYPE_DEVICE_GROUP_RENDER_PASS_BEGIN_INFO = 1000060003,
1143     VK_STRUCTURE_TYPE_DEVICE_GROUP_COMMAND_BUFFER_BEGIN_INFO = 1000060004,
1144     VK_STRUCTURE_TYPE_DEVICE_GROUP_SUBMIT_INFO = 1000060005,
1145     VK_STRUCTURE_TYPE_DEVICE_GROUP_BIND_SPARSE_INFO = 1000060006,
1146     VK_STRUCTURE_TYPE_BIND_BUFFER_MEMORY_DEVICE_GROUP_INFO = 1000060013,
1147     VK_STRUCTURE_TYPE_BIND_IMAGE_MEMORY_DEVICE_GROUP_INFO = 1000060014,
1148     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_GROUP_PROPERTIES = 1000070000,
1149     VK_STRUCTURE_TYPE_DEVICE_GROUP_DEVICE_CREATE_INFO = 1000070001,
1150     VK_STRUCTURE_TYPE_BUFFER_MEMORY_REQUIREMENTS_INFO_2 = 1000146000,
```

```
1151     VK_STRUCTURE_TYPE_IMAGE_MEMORY_REQUIREMENTS_INFO_2 = 1000146001,
1152     VK_STRUCTURE_TYPE_IMAGE_SPARSE_MEMORY_REQUIREMENTS_INFO_2 = 1000146002,
1153     VK_STRUCTURE_TYPE_MEMORY_REQUIREMENTS_2 = 1000146003,
1154     VK_STRUCTURE_TYPE_SPARSE_IMAGE_MEMORY_REQUIREMENTS_2 = 1000146004,
1155     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_FEATURES_2 = 1000059000,
1156     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_PROPERTIES_2 = 1000059001,
1157     VK_STRUCTURE_TYPE_FORMAT_PROPERTIES_2 = 1000059002,
1158     VK_STRUCTURE_TYPE_IMAGE_FORMAT_PROPERTIES_2 = 1000059003,
1159     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_IMAGE_FORMAT_INFO_2 = 1000059004,
1160     VK_STRUCTURE_TYPE_QUEUE_FAMILY_PROPERTIES_2 = 1000059005,
1161     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_MEMORY_PROPERTIES_2 = 1000059006,
1162     VK_STRUCTURE_TYPE_SPARSE_IMAGE_FORMAT_PROPERTIES_2 = 1000059007,
1163     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_SPARSE_IMAGE_FORMAT_INFO_2 = 1000059008,
1164     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_POINT_CLIPPING_PROPERTIES = 1000117000,
1165     VK_STRUCTURE_TYPE_RENDER_PASS_INPUT_ATTACHMENT_ASPECT_CREATE_INFO = 1000117001,
1166     VK_STRUCTURE_TYPE_IMAGE_VIEW_USAGE_CREATE_INFO = 1000117002,
1167     VK_STRUCTURE_TYPE_PIPELINE_TESSELLATION_DOMAIN_ORIGIN_STATE_CREATE_INFO = 1000117003,
1168     VK_STRUCTURE_TYPE_RENDER_PASS_MULTIVIEW_CREATE_INFO = 1000053000,
1169     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_MULTIVIEW_FEATURES = 1000053001,
1170     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_MULTIVIEW_PROPERTIES = 1000053002,
1171     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_VARIABLE_POINTERS_FEATURES = 1000120000,
1172     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_VARIABLE_POINTER_FEATURES =
       VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_VARIABLE_POINTERS_FEATURES,
1173     VK_STRUCTURE_TYPE_PROTECTED_SUBMIT_INFO = 1000145000,
1174     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_PROTECTED_MEMORY_FEATURES = 1000145001,
1175     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_PROTECTED_MEMORY_PROPERTIES = 1000145002,
1176     VK_STRUCTURE_TYPE_DEVICE_QUEUE_INFO_2 = 1000145003,
1177     VK_STRUCTURE_TYPE_SAMPLER_YCBCR_CONVERSION_CREATE_INFO = 1000156000,
1178     VK_STRUCTURE_TYPE_SAMPLER_YCBCR_CONVERSION_INFO = 1000156001,
1179     VK_STRUCTURE_TYPE_BIND_IMAGE_PLANE_MEMORY_INFO = 1000156002,
1180     VK_STRUCTURE_TYPE_IMAGE_PLANE_MEMORY_REQUIREMENTS_INFO = 1000156003,
1181     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_SAMPLER_YCBCR_CONVERSION_FEATURES = 1000156004,
1182     VK_STRUCTURE_TYPE_SAMPLER_YCBCR_CONVERSION_IMAGE_FORMAT_PROPERTIES = 1000156005,
1183     VK_STRUCTURE_TYPE_DESCRIPTOR_UPDATE_TEMPLATE_CREATE_INFO = 1000085000,
1184     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_EXTERNAL_IMAGE_FORMAT_INFO = 1000071000,
1185     VK_STRUCTURE_TYPE_EXTERNAL_IMAGE_FORMAT_PROPERTIES = 1000071001,
1186     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_EXTERNAL_BUFFER_INFO = 1000071002,
1187     VK_STRUCTURE_TYPE_EXTERNAL_BUFFER_PROPERTIES = 1000071003,
1188     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_ID_PROPERTIES = 1000071004,
1189     VK_STRUCTURE_TYPE_EXTERNAL_MEMORY_BUFFER_CREATE_INFO = 1000072000,
1190     VK_STRUCTURE_TYPE_EXTERNAL_MEMORY_IMAGE_CREATE_INFO = 1000072001,
1191     VK_STRUCTURE_TYPE_EXPORT_MEMORY_ALLOCATE_INFO = 1000072002,
1192     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_EXTERNAL_FENCE_INFO = 1000112000,
1193     VK_STRUCTURE_TYPE_EXTERNAL_FENCE_PROPERTIES = 1000112001,
1194     VK_STRUCTURE_TYPE_EXPORT_FENCE_CREATE_INFO = 1000113000,
1195     VK_STRUCTURE_TYPE_EXPORT_SEMAPHORE_CREATE_INFO = 1000077000,
1196     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_EXTERNAL_SEMAPHORE_INFO = 1000076000,
1197     VK_STRUCTURE_TYPE_EXTERNAL_SEMAPHORE_PROPERTIES = 1000076001,
1198     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_MAINTENANCE_3_PROPERTIES = 1000168000,
1199     VK_STRUCTURE_TYPE_DESCRIPTOR_SET_LAYOUT_SUPPORT = 1000168001,
1200     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_SHADER_DRAW_PARAMETERS_FEATURES = 1000063000,
1201     VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_SHADER_DRAW_PARAMETER_FEATURES =
       VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_SHADER_DRAW_PARAMETERS_FEATURES,
1202     VK_STRUCTURE_TYPE_SWAPCHAIN_CREATE_INFO_KHR = 1000001000,
1203     VK_STRUCTURE_TYPE_PRESENT_INFO_KHR = 1000001001,
1204     VK_STRUCTURE_TYPE_DEVICE_GROUP_PRESENT_CAPABILITIES_KHR = 1000060007,
1205     VK_STRUCTURE_TYPE_IMAGE_SWAPCHAIN_CREATE_INFO_KHR = 1000060008,
1206     VK_STRUCTURE_TYPE_BIND_IMAGE_MEMORY_SWAPCHAIN_INFO_KHR = 1000060009,
1207     VK_STRUCTURE_TYPE_ACQUIRE_NEXT_IMAGE_INFO_KHR = 1000060010,
1208     VK_STRUCTURE_TYPE_DEVICE_GROUP_PRESENT_INFO_KHR = 1000060011,
1209     VK_STRUCTURE_TYPE_DEVICE_GROUP_SWAPCHAIN_CREATE_INFO_KHR = 1000060012,
1210     VK_STRUCTURE_TYPE_DEBUG_REPORT_CALLBACK_CREATE_INFO_EXT = 1000011000,
1211     VK_STRUCTURE_TYPE_DEBUG_REPORT_CREATE_INFO_EXT =
       VK_STRUCTURE_TYPE_DEBUG_REPORT_CALLBACK_CREATE_INFO_EXT
1212 } VkStructureType;
1213
1214 typedef enum VkSystemAllocationScope {
1215     VK_SYSTEM_ALLOCATION_SCOPE_COMMAND = 0,
1216     VK_SYSTEM_ALLOCATION_SCOPE_OBJECT = 1,
1217     VK_SYSTEM_ALLOCATION_SCOPE_CACHE = 2,
1218     VK_SYSTEM_ALLOCATION_SCOPE_DEVICE = 3,
1219     VK_SYSTEM_ALLOCATION_SCOPE_INSTANCE = 4
1220 } VkSystemAllocationScope;
1221
1222 typedef enum VkInternalAllocationType {
1223     VK_INTERNAL_ALLOCATION_TYPE_EXECUTABLE = 0
1224 } VkInternalAllocationType;
1225
1226 typedef enum VkSamplerAddressMode {
1227     VK_SAMPLER_ADDRESS_MODE_REPEAT = 0,
1228     VK_SAMPLER_ADDRESS_MODE_MIRRORED_REPEAT = 1,
1229     VK_SAMPLER_ADDRESS_MODE_CLAMP_TO_EDGE = 2,
1230     VK_SAMPLER_ADDRESS_MODE_CLAMP_TO_BORDER = 3
1231 } VkSamplerAddressMode;
1232
1233 typedef enum VkFilter {
1234     VK_FILTER_NEAREST = 0,
```

```
1235     VK_FILTER_LINEAR = 1
1236 } VkFilter;
1237
1238 typedef enum VkSamplerMipmapMode {
1239     VK_SAMPLER_MIPMAP_MODE_NEAREST = 0,
1240     VK_SAMPLER_MIPMAP_MODE_LINEAR = 1
1241 } VkSamplerMipmapMode;
1242
1243 typedef enum VkVertexInputRate {
1244     VK_VERTEX_INPUT_RATE_VERTEX = 0,
1245     VK_VERTEX_INPUT_RATE_INSTANCE = 1
1246 } VkVertexInputRate;
1247
1248 typedef enum VkPipelineStageFlagBits {
1249     VK_PIPELINE_STAGE_TOP_OF_PIPE_BIT = 1,
1250     VK_PIPELINE_STAGE_DRAW_INDIRECT_BIT = 2,
1251     VK_PIPELINE_STAGE_VERTEX_INPUT_BIT = 4,
1252     VK_PIPELINE_STAGE_VERTEX_SHADER_BIT = 8,
1253     VK_PIPELINE_STAGE_TESSELLATION_CONTROL_SHADER_BIT = 16,
1254     VK_PIPELINE_STAGE_TESSELLATION_EVALUATION_SHADER_BIT = 32,
1255     VK_PIPELINE_STAGE_GEOMETRY_SHADER_BIT = 64,
1256     VK_PIPELINE_STAGE_FRAGMENT_SHADER_BIT = 128,
1257     VK_PIPELINE_STAGE_EARLY_FRAGMENT_TESTS_BIT = 256,
1258     VK_PIPELINE_STAGE_LATE_FRAGMENT_TESTS_BIT = 512,
1259     VK_PIPELINE_STAGE_COLOR_ATTACHMENT_OUTPUT_BIT = 1024,
1260     VK_PIPELINE_STAGE_COMPUTE_SHADER_BIT = 2048,
1261     VK_PIPELINE_STAGE_TRANSFER_BIT = 4096,
1262     VK_PIPELINE_STAGE_BOTTOM_OF_PIPE_BIT = 8192,
1263     VK_PIPELINE_STAGE_HOST_BIT = 16384,
1264     VK_PIPELINE_STAGE_ALL_GRAPHICS_BIT = 32768,
1265     VK_PIPELINE_STAGE_ALL_COMMANDS_BIT = 65536
1266 } VkPipelineStageFlagBits;
1267
1268 typedef enum VkSparseImageFormatFlagBits {
1269     VK_SPARSE_IMAGE_FORMAT_SINGLE_MIPTAIL_BIT = 1,
1270     VK_SPARSE_IMAGE_FORMAT_ALIGNED_MIP_SIZE_BIT = 2,
1271     VK_SPARSE_IMAGE_FORMAT_NONSTANDARD_BLOCK_SIZE_BIT = 4
1272 } VkSparseImageFormatFlagBits;
1273
1274 typedef enum VkSampleCountFlagBits {
1275     VK_SAMPLE_COUNT_1_BIT = 1,
1276     VK_SAMPLE_COUNT_2_BIT = 2,
1277     VK_SAMPLE_COUNT_4_BIT = 4,
1278     VK_SAMPLE_COUNT_8_BIT = 8,
1279     VK_SAMPLE_COUNT_16_BIT = 16,
1280     VK_SAMPLE_COUNT_32_BIT = 32,
1281     VK_SAMPLE_COUNT_64_BIT = 64
1282 } VkSampleCountFlagBits;
1283
1284 typedef enum VkAttachmentDescriptionFlagBits {
1285     VK_ATTACHMENT_DESCRIPTION_MAY_ALIAS_BIT = 1
1286 } VkAttachmentDescriptionFlagBits;
1287
1288 typedef enum VkDescriptorPoolCreateFlagBits {
1289     VK_DESCRIPTOR_POOL_CREATE_FREE_DESCRIPTOR_SET_BIT = 1
1290 } VkDescriptorPoolCreateFlagBits;
1291
1292 typedef enum VkDependencyFlagBits {
1293     VK_DEPENDENCY_BY_REGION_BIT = 1,
1294     VK_DEPENDENCY_DEVICE_GROUP_BIT = 4,
1295     VK_DEPENDENCY_VIEW_LOCAL_BIT = 2
1296 } VkDependencyFlagBits;
1297
1298 typedef enum VkObjectType {
1299     VK_OBJECT_TYPE_UNKNOWN = 0,
1300     VK_OBJECT_TYPE_INSTANCE = 1,
1301     VK_OBJECT_TYPE_PHYSICAL_DEVICE = 2,
1302     VK_OBJECT_TYPE_DEVICE = 3,
1303     VK_OBJECT_TYPE_QUEUE = 4,
1304     VK_OBJECT_TYPE_SEMAPHORE = 5,
1305     VK_OBJECT_TYPE_COMMAND_BUFFER = 6,
1306     VK_OBJECT_TYPE_FENCE = 7,
1307     VK_OBJECT_TYPE_DEVICE_MEMORY = 8,
1308     VK_OBJECT_TYPE_BUFFER = 9,
1309     VK_OBJECT_TYPE_IMAGE = 10,
1310     VK_OBJECT_TYPE_EVENT = 11,
1311     VK_OBJECT_TYPE_QUERY_POOL = 12,
1312     VK_OBJECT_TYPE_BUFFER_VIEW = 13,
1313     VK_OBJECT_TYPE_IMAGE_VIEW = 14,
1314     VK_OBJECT_TYPE_SHADER_MODULE = 15,
1315     VK_OBJECT_TYPE_PIPELINE_CACHE = 16,
1316     VK_OBJECT_TYPE_PIPELINE_LAYOUT = 17,
1317     VK_OBJECT_TYPE_RENDER_PASS = 18,
1318     VK_OBJECT_TYPE_PIPELINE = 19,
1319     VK_OBJECT_TYPE_DESCRIPTOR_SET_LAYOUT = 20,
1320     VK_OBJECT_TYPE_SAMPLER = 21,
1321     VK_OBJECT_TYPE_DESCRIPTOR_POOL = 22,
```

```
1322     VK_OBJECT_TYPE_DESCRIPTOR_SET = 23,
1323     VK_OBJECT_TYPE_FRAMEBUFFER = 24,
1324     VK_OBJECT_TYPE_COMMAND_POOL = 25,
1325     VK_OBJECT_TYPE_SAMPLER_YCBCR_CONVERSION = 1000156000,
1326     VK_OBJECT_TYPE_DESCRIPTOR_UPDATE_TEMPLATE = 1000085000,
1327     VK_OBJECT_TYPE_SURFACE_KHR = 1000000000,
1328     VK_OBJECT_TYPE_SWAPCHAIN_KHR = 1000001000,
1329     VK_OBJECT_TYPE_DEBUG_REPORT_CALLBACK_EXT = 1000011000
1330 } VkObjectType;
1331
1332 typedef enum VkDescriptorUpdateTemplateType {
1333     VK_DESCRIPTOR_UPDATE_TEMPLATE_TYPE_DESCRIPTOR_SET = 0
1334 } VkDescriptorUpdateTemplateType;
1335
1336
1337 typedef enum VkPointClippingBehavior {
1338     VK_POINT_CLIPPING_BEHAVIOR_ALL_CLIP_PLANES = 0,
1339     VK_POINT_CLIPPING_BEHAVIOR_USER_CLIP_PLANES_ONLY = 1
1340 } VkPointClippingBehavior;
1341
1342
1343 typedef enum VkColorSpaceKHR {
1344     VK_COLOR_SPACE_SRGB_NONLINEAR_KHR = 0,
1345     VK_COLORSPACE_SRGB_NONLINEAR_KHR = VK_COLOR_SPACE_SRGB_NONLINEAR_KHR
1346 } VkColorSpaceKHR;
1347
1348 typedef enum VkCompositeAlphaFlagBitsKHR {
1349     VK_COMPOSITE_ALPHA_OPAQUE_BIT_KHR = 1,
1350     VK_COMPOSITE_ALPHA_PRE_MULTIPLIED_BIT_KHR = 2,
1351     VK_COMPOSITE_ALPHA_POST_MULTIPLIED_BIT_KHR = 4,
1352     VK_COMPOSITE_ALPHA_INHERIT_BIT_KHR = 8
1353 } VkCompositeAlphaFlagBitsKHR;
1354
1355 typedef enum VkPresentModeKHR {
1356     VK_PRESENT_MODE_IMMEDIATE_KHR = 0,
1357     VK_PRESENT_MODE_MAILBOX_KHR = 1,
1358     VK_PRESENT_MODE_FIFO_KHR = 2,
1359     VK_PRESENT_MODE_FIFO_RELAXED_KHR = 3
1360 } VkPresentModeKHR;
1361
1362 typedef enum VkSurfaceTransformFlagBitsKHR {
1363     VK_SURFACE_TRANSFORM_IDENTITY_BIT_KHR = 1,
1364     VK_SURFACE_TRANSFORM_ROTATE_90_BIT_KHR = 2,
1365     VK_SURFACE_TRANSFORM_ROTATE_180_BIT_KHR = 4,
1366     VK_SURFACE_TRANSFORM_ROTATE_270_BIT_KHR = 8,
1367     VK_SURFACE_TRANSFORM_HORIZONTAL_MIRROR_BIT_KHR = 16,
1368     VK_SURFACE_TRANSFORM_HORIZONTAL_MIRROR_ROTATE_90_BIT_KHR = 32,
1369     VK_SURFACE_TRANSFORM_HORIZONTAL_MIRROR_ROTATE_180_BIT_KHR = 64,
1370     VK_SURFACE_TRANSFORM_HORIZONTAL_MIRROR_ROTATE_270_BIT_KHR = 128,
1371     VK_SURFACE_TRANSFORM_INHERIT_BIT_KHR = 256
1372 } VkSurfaceTransformFlagBitsKHR;
1373
1374 typedef enum VkDebugReportFlagBitsEXT {
1375     VK_DEBUG_REPORT_INFORMATION_BIT_EXT = 1,
1376     VK_DEBUG_REPORT_WARNING_BIT_EXT = 2,
1377     VK_DEBUG_REPORT_PERFORMANCE_WARNING_BIT_EXT = 4,
1378     VK_DEBUG_REPORT_ERROR_BIT_EXT = 8,
1379     VK_DEBUG_REPORT_DEBUG_BIT_EXT = 16
1380 } VkDebugReportFlagBitsEXT;
1381
1382 typedef enum VkDebugReportObjectTypeEXT {
1383     VK_DEBUG_REPORT_OBJECT_TYPE_UNKNOWN_EXT = 0,
1384     VK_DEBUG_REPORT_OBJECT_TYPE_INSTANCE_EXT = 1,
1385     VK_DEBUG_REPORT_OBJECT_TYPE_PHYSICAL_DEVICE_EXT = 2,
1386     VK_DEBUG_REPORT_OBJECT_TYPE_DEVICE_EXT = 3,
1387     VK_DEBUG_REPORT_OBJECT_TYPE_QUEUE_EXT = 4,
1388     VK_DEBUG_REPORT_OBJECT_TYPE_SEMAPHORE_EXT = 5,
1389     VK_DEBUG_REPORT_OBJECT_TYPE_COMMAND_BUFFER_EXT = 6,
1390     VK_DEBUG_REPORT_OBJECT_TYPE_FENCE_EXT = 7,
1391     VK_DEBUG_REPORT_OBJECT_TYPE_DEVICE_MEMORY_EXT = 8,
1392     VK_DEBUG_REPORT_OBJECT_TYPE_BUFFER_EXT = 9,
1393     VK_DEBUG_REPORT_OBJECT_TYPE_IMAGE_EXT = 10,
1394     VK_DEBUG_REPORT_OBJECT_TYPE_EVENT_EXT = 11,
1395     VK_DEBUG_REPORT_OBJECT_TYPE_QUERY_POOL_EXT = 12,
1396     VK_DEBUG_REPORT_OBJECT_TYPE_BUFFER_VIEW_EXT = 13,
1397     VK_DEBUG_REPORT_OBJECT_TYPE_IMAGE_VIEW_EXT = 14,
1398     VK_DEBUG_REPORT_OBJECT_TYPE_SHADER_MODULE_EXT = 15,
1399     VK_DEBUG_REPORT_OBJECT_TYPE_PIPELINE_CACHE_EXT = 16,
1400     VK_DEBUG_REPORT_OBJECT_TYPE_PIPELINE_LAYOUT_EXT = 17,
1401     VK_DEBUG_REPORT_OBJECT_TYPE_RENDER_PASS_EXT = 18,
1402     VK_DEBUG_REPORT_OBJECT_TYPE_PIPELINE_EXT = 19,
1403     VK_DEBUG_REPORT_OBJECT_TYPE_DESCRIPTOR_SET_LAYOUT_EXT = 20,
1404     VK_DEBUG_REPORT_OBJECT_TYPE_SAMPLER_EXT = 21,
1405     VK_DEBUG_REPORT_OBJECT_TYPE_DESCRIPTOR_POOL_EXT = 22,
1406     VK_DEBUG_REPORT_OBJECT_TYPE_DESCRIPTOR_SET_EXT = 23,
1407     VK_DEBUG_REPORT_OBJECT_TYPE_FRAMEBUFFER_EXT = 24,
1408     VK_DEBUG_REPORT_OBJECT_TYPE_COMMAND_POOL_EXT = 25,
```

```
1409     VK_DEBUG_REPORT_OBJECT_TYPE_SURFACE_KHR_EXT = 26,
1410     VK_DEBUG_REPORT_OBJECT_TYPE_SWAPCHAIN_KHR_EXT = 27,
1411     VK_DEBUG_REPORT_OBJECT_TYPE_DEBUG_REPORT_CALLBACK_EXT_EXT = 28,
1412     VK_DEBUG_REPORT_OBJECT_TYPE_DEBUG_REPORT_EXT =
       VK_DEBUG_REPORT_OBJECT_TYPE_DEBUG_REPORT_CALLBACK_EXT_EXT,
1413     VK_DEBUG_REPORT_OBJECT_TYPE_DISPLAY_KHR_EXT = 29,
1414     VK_DEBUG_REPORT_OBJECT_TYPE_DISPLAY_MODE_KHR_EXT = 30,
1415     VK_DEBUG_REPORT_OBJECT_TYPE_VALIDATION_CACHE_EXT_EXT = 33,
1416     VK_DEBUG_REPORT_OBJECT_TYPE_VALIDATION_CACHE_EXT =
       VK_DEBUG_REPORT_OBJECT_TYPE_VALIDATION_CACHE_EXT_EXT,
1417     VK_DEBUG_REPORT_OBJECT_TYPE_SAMPLER_YCBCR_CONVERSION_EXT = 1000156000,
1418     VK_DEBUG_REPORT_OBJECT_TYPE_DESCRIPTOR_UPDATE_TEMPLATE_EXT = 1000085000
1419 } VkDebugReportObjectTypeEXT;
1420
1421 typedef enum VkExternalMemoryHandleTypeFlagBits {
1422     VK_EXTERNAL_MEMORY_HANDLE_TYPE_OPAQUE_FD_BIT = 1,
1423     VK_EXTERNAL_MEMORY_HANDLE_TYPE_OPAQUE_WIN32_BIT = 2,
1424     VK_EXTERNAL_MEMORY_HANDLE_TYPE_OPAQUE_WIN32_KMT_BIT = 4,
1425     VK_EXTERNAL_MEMORY_HANDLE_TYPE_D3D11_TEXTURE_BIT = 8,
1426     VK_EXTERNAL_MEMORY_HANDLE_TYPE_D3D11_TEXTURE_KMT_BIT = 16,
1427     VK_EXTERNAL_MEMORY_HANDLE_TYPE_D3D12_HEAP_BIT = 32,
1428     VK_EXTERNAL_MEMORY_HANDLE_TYPE_D3D12_RESOURCE_BIT = 64
1429 } VkExternalMemoryHandleTypeFlagBits;
1430
1431 typedef enum VkExternalMemoryFeatureFlagBits {
1432     VK_EXTERNAL_MEMORY_FEATURE_DEDICATED_ONLY_BIT = 1,
1433     VK_EXTERNAL_MEMORY_FEATURE_EXPORTABLE_BIT = 2,
1434     VK_EXTERNAL_MEMORY_FEATURE_IMPORTABLE_BIT = 4
1435 } VkExternalMemoryFeatureFlagBits;
1436
1437 typedef enum VkExternalSemaphoreHandleTypeFlagBits {
1438     VK_EXTERNAL_SEMAPHORE_HANDLE_TYPE_OPAQUE_FD_BIT = 1,
1439     VK_EXTERNAL_SEMAPHORE_HANDLE_TYPE_OPAQUE_WIN32_BIT = 2,
1440     VK_EXTERNAL_SEMAPHORE_HANDLE_TYPE_OPAQUE_WIN32_KMT_BIT = 4,
1441     VK_EXTERNAL_SEMAPHORE_HANDLE_TYPE_D3D12_FENCE_BIT = 8,
1442     VK_EXTERNAL_SEMAPHORE_HANDLE_TYPE_D3D11_FENCE_BIT =
       VK_EXTERNAL_SEMAPHORE_HANDLE_TYPE_D3D12_FENCE_BIT,
1443     VK_EXTERNAL_SEMAPHORE_HANDLE_TYPE_SYNC_FD_BIT = 16
1444 } VkExternalSemaphoreHandleTypeFlagBits;
1445
1446 typedef enum VkExternalSemaphoreFeatureFlagBits {
1447     VK_EXTERNAL_SEMAPHORE_FEATURE_EXPORTABLE_BIT = 1,
1448     VK_EXTERNAL_SEMAPHORE_FEATURE_IMPORTABLE_BIT = 2
1449 } VkExternalSemaphoreFeatureFlagBits;
1450
1451 typedef enum VkSemaphoreImportFlagBits {
1452     VK_SEMAPHORE_IMPORT_TEMPORARY_BIT = 1
1453 } VkSemaphoreImportFlagBits;
1454
1455 typedef enum VkExternalFenceHandleTypeFlagBits {
1456     VK_EXTERNAL_FENCE_HANDLE_TYPE_OPAQUE_FD_BIT = 1,
1457     VK_EXTERNAL_FENCE_HANDLE_TYPE_OPAQUE_WIN32_BIT = 2,
1458     VK_EXTERNAL_FENCE_HANDLE_TYPE_OPAQUE_WIN32_KMT_BIT = 4,
1459     VK_EXTERNAL_FENCE_HANDLE_TYPE_SYNC_FD_BIT = 8
1460 } VkExternalFenceHandleTypeFlagBits;
1461
1462 typedef enum VkExternalFenceFeatureFlagBits {
1463     VK_EXTERNAL_FENCE_FEATURE_EXPORTABLE_BIT = 1,
1464     VK_EXTERNAL_FENCE_FEATURE_IMPORTABLE_BIT = 2
1465 } VkExternalFenceFeatureFlagBits;
1466
1467 typedef enum VkFenceImportFlagBits {
1468     VK_FENCE_IMPORT_TEMPORARY_BIT = 1
1469 } VkFenceImportFlagBits;
1470
1471 typedef enum VkPeerMemoryFeatureFlagBits {
1472     VK_PEER_MEMORY_FEATURE_COPY_SRC_BIT = 1,
1473     VK_PEER_MEMORY_FEATURE_COPY_DST_BIT = 2,
1474     VK_PEER_MEMORY_FEATURE_GENERIC_SRC_BIT = 4,
1475     VK_PEER_MEMORY_FEATURE_GENERIC_DST_BIT = 8
1476 } VkPeerMemoryFeatureFlagBits;
1477
1478 typedef enum VkMemoryAllocateFlagBits {
1479     VK_MEMORY_ALLOCATE_DEVICE_MASK_BIT = 1
1480 } VkMemoryAllocateFlagBits;
1481
1482 typedef enum VkDeviceGroupPresentModeFlagBitsKHR {
1483     VK_DEVICE_GROUP_PRESENT_MODE_LOCAL_BIT_KHR = 1,
1484     VK_DEVICE_GROUP_PRESENT_MODE_REMOTE_BIT_KHR = 2,
1485     VK_DEVICE_GROUP_PRESENT_MODE_SUM_BIT_KHR = 4,
1486     VK_DEVICE_GROUP_PRESENT_MODE_LOCAL_MULTI_DEVICE_BIT_KHR = 8
1487 } VkDeviceGroupPresentModeFlagBitsKHR;
1488
1489 typedef enum VkSwapchainCreateFlagBitsKHR {
1490     VK_SWAPCHAIN_CREATE_SPLIT_INSTANCE_BIND_REGIONS_BIT_KHR = 1,
1491     VK_SWAPCHAIN_CREATE_PROTECTED_BIT_KHR = 2
1492 } VkSwapchainCreateFlagBitsKHR;
```

```
1493
1494 typedef enum VkSubgroupFeatureFlagBits {
1495     VK_SUBGROUP_FEATURE_BASIC_BIT = 1,
1496     VK_SUBGROUP_FEATURE_VOTE_BIT = 2,
1497     VK_SUBGROUP_FEATURE_ARITHMETIC_BIT = 4,
1498     VK_SUBGROUP_FEATURE_BALLOT_BIT = 8,
1499     VK_SUBGROUP_FEATURE_SHUFFLE_BIT = 16,
1500     VK_SUBGROUP_FEATURE_SHUFFLE_RELATIVE_BIT = 32,
1501     VK_SUBGROUP_FEATURE_CLUSTERED_BIT = 64,
1502     VK_SUBGROUP_FEATURE_QUAD_BIT = 128
1503 } VkSubgroupFeatureFlagBits;
1504
1505 typedef enum VkTessellationDomainOrigin {
1506     VK_TESSELLATION_DOMAIN_ORIGIN_UPPER_LEFT = 0,
1507     VK_TESSELLATION_DOMAIN_ORIGIN_LOWER_LEFT = 1
1508 } VkTessellationDomainOrigin;
1509
1510 typedef enum VkSamplerYcbcrModelConversion {
1511     VK_SAMPLER_YCBCR_MODEL_CONVERSION_RGB_IDENTITY = 0,
1512     VK_SAMPLER_YCBCR_MODEL_CONVERSION_YCBCR_IDENTITY = 1,
1513     VK_SAMPLER_YCBCR_MODEL_CONVERSION_YCBCR_709 = 2,
1514     VK_SAMPLER_YCBCR_MODEL_CONVERSION_YCBCR_601 = 3,
1515     VK_SAMPLER_YCBCR_MODEL_CONVERSION_YCBCR_2020 = 4
1516 } VkSamplerYcbcrModelConversion;
1517
1518 typedef enum VkSamplerYcbcrRange {
1519     VK_SAMPLER_YCBCR_RANGE_ITU_FULL = 0,
1520     VK_SAMPLER_YCBCR_RANGE_ITU_NARROW = 1
1521 } VkSamplerYcbcrRange;
1522
1523 typedef enum VkChromaLocation {
1524     VK_CHROMA_LOCATION_COSITED_EVEN = 0,
1525     VK_CHROMA_LOCATION_MIDPOINT = 1
1526 } VkChromaLocation;
1527
1528 typedef enum VkVendorId {
1529     VK_VENDOR_ID_VIV = 0x10001,
1530     VK_VENDOR_ID_VSI = 0x10002,
1531     VK_VENDOR_ID_KAZAN = 0x10003,
1532     VK_VENDOR_ID_CODEPLAY = 0x10004,
1533     VK_VENDOR_ID_MESA = 0x10005
1534 } VkVendorId;
1535
1536 typedef void (VKAPI_PTR *PFN_vkInternalAllocationNotification)(
1537     void*                                       pUserData,
1538     size_t                                      size,
1539     VkInternalAllocationType                    allocationType,
1540     VkSystemAllocationScope                     allocationScope);
1541
1542 typedef void (VKAPI_PTR *PFN_vkInternalFreeNotification)(
1543     void*                                       pUserData,
1544     size_t                                      size,
1545     VkInternalAllocationType                    allocationType,
1546     VkSystemAllocationScope                     allocationScope);
1547
1548 typedef void* (VKAPI_PTR *PFN_vkReallocationFunction)(
1549     void*                                       pUserData,
1550     void*                                       pOriginal,
1551     size_t                                      size,
1552     size_t                                      alignment,
1553     VkSystemAllocationScope                     allocationScope);
1554
1555 typedef void* (VKAPI_PTR *PFN_vkAllocationFunction)(
1556     void*                                       pUserData,
1557     size_t                                      size,
1558     size_t                                      alignment,
1559     VkSystemAllocationScope                     allocationScope);
1560
1561 typedef void (VKAPI_PTR *PFN_vkFreeFunction)(
1562     void*                                       pUserData,
1563     void*                                       pMemory);
1564
1565 typedef void (VKAPI_PTR *PFN_vkVoidFunction)(void);
1566
1567 typedef struct VkBaseOutStructure {
1568     VkStructureType    sType;
1569     struct  VkBaseOutStructure *  pNext;
1570 } VkBaseOutStructure;
1571
1572 typedef struct VkBaseInStructure {
1573     VkStructureType    sType;
1574     const struct  VkBaseInStructure *  pNext;
1575 } VkBaseInStructure;
1576
1577 typedef struct VkOffset2D {
1578     int32_t          x;
1579     int32_t          y;
```

```
1580 } VkOffset2D;
1581
1582 typedef struct VkOffset3D {
1583     int32_t          x;
1584     int32_t          y;
1585     int32_t          z;
1586 } VkOffset3D;
1587
1588 typedef struct VkExtent2D {
1589     uint32_t          width;
1590     uint32_t          height;
1591 } VkExtent2D;
1592
1593 typedef struct VkExtent3D {
1594     uint32_t          width;
1595     uint32_t          height;
1596     uint32_t          depth;
1597 } VkExtent3D;
1598
1599 typedef struct VkViewport {
1600     float   x;
1601     float   y;
1602     float   width;
1603     float   height;
1604     float                          minDepth;
1605     float                          maxDepth;
1606 } VkViewport;
1607
1608 typedef struct VkRect2D {
1609     VkOffset2D       offset;
1610     VkExtent2D       extent;
1611 } VkRect2D;
1612
1613 typedef struct VkClearRect {
1614     VkRect2D         rect;
1615     uint32_t         baseArrayLayer;
1616     uint32_t         layerCount;
1617 } VkClearRect;
1618
1619 typedef struct VkComponentMapping {
1620     VkComponentSwizzle   r;
1621     VkComponentSwizzle   g;
1622     VkComponentSwizzle   b;
1623     VkComponentSwizzle   a;
1624 } VkComponentMapping;
1625
1626 typedef struct VkExtensionProperties {
1627     char              extensionName [ VK_MAX_EXTENSION_NAME_SIZE ];
1628     uint32_t          specVersion;
1629 } VkExtensionProperties;
1630
1631 typedef struct VkLayerProperties {
1632     char              layerName [ VK_MAX_EXTENSION_NAME_SIZE ];
1633     uint32_t          specVersion;
1634     uint32_t          implementationVersion;
1635     char              description [ VK_MAX_DESCRIPTION_SIZE ];
1636 } VkLayerProperties;
1637
1638 typedef struct VkApplicationInfo {
1639     VkStructureType   sType;
1640     const  void *      pNext;
1641     const  char *      pApplicationName;
1642     uint32_t          applicationVersion;
1643     const  char *      pEngineName;
1644     uint32_t          engineVersion;
1645     uint32_t          apiVersion;
1646 } VkApplicationInfo;
1647
1648 typedef struct VkAllocationCallbacks {
1649     void *            pUserData;
1650     PFN_vkAllocationFunction     pfnAllocation;
1651     PFN_vkReallocationFunction   pfnReallocation;
1652     PFN_vkFreeFunction      pfnFree;
1653     PFN_vkInternalAllocationNotification   pfnInternalAllocation;
1654     PFN_vkInternalFreeNotification   pfnInternalFree;
1655 } VkAllocationCallbacks;
1656
1657 typedef struct VkDescriptorImageInfo {
1658     VkSampler         sampler;
1659     VkImageView       imageView;
1660     VkImageLayout     imageLayout;
1661 } VkDescriptorImageInfo;
1662
1663 typedef struct VkCopyDescriptorSet {
1664     VkStructureType   sType;
1665     const  void *             pNext;
1666     VkDescriptorSet          srcSet;
```

```
1667     uint32_t                 srcBinding;
1668     uint32_t                 srcArrayElement;
1669     VkDescriptorSet          dstSet;
1670     uint32_t                 dstBinding;
1671     uint32_t                 dstArrayElement;
1672     uint32_t                 descriptorCount;
1673 } VkCopyDescriptorSet;
1674
1675 typedef struct VkDescriptorPoolSize {
1676     VkDescriptorType         type;
1677     uint32_t                 descriptorCount;
1678 } VkDescriptorPoolSize;
1679
1680 typedef struct VkDescriptorSetAllocateInfo {
1681     VkStructureType    sType;
1682     const  void *             pNext;
1683     VkDescriptorPool         descriptorPool;
1684     uint32_t                 descriptorSetCount;
1685     const  VkDescriptorSetLayout *  pSetLayouts;
1686 } VkDescriptorSetAllocateInfo;
1687
1688 typedef struct VkSpecializationMapEntry {
1689     uint32_t                       constantID;
1690     uint32_t                       offset;
1691     size_t    size;
1692 } VkSpecializationMapEntry;
1693
1694 typedef struct VkSpecializationInfo {
1695     uint32_t                 mapEntryCount;
1696     const  VkSpecializationMapEntry *  pMapEntries;
1697     size_t                   dataSize;
1698     const  void *             pData;
1699 } VkSpecializationInfo;
1700
1701 typedef struct VkVertexInputBindingDescription {
1702     uint32_t                 binding;
1703     uint32_t                 stride;
1704     VkVertexInputRate        inputRate;
1705 } VkVertexInputBindingDescription;
1706
1707 typedef struct VkVertexInputAttributeDescription {
1708     uint32_t                 location;
1709     uint32_t                 binding;
1710     VkFormat                 format;
1711     uint32_t                 offset;
1712 } VkVertexInputAttributeDescription;
1713
1714 typedef struct VkStencilOpState {
1715     VkStencilOp              failOp;
1716     VkStencilOp              passOp;
1717     VkStencilOp              depthFailOp;
1718     VkCompareOp              compareOp;
1719     uint32_t                 compareMask;
1720     uint32_t                 writeMask;
1721     uint32_t                 reference;
1722 } VkStencilOpState;
1723
1724 typedef struct VkCommandBufferAllocateInfo {
1725     VkStructureType    sType;
1726     const  void *             pNext;
1727     VkCommandPool            commandPool;
1728     VkCommandBufferLevel     level;
1729     uint32_t                 commandBufferCount;
1730 } VkCommandBufferAllocateInfo;
1731
1732 typedef union VkClearColorValue {
1733     float                    float32 [4];
1734     int32_t                  int32 [4];
1735     uint32_t                 uint32 [4];
1736 } VkClearColorValue;
1737
1738 typedef struct VkClearDepthStencilValue {
1739     float                    depth;
1740     uint32_t                 stencil;
1741 } VkClearDepthStencilValue;
1742
1743 typedef union VkClearValue {
1744     VkClearColorValue        color;
1745     VkClearDepthStencilValue   depthStencil;
1746 } VkClearValue;
1747
1748 typedef struct VkAttachmentReference {
1749     uint32_t                 attachment;
1750     VkImageLayout            layout;
1751 } VkAttachmentReference;
1752
1753 typedef struct VkDrawIndirectCommand {
```

```
1754     uint32_t                           vertexCount;
1755     uint32_t                           instanceCount;
1756     uint32_t                           firstVertex;
1757     uint32_t   firstInstance;
1758 } VkDrawIndirectCommand;
1759
1760 typedef struct VkDrawIndexedIndirectCommand {
1761     uint32_t                           indexCount;
1762     uint32_t                           instanceCount;
1763     uint32_t                           firstIndex;
1764     int32_t                            vertexOffset;
1765     uint32_t   firstInstance;
1766 } VkDrawIndexedIndirectCommand;
1767
1768 typedef struct VkDispatchIndirectCommand {
1769     uint32_t   x;
1770     uint32_t   y;
1771     uint32_t   z;
1772 } VkDispatchIndirectCommand;
1773
1774 typedef struct VkSurfaceFormatKHR {
1775     VkFormat                           format;
1776     VkColorSpaceKHR                    colorSpace;
1777 } VkSurfaceFormatKHR;
1778
1779 typedef struct VkPresentInfoKHR {
1780     VkStructureType   sType;
1781     const  void *   pNext;
1782     uint32_t           waitSemaphoreCount;
1783     const  VkSemaphore *  pWaitSemaphores;
1784     uint32_t                           swapchainCount;
1785     const  VkSwapchainKHR *  pSwapchains;
1786     const  uint32_t *  pImageIndices;
1787     VkResult *  pResults;
1788 } VkPresentInfoKHR;
1789
1790 typedef struct VkPhysicalDeviceExternalImageFormatInfo {
1791     VkStructureType   sType;
1792     const  void *                       pNext;
1793     VkExternalMemoryHandleTypeFlagBits   handleType;
1794 } VkPhysicalDeviceExternalImageFormatInfo;
1795
1796 typedef struct VkPhysicalDeviceExternalSemaphoreInfo {
1797     VkStructureType   sType;
1798     const  void *                       pNext;
1799     VkExternalSemaphoreHandleTypeFlagBits   handleType;
1800 } VkPhysicalDeviceExternalSemaphoreInfo;
1801
1802 typedef struct VkPhysicalDeviceExternalFenceInfo {
1803     VkStructureType   sType;
1804     const  void *                       pNext;
1805     VkExternalFenceHandleTypeFlagBits   handleType;
1806 } VkPhysicalDeviceExternalFenceInfo;
1807
1808 typedef struct VkPhysicalDeviceMultiviewProperties {
1809     VkStructureType   sType;
1810     void *                             pNext;
1811     uint32_t                           maxMultiviewViewCount;
1812     uint32_t                           maxMultiviewInstanceIndex;
1813 } VkPhysicalDeviceMultiviewProperties;
1814
1815 typedef struct VkRenderPassMultiviewCreateInfo {
1816     VkStructureType          sType;
1817     const  void *             pNext;
1818     uint32_t                 subpassCount;
1819     const  uint32_t *      pViewMasks;
1820     uint32_t                   dependencyCount;
1821     const  int32_t *    pViewOffsets;
1822     uint32_t                   correlationMaskCount;
1823     const  uint32_t *  pCorrelationMasks;
1824 } VkRenderPassMultiviewCreateInfo;
1825
1826 typedef struct VkBindBufferMemoryDeviceGroupInfo {
1827     VkStructureType   sType;
1828     const  void *                       pNext;
1829     uint32_t           deviceIndexCount;
1830     const  uint32_t *   pDeviceIndices;
1831 } VkBindBufferMemoryDeviceGroupInfo;
1832
1833 typedef struct VkBindImageMemoryDeviceGroupInfo {
1834     VkStructureType   sType;
1835     const  void *                       pNext;
1836     uint32_t           deviceIndexCount;
1837     const  uint32_t *   pDeviceIndices;
1838     uint32_t           splitInstanceBindRegionCount;
1839     const  VkRect2D *   pSplitInstanceBindRegions;
1840 } VkBindImageMemoryDeviceGroupInfo;
```

```
1841
1842 typedef struct VkDeviceGroupRenderPassBeginInfo {
1843     VkStructureType    sType;
1844     const  void *                       pNext;
1845     uint32_t                           deviceMask;
1846     uint32_t           deviceRenderAreaCount;
1847     const  VkRect2D *    pDeviceRenderAreas;
1848 } VkDeviceGroupRenderPassBeginInfo;
1849
1850 typedef struct VkDeviceGroupCommandBufferBeginInfo {
1851     VkStructureType    sType;
1852     const  void *                       pNext;
1853     uint32_t                           deviceMask;
1854 } VkDeviceGroupCommandBufferBeginInfo;
1855
1856 typedef struct VkDeviceGroupSubmitInfo {
1857     VkStructureType    sType;
1858     const  void *                       pNext;
1859     uint32_t           waitSemaphoreCount;
1860     const  uint32_t *      pWaitSemaphoreDeviceIndices;
1861     uint32_t           commandBufferCount;
1862     const  uint32_t *      pCommandBufferDeviceMasks;
1863     uint32_t           signalSemaphoreCount;
1864     const  uint32_t *    pSignalSemaphoreDeviceIndices;
1865 } VkDeviceGroupSubmitInfo;
1866
1867 typedef struct VkDeviceGroupBindSparseInfo {
1868     VkStructureType    sType;
1869     const  void *                       pNext;
1870     uint32_t                           resourceDeviceIndex;
1871     uint32_t                           memoryDeviceIndex;
1872 } VkDeviceGroupBindSparseInfo;
1873
1874 typedef struct VkImageSwapchainCreateInfoKHR {
1875     VkStructureType    sType;
1876     const  void *                       pNext;
1877     VkSwapchainKHR     swapchain;
1878 } VkImageSwapchainCreateInfoKHR;
1879
1880 typedef struct VkBindImageMemorySwapchainInfoKHR {
1881     VkStructureType    sType;
1882     const  void *                       pNext;
1883     VkSwapchainKHR    swapchain;
1884     uint32_t                           imageIndex;
1885 } VkBindImageMemorySwapchainInfoKHR;
1886
1887 typedef struct VkAcquireNextImageInfoKHR {
1888     VkStructureType    sType;
1889     const  void *                       pNext;
1890     VkSwapchainKHR    swapchain;
1891     uint64_t                           timeout;
1892     VkSemaphore    semaphore;
1893     VkFence    fence;
1894     uint32_t                           deviceMask;
1895 } VkAcquireNextImageInfoKHR;
1896
1897 typedef struct VkDeviceGroupPresentInfoKHR {
1898     VkStructureType    sType;
1899     const  void *                       pNext;
1900     uint32_t           swapchainCount;
1901     const  uint32_t *  pDeviceMasks;
1902     VkDeviceGroupPresentModeFlagBitsKHR    mode;
1903 } VkDeviceGroupPresentInfoKHR;
1904
1905 typedef struct VkDeviceGroupDeviceCreateInfo {
1906     VkStructureType    sType;
1907     const  void *                       pNext;
1908     uint32_t                           physicalDeviceCount;
1909     const  VkPhysicalDevice *    pPhysicalDevices;
1910 } VkDeviceGroupDeviceCreateInfo;
1911
1912 typedef struct VkDescriptorUpdateTemplateEntry {
1913     uint32_t                           dstBinding;
1914     uint32_t                           dstArrayElement;
1915     uint32_t                           descriptorCount;
1916     VkDescriptorType                   descriptorType;
1917     size_t                             offset;
1918     size_t                             stride;
1919 } VkDescriptorUpdateTemplateEntry;
1920
1921 typedef struct VkBufferMemoryRequirementsInfo2 {
1922     VkStructureType    sType;
1923     const  void *                                                       pNext;
1924     VkBuffer                                                           buffer;
1925 } VkBufferMemoryRequirementsInfo2;
1926
1927 typedef struct VkImageMemoryRequirementsInfo2 {
```

```
1928     VkStructureType    sType;
1929     const  void *                                                    pNext;
1930     VkImage                                                         image;
1931 } VkImageMemoryRequirementsInfo2;
1932
1933 typedef struct VkImageSparseMemoryRequirementsInfo2 {
1934     VkStructureType    sType;
1935     const  void *                                                    pNext;
1936     VkImage                                                         image;
1937 } VkImageSparseMemoryRequirementsInfo2;
1938
1939 typedef struct VkPhysicalDevicePointClippingProperties {
1940     VkStructureType    sType;
1941     void *                              pNext;
1942     VkPointClippingBehavior         pointClippingBehavior;
1943 } VkPhysicalDevicePointClippingProperties;
1944
1945 typedef struct VkMemoryDedicatedAllocateInfo {
1946     VkStructureType    sType;
1947     const  void *                       pNext;
1948     VkImage             image;
1949     VkBuffer            buffer;
1950 } VkMemoryDedicatedAllocateInfo;
1951
1952 typedef struct VkPipelineTessellationDomainOriginStateCreateInfo {
1953     VkStructureType    sType;
1954     const  void *                       pNext;
1955     VkTessellationDomainOrigin      domainOrigin;
1956 } VkPipelineTessellationDomainOriginStateCreateInfo;
1957
1958 typedef struct VkSamplerYcbcrConversionInfo {
1959     VkStructureType    sType;
1960     const  void *                       pNext;
1961     VkSamplerYcbcrConversion        conversion;
1962 } VkSamplerYcbcrConversionInfo;
1963
1964 typedef struct VkBindImagePlaneMemoryInfo {
1965     VkStructureType    sType;
1966     const  void *                       pNext;
1967     VkImageAspectFlagBits               planeAspect;
1968 } VkBindImagePlaneMemoryInfo;
1969
1970 typedef struct VkImagePlaneMemoryRequirementsInfo {
1971     VkStructureType    sType;
1972     const  void *                       pNext;
1973     VkImageAspectFlagBits               planeAspect;
1974 } VkImagePlaneMemoryRequirementsInfo;
1975
1976 typedef struct VkSamplerYcbcrConversionImageFormatProperties {
1977     VkStructureType    sType;
1978     void *       pNext;
1979     uint32_t                            combinedImageSamplerDescriptorCount;
1980 } VkSamplerYcbcrConversionImageFormatProperties;
1981
1982 typedef uint32_t VkSampleMask;
1983
1984 typedef uint32_t VkBool32;
1985
1986 typedef uint32_t VkFlags;
1987
1988 typedef uint64_t VkDeviceSize;
1989
1990 typedef uint64_t VkDeviceAddress;
1991
1992 typedef VkFlags VkFramebufferCreateFlags;
1993
1994 typedef VkFlags VkQueryPoolCreateFlags;
1995
1996 typedef VkFlags VkRenderPassCreateFlags;
1997
1998 typedef VkFlags VkSamplerCreateFlags;
1999
2000 typedef VkFlags VkPipelineLayoutCreateFlags;
2001
2002 typedef VkFlags VkPipelineCacheCreateFlags;
2003
2004 typedef VkFlags VkPipelineDepthStencilStateCreateFlags;
2005
2006 typedef VkFlags VkPipelineDynamicStateCreateFlags;
2007
2008 typedef VkFlags VkPipelineColorBlendStateCreateFlags;
2009
2010 typedef VkFlags VkPipelineMultisampleStateCreateFlags;
2011
2012 typedef VkFlags VkPipelineRasterizationStateCreateFlags;
2013
2014 typedef VkFlags VkPipelineViewportStateCreateFlags;
```

```
2015
2016 typedef VkFlags VkPipelineTessellationStateCreateFlags;
2017
2018 typedef VkFlags VkPipelineInputAssemblyStateCreateFlags;
2019
2020 typedef VkFlags VkPipelineVertexInputStateCreateFlags;
2021
2022 typedef VkFlags VkPipelineShaderStageCreateFlags;
2023
2024 typedef VkFlags VkDescriptorSetLayoutCreateFlags;
2025
2026 typedef VkFlags VkBufferViewCreateFlags;
2027
2028 typedef VkFlags VkInstanceCreateFlags;
2029
2030 typedef VkFlags VkDeviceCreateFlags;
2031
2032 typedef VkFlags VkDeviceQueueCreateFlags;
2033
2034 typedef VkFlags VkQueueFlags;
2035
2036 typedef VkFlags VkMemoryPropertyFlags;
2037
2038 typedef VkFlags VkMemoryHeapFlags;
2039
2040 typedef VkFlags VkAccessFlags;
2041
2042 typedef VkFlags VkBufferUsageFlags;
2043
2044 typedef VkFlags VkBufferCreateFlags;
2045
2046 typedef VkFlags VkShaderStageFlags;
2047
2048 typedef VkFlags VkImageUsageFlags;
2049
2050 typedef VkFlags VkImageCreateFlags;
2051
2052 typedef VkFlags VkImageViewCreateFlags;
2053
2054 typedef VkFlags VkPipelineCreateFlags;
2055
2056 typedef VkFlags VkColorComponentFlags;
2057
2058 typedef VkFlags VkFenceCreateFlags;
2059
2060 typedef VkFlags VkSemaphoreCreateFlags;
2061
2062 typedef VkFlags VkFormatFeatureFlags;
2063
2064 typedef VkFlags VkQueryControlFlags;
2065
2066 typedef VkFlags VkQueryResultFlags;
2067
2068 typedef VkFlags VkShaderModuleCreateFlags;
2069
2070 typedef VkFlags VkEventCreateFlags;
2071
2072 typedef VkFlags VkCommandPoolCreateFlags;
2073
2074 typedef VkFlags VkCommandPoolResetFlags;
2075
2076 typedef VkFlags VkCommandBufferResetFlags;
2077
2078 typedef VkFlags VkCommandBufferUsageFlags;
2079
2080 typedef VkFlags VkQueryPipelineStatisticFlags;
2081
2082 typedef VkFlags VkMemoryMapFlags;
2083
2084 typedef VkFlags VkImageAspectFlags;
2085
2086 typedef VkFlags VkSparseMemoryBindFlags;
2087
2088 typedef VkFlags VkSparseImageFormatFlags;
2089
2090 typedef VkFlags VkSubpassDescriptionFlags;
2091
2092 typedef VkFlags VkPipelineStageFlags;
2093
2094 typedef VkFlags VkSampleCountFlags;
2095
2096 typedef VkFlags VkAttachmentDescriptionFlags;
2097
2098 typedef VkFlags VkStencilFaceFlags;
2099
2100 typedef VkFlags VkCullModeFlags;
2101
```

```
2102 typedef VkFlags VkDescriptorPoolCreateFlags;
2103
2104 typedef VkFlags VkDescriptorPoolResetFlags;
2105
2106 typedef VkFlags VkDependencyFlags;
2107
2108 typedef VkFlags VkSubgroupFeatureFlags;
2109
2110 typedef VkFlags VkDescriptorUpdateTemplateCreateFlags;
2111
2112 typedef VkFlags VkCompositeAlphaFlagsKHR;
2113
2114 typedef VkFlags VkSurfaceTransformFlagsKHR;
2115
2116 typedef VkFlags VkSwapchainCreateFlagsKHR;
2117
2118 typedef VkFlags VkPeerMemoryFeatureFlags;
2119
2120 typedef VkFlags VkMemoryAllocateFlags;
2121
2122 typedef VkFlags VkDeviceGroupPresentModeFlagsKHR;
2123
2124 typedef VkFlags VkDebugReportFlagsEXT;
2125
2126 typedef VkFlags VkCommandPoolTrimFlags;
2127
2128 typedef VkFlags VkExternalMemoryHandleTypeFlags;
2129
2130 typedef VkFlags VkExternalMemoryFeatureFlags;
2131
2132 typedef VkFlags VkExternalSemaphoreHandleTypeFlags;
2133
2134 typedef VkFlags VkExternalSemaphoreFeatureFlags;
2135
2136 typedef VkFlags VkSemaphoreImportFlags;
2137
2138 typedef VkFlags VkExternalFenceHandleTypeFlags;
2139
2140 typedef VkFlags VkExternalFenceFeatureFlags;
2141
2142 typedef VkFlags VkFenceImportFlags;
2143
2144 typedef VkBool32 (VKAPI_PTR *PFN_vkDebugReportCallbackEXT)(
2145     VkDebugReportFlagsEXT                       flags,
2146     VkDebugReportObjectTypeEXT                  objectType,
2147     uint64_t                                    object,
2148     size_t                                      location,
2149     int32_t                                     messageCode,
2150     const char*                                 pLayerPrefix,
2151     const char*                                 pMessage,
2152     void*                                       pUserData);
2153
2154 typedef struct VkDeviceQueueCreateInfo {
2155     VkStructureType   sType;
2156     const  void *      pNext;
2157     VkDeviceQueueCreateFlags      flags;
2158     uint32_t          queueFamilyIndex;
2159     uint32_t          queueCount;
2160     const  float *     pQueuePriorities;
2161 } VkDeviceQueueCreateInfo;
2162
2163 typedef struct VkInstanceCreateInfo {
2164     VkStructureType   sType;
2165     const  void *      pNext;
2166     VkInstanceCreateFlags    flags;
2167     const  VkApplicationInfo *  pApplicationInfo;
2168     uint32_t                 enabledLayerCount;
2169     const  char * const*       ppEnabledLayerNames;
2170     uint32_t                 enabledExtensionCount;
2171     const  char * const*       ppEnabledExtensionNames;
2172 } VkInstanceCreateInfo;
2173
2174 typedef struct VkQueueFamilyProperties {
2175     VkQueueFlags             queueFlags;
2176     uint32_t                 queueCount;
2177     uint32_t                 timestampValidBits;
2178     VkExtent3D               minImageTransferGranularity;
2179 } VkQueueFamilyProperties;
2180
2181 typedef struct VkMemoryAllocateInfo {
2182     VkStructureType   sType;
2183     const  void *             pNext;
2184     VkDeviceSize             allocationSize;
2185     uint32_t                 memoryTypeIndex;
2186 } VkMemoryAllocateInfo;
2187
2188 typedef struct VkMemoryRequirements {
```

```
2189     VkDeviceSize             size;
2190     VkDeviceSize             alignment;
2191     uint32_t                 memoryTypeBits;
2192 } VkMemoryRequirements;
2193
2194 typedef struct VkSparseImageFormatProperties {
2195     VkImageAspectFlags       aspectMask;
2196     VkExtent3D               imageGranularity;
2197     VkSparseImageFormatFlags   flags;
2198 } VkSparseImageFormatProperties;
2199
2200 typedef struct VkSparseImageMemoryRequirements {
2201     VkSparseImageFormatProperties   formatProperties;
2202     uint32_t                 imageMipTailFirstLod;
2203     VkDeviceSize             imageMipTailSize;
2204     VkDeviceSize             imageMipTailOffset;
2205     VkDeviceSize             imageMipTailStride;
2206 } VkSparseImageMemoryRequirements;
2207
2208 typedef struct VkMemoryType {
2209     VkMemoryPropertyFlags    propertyFlags;
2210     uint32_t                 heapIndex;
2211 } VkMemoryType;
2212
2213 typedef struct VkMemoryHeap {
2214     VkDeviceSize             size;
2215     VkMemoryHeapFlags        flags;
2216 } VkMemoryHeap;
2217
2218 typedef struct VkMappedMemoryRange {
2219     VkStructureType   sType;
2220     const  void *             pNext;
2221     VkDeviceMemory           memory;
2222     VkDeviceSize             offset;
2223     VkDeviceSize             size;
2224 } VkMappedMemoryRange;
2225
2226 typedef struct VkFormatProperties {
2227     VkFormatFeatureFlags     linearTilingFeatures;
2228     VkFormatFeatureFlags     optimalTilingFeatures;
2229     VkFormatFeatureFlags     bufferFeatures;
2230 } VkFormatProperties;
2231
2232 typedef struct VkImageFormatProperties {
2233     VkExtent3D               maxExtent;
2234     uint32_t                 maxMipLevels;
2235     uint32_t                 maxArrayLayers;
2236     VkSampleCountFlags       sampleCounts;
2237     VkDeviceSize             maxResourceSize;
2238 } VkImageFormatProperties;
2239
2240 typedef struct VkDescriptorBufferInfo {
2241     VkBuffer                 buffer;
2242     VkDeviceSize             offset;
2243     VkDeviceSize             range;
2244 } VkDescriptorBufferInfo;
2245
2246 typedef struct VkWriteDescriptorSet {
2247     VkStructureType   sType;
2248     const  void *             pNext;
2249     VkDescriptorSet          dstSet;
2250     uint32_t                 dstBinding;
2251     uint32_t                 dstArrayElement;
2252     uint32_t                 descriptorCount;
2253     VkDescriptorType         descriptorType;
2254     const  VkDescriptorImageInfo *  pImageInfo;
2255     const  VkDescriptorBufferInfo *  pBufferInfo;
2256     const  VkBufferView *     pTexelBufferView;
2257 } VkWriteDescriptorSet;
2258
2259 typedef struct VkBufferCreateInfo {
2260     VkStructureType   sType;
2261     const  void *             pNext;
2262     VkBufferCreateFlags      flags;
2263     VkDeviceSize             size;
2264     VkBufferUsageFlags       usage;
2265     VkSharingMode            sharingMode;
2266     uint32_t                 queueFamilyIndexCount;
2267     const  uint32_t *         pQueueFamilyIndices;
2268 } VkBufferCreateInfo;
2269
2270 typedef struct VkBufferViewCreateInfo {
2271     VkStructureType   sType;
2272     const  void *             pNext;
2273     VkBufferViewCreateFlags flags;
2274     VkBuffer                 buffer;
2275     VkFormat                 format;
```

```
2276     VkDeviceSize             offset;
2277     VkDeviceSize             range;
2278 } VkBufferViewCreateInfo;
2279
2280 typedef struct VkImageSubresource {
2281     VkImageAspectFlags       aspectMask;
2282     uint32_t                 mipLevel;
2283     uint32_t                 arrayLayer;
2284 } VkImageSubresource;
2285
2286 typedef struct VkImageSubresourceLayers {
2287     VkImageAspectFlags       aspectMask;
2288     uint32_t                 mipLevel;
2289     uint32_t                 baseArrayLayer;
2290     uint32_t                 layerCount;
2291 } VkImageSubresourceLayers;
2292
2293 typedef struct VkImageSubresourceRange {
2294     VkImageAspectFlags       aspectMask;
2295     uint32_t                 baseMipLevel;
2296     uint32_t                 levelCount;
2297     uint32_t                 baseArrayLayer;
2298     uint32_t                 layerCount;
2299 } VkImageSubresourceRange;
2300
2301 typedef struct VkMemoryBarrier {
2302     VkStructureType   sType;
2303     const  void *             pNext;
2304     VkAccessFlags            srcAccessMask;
2305     VkAccessFlags            dstAccessMask;
2306 } VkMemoryBarrier;
2307
2308 typedef struct VkBufferMemoryBarrier {
2309     VkStructureType   sType;
2310     const  void *             pNext;
2311     VkAccessFlags            srcAccessMask;
2312     VkAccessFlags            dstAccessMask;
2313     uint32_t                 srcQueueFamilyIndex;
2314     uint32_t                 dstQueueFamilyIndex;
2315     VkBuffer                 buffer;
2316     VkDeviceSize             offset;
2317     VkDeviceSize             size;
2318 } VkBufferMemoryBarrier;
2319
2320 typedef struct VkImageMemoryBarrier {
2321     VkStructureType   sType;
2322     const  void *             pNext;
2323     VkAccessFlags            srcAccessMask;
2324     VkAccessFlags            dstAccessMask;
2325     VkImageLayout            oldLayout;
2326     VkImageLayout            newLayout;
2327     uint32_t                 srcQueueFamilyIndex;
2328     uint32_t                 dstQueueFamilyIndex;
2329     VkImage                  image;
2330     VkImageSubresourceRange   subresourceRange;
2331 } VkImageMemoryBarrier;
2332
2333 typedef struct VkImageCreateInfo {
2334     VkStructureType   sType;
2335     const  void *             pNext;
2336     VkImageCreateFlags       flags;
2337     VkImageType              imageType;
2338     VkFormat                 format;
2339     VkExtent3D               extent;
2340     uint32_t                 mipLevels;
2341     uint32_t                 arrayLayers;
2342     VkSampleCountFlagBits    samples;
2343     VkImageTiling            tiling;
2344     VkImageUsageFlags        usage;
2345     VkSharingMode            sharingMode;
2346     uint32_t                 queueFamilyIndexCount;
2347     const  uint32_t *         pQueueFamilyIndices;
2348     VkImageLayout            initialLayout;
2349 } VkImageCreateInfo;
2350
2351 typedef struct VkSubresourceLayout {
2352     VkDeviceSize             offset;
2353     VkDeviceSize             size;
2354     VkDeviceSize             rowPitch;
2355     VkDeviceSize             arrayPitch;
2356     VkDeviceSize             depthPitch;
2357 } VkSubresourceLayout;
2358
2359 typedef struct VkImageViewCreateInfo {
2360     VkStructureType   sType;
2361     const  void *             pNext;
2362     VkImageViewCreateFlags   flags;
```

```
2363     VkImage                  image;
2364     VkImageViewType          viewType;
2365     VkFormat                 format;
2366     VkComponentMapping       components;
2367     VkImageSubresourceRange   subresourceRange;
2368 } VkImageViewCreateInfo;
2369
2370 typedef struct VkBufferCopy {
2371     VkDeviceSize                        srcOffset;
2372     VkDeviceSize                        dstOffset;
2373     VkDeviceSize   size;
2374 } VkBufferCopy;
2375
2376 typedef struct VkSparseMemoryBind {
2377     VkDeviceSize             resourceOffset;
2378     VkDeviceSize             size;
2379     VkDeviceMemory           memory;
2380     VkDeviceSize             memoryOffset;
2381     VkSparseMemoryBindFlags flags;
2382 } VkSparseMemoryBind;
2383
2384 typedef struct VkSparseImageMemoryBind {
2385     VkImageSubresource       subresource;
2386     VkOffset3D               offset;
2387     VkExtent3D               extent;
2388     VkDeviceMemory           memory;
2389     VkDeviceSize             memoryOffset;
2390     VkSparseMemoryBindFlags flags;
2391 } VkSparseImageMemoryBind;
2392
2393 typedef struct VkSparseBufferMemoryBindInfo {
2394     VkBuffer   buffer;
2395     uint32_t                 bindCount;
2396     const  VkSparseMemoryBind *  pBinds;
2397 } VkSparseBufferMemoryBindInfo;
2398
2399 typedef struct VkSparseImageOpaqueMemoryBindInfo {
2400     VkImage   image;
2401     uint32_t                 bindCount;
2402     const  VkSparseMemoryBind *  pBinds;
2403 } VkSparseImageOpaqueMemoryBindInfo;
2404
2405 typedef struct VkSparseImageMemoryBindInfo {
2406     VkImage   image;
2407     uint32_t                 bindCount;
2408     const  VkSparseImageMemoryBind *  pBinds;
2409 } VkSparseImageMemoryBindInfo;
2410
2411 typedef struct VkBindSparseInfo {
2412     VkStructureType   sType;
2413     const  void *             pNext;
2414     uint32_t                 waitSemaphoreCount;
2415     const  VkSemaphore *      pWaitSemaphores;
2416     uint32_t                 bufferBindCount;
2417     const  VkSparseBufferMemoryBindInfo *  pBufferBinds;
2418     uint32_t                 imageOpaqueBindCount;
2419     const  VkSparseImageOpaqueMemoryBindInfo *  pImageOpaqueBinds;
2420     uint32_t                 imageBindCount;
2421     const  VkSparseImageMemoryBindInfo *  pImageBinds;
2422     uint32_t                 signalSemaphoreCount;
2423     const  VkSemaphore *      pSignalSemaphores;
2424 } VkBindSparseInfo;
2425
2426 typedef struct VkImageCopy {
2427     VkImageSubresourceLayers   srcSubresource;
2428     VkOffset3D               srcOffset;
2429     VkImageSubresourceLayers   dstSubresource;
2430     VkOffset3D               dstOffset;
2431     VkExtent3D               extent;
2432 } VkImageCopy;
2433
2434 typedef struct VkImageBlit {
2435     VkImageSubresourceLayers   srcSubresource;
2436     VkOffset3D               srcOffsets [2];
2437     VkImageSubresourceLayers   dstSubresource;
2438     VkOffset3D               dstOffsets [2];
2439 } VkImageBlit;
2440
2441 typedef struct VkBufferImageCopy {
2442     VkDeviceSize             bufferOffset;
2443     uint32_t                 bufferRowLength;
2444     uint32_t                 bufferImageHeight;
2445     VkImageSubresourceLayers   imageSubresource;
2446     VkOffset3D               imageOffset;
2447     VkExtent3D               imageExtent;
2448 } VkBufferImageCopy;
2449
```

```
2450 typedef struct VkImageResolve {
2451     VkImageSubresourceLayers   srcSubresource;
2452     VkOffset3D                 srcOffset;
2453     VkImageSubresourceLayers   dstSubresource;
2454     VkOffset3D                 dstOffset;
2455     VkExtent3D                 extent;
2456 } VkImageResolve;
2457
2458 typedef struct VkShaderModuleCreateInfo {
2459     VkStructureType   sType;
2460     const  void *             pNext;
2461     VkShaderModuleCreateFlags   flags;
2462     size_t                    codeSize;
2463     const  uint32_t *             pCode;
2464 } VkShaderModuleCreateInfo;
2465
2466 typedef struct VkDescriptorSetLayoutBinding {
2467     uint32_t                  binding;
2468     VkDescriptorType          descriptorType;
2469     uint32_t   descriptorCount;
2470     VkShaderStageFlags        stageFlags;
2471     const  VkSampler *        pImmutableSamplers;
2472 } VkDescriptorSetLayoutBinding;
2473
2474 typedef struct VkDescriptorSetLayoutCreateInfo {
2475     VkStructureType   sType;
2476     const  void *             pNext;
2477     VkDescriptorSetLayoutCreateFlags      flags;
2478     uint32_t                  bindingCount;
2479     const  VkDescriptorSetLayoutBinding *  pBindings;
2480 } VkDescriptorSetLayoutCreateInfo;
2481
2482 typedef struct VkDescriptorPoolCreateInfo {
2483     VkStructureType   sType;
2484     const  void *             pNext;
2485     VkDescriptorPoolCreateFlags    flags;
2486     uint32_t                  maxSets;
2487     uint32_t                  poolSizeCount;
2488     const  VkDescriptorPoolSize *  pPoolSizes;
2489 } VkDescriptorPoolCreateInfo;
2490
2491 typedef struct VkPipelineShaderStageCreateInfo {
2492     VkStructureType   sType;
2493     const  void *             pNext;
2494     VkPipelineShaderStageCreateFlags      flags;
2495     VkShaderStageFlagBits     stage;
2496     VkShaderModule            module;
2497     const  char *             pName;
2498     const  VkSpecializationInfo *  pSpecializationInfo;
2499 } VkPipelineShaderStageCreateInfo;
2500
2501 typedef struct VkComputePipelineCreateInfo {
2502     VkStructureType   sType;
2503     const  void *             pNext;
2504     VkPipelineCreateFlags     flags;
2505     VkPipelineShaderStageCreateInfo   stage;
2506     VkPipelineLayout          layout;
2507     VkPipeline        basePipelineHandle;
2508     int32_t                   basePipelineIndex;
2509 } VkComputePipelineCreateInfo;
2510
2511 typedef struct VkPipelineVertexInputStateCreateInfo {
2512     VkStructureType   sType;
2513     const  void *             pNext;
2514     VkPipelineVertexInputStateCreateFlags      flags;
2515     uint32_t                  vertexBindingDescriptionCount;
2516     const  VkVertexInputBindingDescription *  pVertexBindingDescriptions;
2517     uint32_t                  vertexAttributeDescriptionCount;
2518     const  VkVertexInputAttributeDescription *  pVertexAttributeDescriptions;
2519 } VkPipelineVertexInputStateCreateInfo;
2520
2521 typedef struct VkPipelineInputAssemblyStateCreateInfo {
2522     VkStructureType   sType;
2523     const  void *             pNext;
2524     VkPipelineInputAssemblyStateCreateFlags      flags;
2525     VkPrimitiveTopology       topology;
2526     VkBool32                  primitiveRestartEnable;
2527 } VkPipelineInputAssemblyStateCreateInfo;
2528
2529 typedef struct VkPipelineTessellationStateCreateInfo {
2530     VkStructureType   sType;
2531     const  void *             pNext;
2532     VkPipelineTessellationStateCreateFlags      flags;
2533     uint32_t                  patchControlPoints;
2534 } VkPipelineTessellationStateCreateInfo;
2535
2536 typedef struct VkPipelineViewportStateCreateInfo {
```

```
2537     VkStructureType    sType;
2538     const  void *              pNext;
2539     VkPipelineViewportStateCreateFlags      flags;
2540     uint32_t                   viewportCount;
2541     const  VkViewport *        pViewports;
2542     uint32_t                   scissorCount;
2543     const  VkRect2D *          pScissors;
2544 } VkPipelineViewportStateCreateInfo;
2545
2546 typedef struct VkPipelineRasterizationStateCreateInfo {
2547     VkStructureType    sType;
2548     const  void *  pNext;
2549     VkPipelineRasterizationStateCreateFlags      flags;
2550     VkBool32                   depthClampEnable;
2551     VkBool32                   rasterizerDiscardEnable;
2552     VkPolygonMode              polygonMode;
2553     VkCullModeFlags            cullMode;
2554     VkFrontFace                frontFace;
2555     VkBool32                   depthBiasEnable;
2556     float                      depthBiasConstantFactor;
2557     float                      depthBiasClamp;
2558     float                      depthBiasSlopeFactor;
2559     float                      lineWidth;
2560 } VkPipelineRasterizationStateCreateInfo;
2561
2562 typedef struct VkPipelineMultisampleStateCreateInfo {
2563     VkStructureType    sType;
2564     const  void *              pNext;
2565     VkPipelineMultisampleStateCreateFlags      flags;
2566     VkSampleCountFlagBits      rasterizationSamples;
2567     VkBool32                   sampleShadingEnable;
2568     float                      minSampleShading;
2569     const  VkSampleMask *      pSampleMask;
2570     VkBool32                   alphaToCoverageEnable;
2571     VkBool32                   alphaToOneEnable;
2572 } VkPipelineMultisampleStateCreateInfo;
2573
2574 typedef struct VkPipelineColorBlendAttachmentState {
2575     VkBool32                   blendEnable;
2576     VkBlendFactor              srcColorBlendFactor;
2577     VkBlendFactor              dstColorBlendFactor;
2578     VkBlendOp                  colorBlendOp;
2579     VkBlendFactor              srcAlphaBlendFactor;
2580     VkBlendFactor              dstAlphaBlendFactor;
2581     VkBlendOp                  alphaBlendOp;
2582     VkColorComponentFlags      colorWriteMask;
2583 } VkPipelineColorBlendAttachmentState;
2584
2585 typedef struct VkPipelineColorBlendStateCreateInfo {
2586     VkStructureType    sType;
2587     const  void *              pNext;
2588     VkPipelineColorBlendStateCreateFlags      flags;
2589     VkBool32                   logicOpEnable;
2590     VkLogicOp                  logicOp;
2591     uint32_t                   attachmentCount;
2592     const  VkPipelineColorBlendAttachmentState *  pAttachments;
2593     float                      blendConstants [4];
2594 } VkPipelineColorBlendStateCreateInfo;
2595
2596 typedef struct VkPipelineDynamicStateCreateInfo {
2597     VkStructureType    sType;
2598     const  void *              pNext;
2599     VkPipelineDynamicStateCreateFlags      flags;
2600     uint32_t                   dynamicStateCount;
2601     const  VkDynamicState *    pDynamicStates;
2602 } VkPipelineDynamicStateCreateInfo;
2603
2604 typedef struct VkPipelineDepthStencilStateCreateInfo {
2605     VkStructureType    sType;
2606     const  void *              pNext;
2607     VkPipelineDepthStencilStateCreateFlags      flags;
2608     VkBool32                   depthTestEnable;
2609     VkBool32                   depthWriteEnable;
2610     VkCompareOp                depthCompareOp;
2611     VkBool32                   depthBoundsTestEnable;
2612     VkBool32                   stencilTestEnable;
2613     VkStencilOpState           front;
2614     VkStencilOpState           back;
2615     float                      minDepthBounds;
2616     float                      maxDepthBounds;
2617 } VkPipelineDepthStencilStateCreateInfo;
2618
2619 typedef struct VkGraphicsPipelineCreateInfo {
2620     VkStructureType    sType;
2621     const  void *              pNext;
2622     VkPipelineCreateFlags      flags;
2623     uint32_t                   stageCount;
```

```
2624     const  VkPipelineShaderStageCreateInfo *  pStages;
2625     const  VkPipelineVertexInputStateCreateInfo *  pVertexInputState;
2626     const  VkPipelineInputAssemblyStateCreateInfo *  pInputAssemblyState;
2627     const  VkPipelineTessellationStateCreateInfo *  pTessellationState;
2628     const  VkPipelineViewportStateCreateInfo *  pViewportState;
2629     const  VkPipelineRasterizationStateCreateInfo *  pRasterizationState;
2630     const  VkPipelineMultisampleStateCreateInfo *  pMultisampleState;
2631     const  VkPipelineDepthStencilStateCreateInfo *  pDepthStencilState;
2632     const  VkPipelineColorBlendStateCreateInfo *  pColorBlendState;
2633     const  VkPipelineDynamicStateCreateInfo *  pDynamicState;
2634     VkPipelineLayout         layout;
2635     VkRenderPass             renderPass;
2636     uint32_t                 subpass;
2637     VkPipeline        basePipelineHandle;
2638     int32_t                  basePipelineIndex;
2639 } VkGraphicsPipelineCreateInfo;
2640
2641 typedef struct VkPipelineCacheCreateInfo {
2642     VkStructureType   sType;
2643     const  void *             pNext;
2644     VkPipelineCacheCreateFlags      flags;
2645     size_t                   initialDataSize;
2646     const  void *             pInitialData;
2647 } VkPipelineCacheCreateInfo;
2648
2649 typedef struct VkPushConstantRange {
2650     VkShaderStageFlags       stageFlags;
2651     uint32_t                 offset;
2652     uint32_t                 size;
2653 } VkPushConstantRange;
2654
2655 typedef struct VkPipelineLayoutCreateInfo {
2656     VkStructureType   sType;
2657     const  void *             pNext;
2658     VkPipelineLayoutCreateFlags      flags;
2659     uint32_t                 setLayoutCount;
2660     const  VkDescriptorSetLayout *  pSetLayouts;
2661     uint32_t                 pushConstantRangeCount;
2662     const  VkPushConstantRange *  pPushConstantRanges;
2663 } VkPipelineLayoutCreateInfo;
2664
2665 typedef struct VkSamplerCreateInfo {
2666     VkStructureType   sType;
2667     const  void *             pNext;
2668     VkSamplerCreateFlags     flags;
2669     VkFilter                 magFilter;
2670     VkFilter                 minFilter;
2671     VkSamplerMipmapMode      mipmapMode;
2672     VkSamplerAddressMode     addressModeU;
2673     VkSamplerAddressMode     addressModeV;
2674     VkSamplerAddressMode     addressModeW;
2675     float                    mipLodBias;
2676     VkBool32                 anisotropyEnable;
2677     float                    maxAnisotropy;
2678     VkBool32                 compareEnable;
2679     VkCompareOp              compareOp;
2680     float                    minLod;
2681     float                    maxLod;
2682     VkBorderColor            borderColor;
2683     VkBool32                 unnormalizedCoordinates;
2684 } VkSamplerCreateInfo;
2685
2686 typedef struct VkCommandPoolCreateInfo {
2687     VkStructureType   sType;
2688     const  void *             pNext;
2689     VkCommandPoolCreateFlags     flags;
2690     uint32_t                 queueFamilyIndex;
2691 } VkCommandPoolCreateInfo;
2692
2693 typedef struct VkCommandBufferInheritanceInfo {
2694     VkStructureType   sType;
2695     const  void *             pNext;
2696     VkRenderPass      renderPass;
2697     uint32_t                 subpass;
2698     VkFramebuffer     framebuffer;
2699     VkBool32                 occlusionQueryEnable;
2700     VkQueryControlFlags      queryFlags;
2701     VkQueryPipelineStatisticFlags   pipelineStatistics;
2702 } VkCommandBufferInheritanceInfo;
2703
2704 typedef struct VkCommandBufferBeginInfo {
2705     VkStructureType   sType;
2706     const  void *             pNext;
2707     VkCommandBufferUsageFlags    flags;
2708     const  VkCommandBufferInheritanceInfo *        pInheritanceInfo;
2709 } VkCommandBufferBeginInfo;
2710
```

```
2711 typedef struct VkRenderPassBeginInfo {
2712     VkStructureType   sType;
2713     const  void *             pNext;
2714     VkRenderPass              renderPass;
2715     VkFramebuffer             framebuffer;
2716     VkRect2D                  renderArea;
2717     uint32_t                  clearValueCount;
2718     const  VkClearValue *     pClearValues;
2719 } VkRenderPassBeginInfo;
2720
2721 typedef struct VkClearAttachment {
2722     VkImageAspectFlags        aspectMask;
2723     uint32_t                  colorAttachment;
2724     VkClearValue              clearValue;
2725 } VkClearAttachment;
2726
2727 typedef struct VkAttachmentDescription {
2728     VkAttachmentDescriptionFlags   flags;
2729     VkFormat                  format;
2730     VkSampleCountFlagBits     samples;
2731     VkAttachmentLoadOp        loadOp;
2732     VkAttachmentStoreOp       storeOp;
2733     VkAttachmentLoadOp        stencilLoadOp;
2734     VkAttachmentStoreOp       stencilStoreOp;
2735     VkImageLayout             initialLayout;
2736     VkImageLayout             finalLayout;
2737 } VkAttachmentDescription;
2738
2739 typedef struct VkSubpassDescription {
2740     VkSubpassDescriptionFlags   flags;
2741     VkPipelineBindPoint       pipelineBindPoint;
2742     uint32_t                  inputAttachmentCount;
2743     const  VkAttachmentReference *  pInputAttachments;
2744     uint32_t                  colorAttachmentCount;
2745     const  VkAttachmentReference *  pColorAttachments;
2746     const  VkAttachmentReference *  pResolveAttachments;
2747     const  VkAttachmentReference *  pDepthStencilAttachment;
2748     uint32_t                  preserveAttachmentCount;
2749     const  uint32_t *  pPreserveAttachments;
2750 } VkSubpassDescription;
2751
2752 typedef struct VkSubpassDependency {
2753     uint32_t                  srcSubpass;
2754     uint32_t                  dstSubpass;
2755     VkPipelineStageFlags      srcStageMask;
2756     VkPipelineStageFlags      dstStageMask;
2757     VkAccessFlags             srcAccessMask;
2758     VkAccessFlags             dstAccessMask;
2759     VkDependencyFlags         dependencyFlags;
2760 } VkSubpassDependency;
2761
2762 typedef struct VkRenderPassCreateInfo {
2763     VkStructureType   sType;
2764     const  void *             pNext;
2765     VkRenderPassCreateFlags   flags;
2766     uint32_t     attachmentCount;
2767     const  VkAttachmentDescription *  pAttachments;
2768     uint32_t                  subpassCount;
2769     const  VkSubpassDescription *  pSubpasses;
2770     uint32_t         dependencyCount;
2771     const  VkSubpassDependency *  pDependencies;
2772 } VkRenderPassCreateInfo;
2773
2774 typedef struct VkEventCreateInfo {
2775     VkStructureType   sType;
2776     const  void *             pNext;
2777     VkEventCreateFlags        flags;
2778 } VkEventCreateInfo;
2779
2780 typedef struct VkFenceCreateInfo {
2781     VkStructureType   sType;
2782     const  void *             pNext;
2783     VkFenceCreateFlags        flags;
2784 } VkFenceCreateInfo;
2785
2786 typedef struct VkPhysicalDeviceFeatures {
2787     VkBool32                  robustBufferAccess;
2788     VkBool32                  fullDrawIndexUint32;
2789     VkBool32                  imageCubeArray;
2790     VkBool32                  independentBlend;
2791     VkBool32                  geometryShader;
2792     VkBool32                  tessellationShader;
2793     VkBool32                  sampleRateShading;
2794     VkBool32                  dualSrcBlend;
2795     VkBool32                  logicOp;
2796     VkBool32                  multiDrawIndirect;
2797     VkBool32                  drawIndirectFirstInstance;
```

```
2798     VkBool32                 depthClamp;
2799     VkBool32                 depthBiasClamp;
2800     VkBool32                 fillModeNonSolid;
2801     VkBool32                 depthBounds;
2802     VkBool32                 wideLines;
2803     VkBool32                 largePoints;
2804     VkBool32                 alphaToOne;
2805     VkBool32                 multiViewport;
2806     VkBool32                 samplerAnisotropy;
2807     VkBool32                 textureCompressionETC2;
2808     VkBool32                 textureCompressionASTC_LDR;
2809     VkBool32                 textureCompressionBC;
2810     VkBool32                 occlusionQueryPrecise;
2811     VkBool32                 pipelineStatisticsQuery;
2812     VkBool32                 vertexPipelineStoresAndAtomics;
2813     VkBool32                 fragmentStoresAndAtomics;
2814     VkBool32                 shaderTessellationAndGeometryPointSize;
2815     VkBool32                 shaderImageGatherExtended;
2816     VkBool32                 shaderStorageImageExtendedFormats;
2817     VkBool32                 shaderStorageImageMultisample;
2818     VkBool32                 shaderStorageImageReadWithoutFormat;
2819     VkBool32                 shaderStorageImageWriteWithoutFormat;
2820     VkBool32                 shaderUniformBufferArrayDynamicIndexing;
2821     VkBool32                 shaderSampledImageArrayDynamicIndexing;
2822     VkBool32                 shaderStorageBufferArrayDynamicIndexing;
2823     VkBool32                 shaderStorageImageArrayDynamicIndexing;
2824     VkBool32                 shaderClipDistance;
2825     VkBool32                 shaderCullDistance;
2826     VkBool32                 shaderFloat64;
2827     VkBool32                 shaderInt64;
2828     VkBool32                 shaderInt16;
2829     VkBool32                 shaderResourceResidency;
2830     VkBool32                 shaderResourceMinLod;
2831     VkBool32                 sparseBinding;
2832     VkBool32                 sparseResidencyBuffer;
2833     VkBool32                 sparseResidencyImage2D;
2834     VkBool32                 sparseResidencyImage3D;
2835     VkBool32                 sparseResidency2Samples;
2836     VkBool32                 sparseResidency4Samples;
2837     VkBool32                 sparseResidency8Samples;
2838     VkBool32                 sparseResidency16Samples;
2839     VkBool32                 sparseResidencyAliased;
2840     VkBool32                 variableMultisampleRate;
2841     VkBool32                 inheritedQueries;
2842 } VkPhysicalDeviceFeatures;
2843
2844 typedef struct VkPhysicalDeviceSparseProperties {
2845     VkBool32                 residencyStandard2DBlockShape;
2846     VkBool32                 residencyStandard2DMultisampleBlockShape;
2847     VkBool32                 residencyStandard3DBlockShape;
2848     VkBool32                 residencyAlignedMipSize;
2849     VkBool32                 residencyNonResidentStrict;
2850 } VkPhysicalDeviceSparseProperties;
2851
2852 typedef struct VkPhysicalDeviceLimits {
2853     uint32_t                 maxImageDimension1D;
2854     uint32_t                 maxImageDimension2D;
2855     uint32_t                 maxImageDimension3D;
2856     uint32_t                 maxImageDimensionCube;
2857     uint32_t                 maxImageArrayLayers;
2858     uint32_t                 maxTexelBufferElements;
2859     uint32_t                 maxUniformBufferRange;
2860     uint32_t                 maxStorageBufferRange;
2861     uint32_t                 maxPushConstantsSize;
2862     uint32_t                 maxMemoryAllocationCount;
2863     uint32_t                 maxSamplerAllocationCount;
2864     VkDeviceSize             bufferImageGranularity;
2865     VkDeviceSize             sparseAddressSpaceSize;
2866     uint32_t                 maxBoundDescriptorSets;
2867     uint32_t                 maxPerStageDescriptorSamplers;
2868     uint32_t                 maxPerStageDescriptorUniformBuffers;
2869     uint32_t                 maxPerStageDescriptorStorageBuffers;
2870     uint32_t                 maxPerStageDescriptorSampledImages;
2871     uint32_t                 maxPerStageDescriptorStorageImages;
2872     uint32_t                 maxPerStageDescriptorInputAttachments;
2873     uint32_t                 maxPerStageResources;
2874     uint32_t                 maxDescriptorSetSamplers;
2875     uint32_t                 maxDescriptorSetUniformBuffers;
2876     uint32_t                 maxDescriptorSetUniformBuffersDynamic;
2877     uint32_t                 maxDescriptorSetStorageBuffers;
2878     uint32_t                 maxDescriptorSetStorageBuffersDynamic;
2879     uint32_t                 maxDescriptorSetSampledImages;
2880     uint32_t                 maxDescriptorSetStorageImages;
2881     uint32_t                 maxDescriptorSetInputAttachments;
2882     uint32_t                 maxVertexInputAttributes;
2883     uint32_t                 maxVertexInputBindings;
2884     uint32_t                 maxVertexInputAttributeOffset;
```

```
2885     uint32_t                    maxVertexInputBindingStride;
2886     uint32_t                    maxVertexOutputComponents;
2887     uint32_t                    maxTessellationGenerationLevel;
2888     uint32_t                    maxTessellationPatchSize;
2889     uint32_t                    maxTessellationControlPerVertexInputComponents;
2890     uint32_t                    maxTessellationControlPerVertexOutputComponents;
2891     uint32_t                    maxTessellationControlPerPatchOutputComponents;
2892     uint32_t                    maxTessellationControlTotalOutputComponents;
2893     uint32_t                    maxTessellationEvaluationInputComponents;
2894     uint32_t                    maxTessellationEvaluationOutputComponents;
2895     uint32_t                    maxGeometryShaderInvocations;
2896     uint32_t                    maxGeometryInputComponents;
2897     uint32_t                    maxGeometryOutputComponents;
2898     uint32_t                    maxGeometryOutputVertices;
2899     uint32_t                    maxGeometryTotalOutputComponents;
2900     uint32_t                    maxFragmentInputComponents;
2901     uint32_t                    maxFragmentOutputAttachments;
2902     uint32_t                    maxFragmentDualSrcAttachments;
2903     uint32_t                    maxFragmentCombinedOutputResources;
2904     uint32_t                    maxComputeSharedMemorySize;
2905     uint32_t                    maxComputeWorkGroupCount [3];
2906     uint32_t                    maxComputeWorkGroupInvocations;
2907     uint32_t                    maxComputeWorkGroupSize [3];
2908     uint32_t                    subPixelPrecisionBits;
2909     uint32_t                    subTexelPrecisionBits;
2910     uint32_t                    mipmapPrecisionBits;
2911     uint32_t                    maxDrawIndexedIndexValue;
2912     uint32_t                    maxDrawIndirectCount;
2913     float                       maxSamplerLodBias;
2914     float                       maxSamplerAnisotropy;
2915     uint32_t                    maxViewports;
2916     uint32_t                    maxViewportDimensions [2];
2917     float                       viewportBoundsRange [2];
2918     uint32_t                    viewportSubPixelBits;
2919     size_t                      minMemoryMapAlignment;
2920     VkDeviceSize                minTexelBufferOffsetAlignment;
2921     VkDeviceSize                minUniformBufferOffsetAlignment;
2922     VkDeviceSize                minStorageBufferOffsetAlignment;
2923     int32_t                     minTexelOffset;
2924     uint32_t                    maxTexelOffset;
2925     int32_t                     minTexelGatherOffset;
2926     uint32_t                    maxTexelGatherOffset;
2927     float                       minInterpolationOffset;
2928     float                       maxInterpolationOffset;
2929     uint32_t                    subPixelInterpolationOffsetBits;
2930     uint32_t                    maxFramebufferWidth;
2931     uint32_t                    maxFramebufferHeight;
2932     uint32_t                    maxFramebufferLayers;
2933     VkSampleCountFlags          framebufferColorSampleCounts;
2934     VkSampleCountFlags          framebufferDepthSampleCounts;
2935     VkSampleCountFlags          framebufferStencilSampleCounts;
2936     VkSampleCountFlags          framebufferNoAttachmentsSampleCounts;
2937     uint32_t                    maxColorAttachments;
2938     VkSampleCountFlags          sampledImageColorSampleCounts;
2939     VkSampleCountFlags          sampledImageIntegerSampleCounts;
2940     VkSampleCountFlags          sampledImageDepthSampleCounts;
2941     VkSampleCountFlags          sampledImageStencilSampleCounts;
2942     VkSampleCountFlags          storageImageSampleCounts;
2943     uint32_t                    maxSampleMaskWords;
2944     VkBool32                    timestampComputeAndGraphics;
2945     float                       timestampPeriod;
2946     uint32_t                    maxClipDistances;
2947     uint32_t                    maxCullDistances;
2948     uint32_t                    maxCombinedClipAndCullDistances;
2949     uint32_t                    discreteQueuePriorities;
2950     float                       pointSizeRange [2];
2951     float                       lineWidthRange [2];
2952     float                       pointSizeGranularity;
2953     float                       lineWidthGranularity;
2954     VkBool32                    strictLines;
2955     VkBool32                    standardSampleLocations;
2956     VkDeviceSize                optimalBufferCopyOffsetAlignment;
2957     VkDeviceSize                optimalBufferCopyRowPitchAlignment;
2958     VkDeviceSize                nonCoherentAtomSize;
2959 } VkPhysicalDeviceLimits;
2960
2961 typedef struct VkSemaphoreCreateInfo {
2962     VkStructureType   sType;
2963     const  void *             pNext;
2964     VkSemaphoreCreateFlags   flags;
2965 } VkSemaphoreCreateInfo;
2966
2967 typedef struct VkQueryPoolCreateInfo {
2968     VkStructureType   sType;
2969     const  void *             pNext;
2970     VkQueryPoolCreateFlags   flags;
2971     VkQueryType              queryType;
```

```
2972     uint32_t                queryCount;
2973     VkQueryPipelineStatisticFlags   pipelineStatistics;
2974 } VkQueryPoolCreateInfo;
2975
2976 typedef struct VkFramebufferCreateInfo {
2977     VkStructureType   sType;
2978     const  void *             pNext;
2979     VkFramebufferCreateFlags      flags;
2980     VkRenderPass              renderPass;
2981     uint32_t                attachmentCount;
2982     const  VkImageView *      pAttachments;
2983     uint32_t                width;
2984     uint32_t                height;
2985     uint32_t                layers;
2986 } VkFramebufferCreateInfo;
2987
2988 typedef struct VkSubmitInfo {
2989     VkStructureType   sType;
2990     const  void *  pNext;
2991     uint32_t          waitSemaphoreCount;
2992     const  VkSemaphore *      pWaitSemaphores;
2993     const  VkPipelineStageFlags *           pWaitDstStageMask;
2994     uint32_t          commandBufferCount;
2995     const  VkCommandBuffer *      pCommandBuffers;
2996     uint32_t          signalSemaphoreCount;
2997     const  VkSemaphore *      pSignalSemaphores;
2998 } VkSubmitInfo;
2999
3000 typedef struct VkSurfaceCapabilitiesKHR {
3001     uint32_t                            minImageCount;
3002     uint32_t                            maxImageCount;
3003     VkExtent2D                          currentExtent;
3004     VkExtent2D                          minImageExtent;
3005     VkExtent2D                          maxImageExtent;
3006     uint32_t                            maxImageArrayLayers;
3007     VkSurfaceTransformFlagsKHR          supportedTransforms;
3008     VkSurfaceTransformFlagBitsKHR       currentTransform;
3009     VkCompositeAlphaFlagsKHR            supportedCompositeAlpha;
3010     VkImageUsageFlags                   supportedUsageFlags;
3011 } VkSurfaceCapabilitiesKHR;
3012
3013 typedef struct VkSwapchainCreateInfoKHR {
3014     VkStructureType   sType;
3015     const  void *                       pNext;
3016     VkSwapchainCreateFlagsKHR           flags;
3017     VkSurfaceKHR                        surface;
3018     uint32_t                            minImageCount;
3019     VkFormat                            imageFormat;
3020     VkColorSpaceKHR                     imageColorSpace;
3021     VkExtent2D                          imageExtent;
3022     uint32_t                            imageArrayLayers;
3023     VkImageUsageFlags                   imageUsage;
3024     VkSharingMode                       imageSharingMode;
3025     uint32_t            queueFamilyIndexCount;
3026     const  uint32_t *                    pQueueFamilyIndices;
3027     VkSurfaceTransformFlagBitsKHR       preTransform;
3028     VkCompositeAlphaFlagBitsKHR         compositeAlpha;
3029     VkPresentModeKHR                    presentMode;
3030     VkBool32                            clipped;
3031     VkSwapchainKHR     oldSwapchain;
3032 } VkSwapchainCreateInfoKHR;
3033
3034 typedef struct VkDebugReportCallbackCreateInfoEXT {
3035     VkStructureType   sType;
3036     const  void *                       pNext;
3037     VkDebugReportFlagsEXT               flags;
3038     PFN_vkDebugReportCallbackEXT        pfnCallback;
3039     void *              pUserData;
3040 } VkDebugReportCallbackCreateInfoEXT;
3041
3042 typedef struct VkPhysicalDeviceFeatures2 {
3043     VkStructureType   sType;
3044     void *                              pNext;
3045     VkPhysicalDeviceFeatures            features;
3046 } VkPhysicalDeviceFeatures2;
3047
3048 typedef struct VkFormatProperties2 {
3049     VkStructureType   sType;
3050     void *                              pNext;
3051     VkFormatProperties                  formatProperties;
3052 } VkFormatProperties2;
3053
3054 typedef struct VkImageFormatProperties2 {
3055     VkStructureType   sType;
3056     void *  pNext;
3057     VkImageFormatProperties             imageFormatProperties;
3058 } VkImageFormatProperties2;
```

```
3059
3060 typedef struct VkPhysicalDeviceImageFormatInfo2 {
3061     VkStructureType    sType;
3062     const  void *  pNext;
3063     VkFormat                            format;
3064     VkImageType                         type;
3065     VkImageTiling                       tiling;
3066     VkImageUsageFlags                   usage;
3067     VkImageCreateFlags    flags;
3068 } VkPhysicalDeviceImageFormatInfo2;
3069
3070 typedef struct VkQueueFamilyProperties2 {
3071     VkStructureType    sType;
3072     void *                              pNext;
3073     VkQueueFamilyProperties             queueFamilyProperties;
3074 } VkQueueFamilyProperties2;
3075
3076 typedef struct VkSparseImageFormatProperties2 {
3077     VkStructureType    sType;
3078     void *                              pNext;
3079     VkSparseImageFormatProperties       properties;
3080 } VkSparseImageFormatProperties2;
3081
3082 typedef struct VkPhysicalDeviceSparseImageFormatInfo2 {
3083     VkStructureType    sType;
3084     const  void *                        pNext;
3085     VkFormat                            format;
3086     VkImageType                         type;
3087     VkSampleCountFlagBits               samples;
3088     VkImageUsageFlags                   usage;
3089     VkImageTiling                       tiling;
3090 } VkPhysicalDeviceSparseImageFormatInfo2;
3091
3092 typedef struct VkPhysicalDeviceVariablePointersFeatures {
3093     VkStructureType    sType;
3094     void *                              pNext;
3095     VkBool32                            variablePointersStorageBuffer;
3096     VkBool32                            variablePointers;
3097 } VkPhysicalDeviceVariablePointersFeatures;
3098
3099 typedef struct VkPhysicalDeviceVariablePointersFeatures  VkPhysicalDeviceVariablePointerFeatures;
3100
3101 typedef struct VkExternalMemoryProperties {
3102     VkExternalMemoryFeatureFlags    externalMemoryFeatures;
3103     VkExternalMemoryHandleTypeFlags   exportFromImportedHandleTypes;
3104     VkExternalMemoryHandleTypeFlags   compatibleHandleTypes;
3105 } VkExternalMemoryProperties;
3106
3107 typedef struct VkExternalImageFormatProperties {
3108     VkStructureType    sType;
3109     void *                              pNext;
3110     VkExternalMemoryProperties   externalMemoryProperties;
3111 } VkExternalImageFormatProperties;
3112
3113 typedef struct VkPhysicalDeviceExternalBufferInfo {
3114     VkStructureType    sType;
3115     const  void *                        pNext;
3116     VkBufferCreateFlags   flags;
3117     VkBufferUsageFlags                  usage;
3118     VkExternalMemoryHandleTypeFlagBits   handleType;
3119 } VkPhysicalDeviceExternalBufferInfo;
3120
3121 typedef struct VkExternalBufferProperties {
3122     VkStructureType    sType;
3123     void *                              pNext;
3124     VkExternalMemoryProperties      externalMemoryProperties;
3125 } VkExternalBufferProperties;
3126
3127 typedef struct VkPhysicalDeviceIDProperties {
3128     VkStructureType    sType;
3129     void *                              pNext;
3130     uint8_t                             deviceUUID [ VK_UUID_SIZE ];
3131     uint8_t                             driverUUID [ VK_UUID_SIZE ];
3132     uint8_t                             deviceLUID [ VK_LUID_SIZE ];
3133     uint32_t                            deviceNodeMask;
3134     VkBool32                            deviceLUIDValid;
3135 } VkPhysicalDeviceIDProperties;
3136
3137 typedef struct VkExternalMemoryImageCreateInfo {
3138     VkStructureType    sType;
3139     const  void *                        pNext;
3140     VkExternalMemoryHandleTypeFlags   handleTypes;
3141 } VkExternalMemoryImageCreateInfo;
3142
3143 typedef struct VkExternalMemoryBufferCreateInfo {
3144     VkStructureType    sType;
3145     const  void *                        pNext;
```

```
3146     VkExternalMemoryHandleTypeFlags   handleTypes;
3147 } VkExternalMemoryBufferCreateInfo;
3148
3149 typedef struct VkExportMemoryAllocateInfo {
3150     VkStructureType   sType;
3151     const  void *                      pNext;
3152     VkExternalMemoryHandleTypeFlags   handleTypes;
3153 } VkExportMemoryAllocateInfo;
3154
3155 typedef struct VkExternalSemaphoreProperties {
3156     VkStructureType   sType;
3157     void *                            pNext;
3158     VkExternalSemaphoreHandleTypeFlags   exportFromImportedHandleTypes;
3159     VkExternalSemaphoreHandleTypeFlags   compatibleHandleTypes;
3160     VkExternalSemaphoreFeatureFlags   externalSemaphoreFeatures;
3161 } VkExternalSemaphoreProperties;
3162
3163 typedef struct VkExportSemaphoreCreateInfo {
3164     VkStructureType   sType;
3165     const  void *                      pNext;
3166     VkExternalSemaphoreHandleTypeFlags   handleTypes;
3167 } VkExportSemaphoreCreateInfo;
3168
3169 typedef struct VkExternalFenceProperties {
3170     VkStructureType   sType;
3171     void *                            pNext;
3172     VkExternalFenceHandleTypeFlags   exportFromImportedHandleTypes;
3173     VkExternalFenceHandleTypeFlags   compatibleHandleTypes;
3174     VkExternalFenceFeatureFlags   externalFenceFeatures;
3175 } VkExternalFenceProperties;
3176
3177 typedef struct VkExportFenceCreateInfo {
3178     VkStructureType   sType;
3179     const  void *                      pNext;
3180     VkExternalFenceHandleTypeFlags   handleTypes;
3181 } VkExportFenceCreateInfo;
3182
3183 typedef struct VkPhysicalDeviceMultiviewFeatures {
3184     VkStructureType   sType;
3185     void *                            pNext;
3186     VkBool32                          multiview;
3187     VkBool32                          multiviewGeometryShader;
3188     VkBool32                          multiviewTessellationShader;
3189 } VkPhysicalDeviceMultiviewFeatures;
3190
3191 typedef struct VkPhysicalDeviceGroupProperties {
3192     VkStructureType   sType;
3193     void *                            pNext;
3194     uint32_t                          physicalDeviceCount;
3195     VkPhysicalDevice                  physicalDevices [ VK_MAX_DEVICE_GROUP_SIZE ];
3196     VkBool32                          subsetAllocation;
3197 } VkPhysicalDeviceGroupProperties;
3198
3199 typedef struct VkMemoryAllocateFlagsInfo {
3200     VkStructureType   sType;
3201     const  void *                      pNext;
3202     VkMemoryAllocateFlags   flags;
3203     uint32_t                          deviceMask;
3204 } VkMemoryAllocateFlagsInfo;
3205
3206 typedef struct VkBindBufferMemoryInfo {
3207     VkStructureType   sType;
3208     const  void *                      pNext;
3209     VkBuffer                          buffer;
3210     VkDeviceMemory                    memory;
3211     VkDeviceSize                      memoryOffset;
3212 } VkBindBufferMemoryInfo;
3213
3214 typedef struct VkBindImageMemoryInfo {
3215     VkStructureType   sType;
3216     const  void *                      pNext;
3217     VkImage                           image;
3218     VkDeviceMemory                    memory;
3219     VkDeviceSize                      memoryOffset;
3220 } VkBindImageMemoryInfo;
3221
3222 typedef struct VkDeviceGroupPresentCapabilitiesKHR {
3223     VkStructureType   sType;
3224     const  void *                      pNext;
3225     uint32_t                          presentMask [ VK_MAX_DEVICE_GROUP_SIZE ];
3226     VkDeviceGroupPresentModeFlagsKHR   modes;
3227 } VkDeviceGroupPresentCapabilitiesKHR;
3228
3229 typedef struct VkDeviceGroupSwapchainCreateInfoKHR {
3230     VkStructureType   sType;
3231     const  void *                      pNext;
3232     VkDeviceGroupPresentModeFlagsKHR                          modes;
```

```
3233 } VkDeviceGroupSwapchainCreateInfoKHR;
3234
3235 typedef struct VkDescriptorUpdateTemplateCreateInfo {
3236     VkStructureType   sType;
3237     const  void *                               pNext;
3238     VkDescriptorUpdateTemplateCreateFlags      flags;
3239     uint32_t                    descriptorUpdateEntryCount;
3240     const  VkDescriptorUpdateTemplateEntry *  pDescriptorUpdateEntries;
3241     VkDescriptorUpdateTemplateType   templateType;
3242     VkDescriptorSetLayout   descriptorSetLayout;
3243     VkPipelineBindPoint   pipelineBindPoint;
3244     VkPipelineLayout pipelineLayout;
3245     uint32_t   set;
3246 } VkDescriptorUpdateTemplateCreateInfo;
3247
3248 typedef struct VkInputAttachmentAspectReference {
3249     uint32_t                          subpass;
3250     uint32_t                          inputAttachmentIndex;
3251     VkImageAspectFlags                aspectMask;
3252 } VkInputAttachmentAspectReference;
3253
3254 typedef struct VkRenderPassInputAttachmentAspectCreateInfo {
3255     VkStructureType   sType;
3256     const  void *                      pNext;
3257     uint32_t                          aspectReferenceCount;
3258     const  VkInputAttachmentAspectReference *  pAspectReferences;
3259 } VkRenderPassInputAttachmentAspectCreateInfo;
3260
3261 typedef struct VkPhysicalDevice16BitStorageFeatures {
3262     VkStructureType   sType;
3263     void *       pNext;
3264     VkBool32                           storageBuffer16BitAccess;
3265     VkBool32                           uniformAndStorageBuffer16BitAccess;
3266     VkBool32                           storagePushConstant16;
3267     VkBool32                           storageInputOutput16;
3268 } VkPhysicalDevice16BitStorageFeatures;
3269
3270 typedef struct VkPhysicalDeviceSubgroupProperties {
3271     VkStructureType   sType;
3272     void *                    pNext;
3273     uint32_t                        subgroupSize;
3274     VkShaderStageFlags              supportedStages;
3275     VkSubgroupFeatureFlags          supportedOperations;
3276     VkBool32   quadOperationsInAllStages;
3277 } VkPhysicalDeviceSubgroupProperties;
3278
3279 typedef struct VkMemoryRequirements2 {
3280     VkStructureType   sType;
3281     void *  pNext;
3282     VkMemoryRequirements                                                memoryRequirements;
3283 } VkMemoryRequirements2;
3284
3285 typedef struct VkSparseImageMemoryRequirements2 {
3286     VkStructureType   sType;
3287     void *                                        pNext;
3288     VkSparseImageMemoryRequirements                                     memoryRequirements;
3289 } VkSparseImageMemoryRequirements2;
3290
3291 typedef struct VkMemoryDedicatedRequirements {
3292     VkStructureType   sType;
3293     void *                            pNext;
3294     VkBool32                          prefersDedicatedAllocation;
3295     VkBool32                          requiresDedicatedAllocation;
3296 } VkMemoryDedicatedRequirements;
3297
3298 typedef struct VkImageViewUsageCreateInfo {
3299     VkStructureType   sType;
3300     const  void *  pNext;
3301     VkImageUsageFlags   usage;
3302 } VkImageViewUsageCreateInfo;
3303
3304 typedef struct VkSamplerYcbcrConversionCreateInfo {
3305     VkStructureType   sType;
3306     const  void *                      pNext;
3307     VkFormat                          format;
3308     VkSamplerYcbcrModelConversion   ycbcrModel;
3309     VkSamplerYcbcrRange             ycbcrRange;
3310     VkComponentMapping                 components;
3311     VkChromaLocation                xChromaOffset;
3312     VkChromaLocation                yChromaOffset;
3313     VkFilter                           chromaFilter;
3314     VkBool32                           forceExplicitReconstruction;
3315 } VkSamplerYcbcrConversionCreateInfo;
3316
3317 typedef struct VkPhysicalDeviceSamplerYcbcrConversionFeatures {
3318     VkStructureType   sType;
3319     void *       pNext;
```

```
3320     VkBool32                            samplerYcbcrConversion;
3321 } VkPhysicalDeviceSamplerYcbcrConversionFeatures;
3322
3323 typedef struct VkProtectedSubmitInfo {
3324     VkStructureType    sType;
3325     const  void *                       pNext;
3326     VkBool32                           protectedSubmit;
3327 } VkProtectedSubmitInfo;
3328
3329 typedef struct VkPhysicalDeviceProtectedMemoryFeatures {
3330     VkStructureType    sType;
3331     void *                                pNext;
3332     VkBool32                              protectedMemory;
3333 } VkPhysicalDeviceProtectedMemoryFeatures;
3334
3335 typedef struct VkPhysicalDeviceProtectedMemoryProperties {
3336     VkStructureType    sType;
3337     void *                                pNext;
3338     VkBool32                              protectedNoFault;
3339 } VkPhysicalDeviceProtectedMemoryProperties;
3340
3341 typedef struct VkDeviceQueueInfo2 {
3342     VkStructureType    sType;
3343     const  void *                         pNext;
3344     VkDeviceQueueCreateFlags              flags;
3345     uint32_t                              queueFamilyIndex;
3346     uint32_t                              queueIndex;
3347 } VkDeviceQueueInfo2;
3348
3349 typedef struct VkPhysicalDeviceMaintenance3Properties {
3350     VkStructureType    sType;
3351     void *                              pNext;
3352     uint32_t                            maxPerSetDescriptors;
3353     VkDeviceSize                        maxMemoryAllocationSize;
3354 } VkPhysicalDeviceMaintenance3Properties;
3355
3356 typedef struct VkDescriptorSetLayoutSupport {
3357     VkStructureType    sType;
3358     void *              pNext;
3359     VkBool32            supported;
3360 } VkDescriptorSetLayoutSupport;
3361
3362 typedef struct VkPhysicalDeviceShaderDrawParametersFeatures {
3363     VkStructureType    sType;
3364     void *                              pNext;
3365     VkBool32                            shaderDrawParameters;
3366 } VkPhysicalDeviceShaderDrawParametersFeatures;
3367
3368 typedef struct VkPhysicalDeviceShaderDrawParametersFeatures
       VkPhysicalDeviceShaderDrawParameterFeatures;
3369
3370 typedef struct VkPhysicalDeviceProperties {
3371     uint32_t          apiVersion;
3372     uint32_t          driverVersion;
3373     uint32_t          vendorID;
3374     uint32_t          deviceID;
3375     VkPhysicalDeviceType    deviceType;
3376     char              deviceName [ VK_MAX_PHYSICAL_DEVICE_NAME_SIZE ];
3377     uint8_t           pipelineCacheUUID [ VK_UUID_SIZE ];
3378     VkPhysicalDeviceLimits    limits;
3379     VkPhysicalDeviceSparseProperties    sparseProperties;
3380 } VkPhysicalDeviceProperties;
3381
3382 typedef struct VkDeviceCreateInfo {
3383     VkStructureType    sType;
3384     const  void *       pNext;
3385     VkDeviceCreateFlags      flags;
3386     uint32_t           queueCreateInfoCount;
3387     const  VkDeviceQueueCreateInfo *  pQueueCreateInfos;
3388     uint32_t                 enabledLayerCount;
3389     const  char * const*       ppEnabledLayerNames;
3390     uint32_t                 enabledExtensionCount;
3391     const  char * const*       ppEnabledExtensionNames;
3392     const  VkPhysicalDeviceFeatures *  pEnabledFeatures;
3393 } VkDeviceCreateInfo;
3394
3395 typedef struct VkPhysicalDeviceMemoryProperties {
3396     uint32_t                 memoryTypeCount;
3397     VkMemoryType             memoryTypes [ VK_MAX_MEMORY_TYPES ];
3398     uint32_t                 memoryHeapCount;
3399     VkMemoryHeap             memoryHeaps [ VK_MAX_MEMORY_HEAPS ];
3400 } VkPhysicalDeviceMemoryProperties;
3401
3402 typedef struct VkPhysicalDeviceProperties2 {
3403     VkStructureType    sType;
3404     void *                              pNext;
3405     VkPhysicalDeviceProperties          properties;
```

```
3406 } VkPhysicalDeviceProperties2;
3407
3408 typedef struct VkPhysicalDeviceMemoryProperties2 {
3409     VkStructureType   sType;
3410     void *                                  pNext;
3411     VkPhysicalDeviceMemoryProperties   memoryProperties;
3412 } VkPhysicalDeviceMemoryProperties2;
3413
3414
3415
3416 #define VK_VERSION_1_0 1
3417 GLAD_API_CALL int GLAD_VK_VERSION_1_0;
3418 #define VK_VERSION_1_1 1
3419 GLAD_API_CALL int GLAD_VK_VERSION_1_1;
3420 #define VK_EXT_debug_report 1
3421 GLAD_API_CALL int GLAD_VK_EXT_debug_report;
3422 #define VK_KHR_surface 1
3423 GLAD_API_CALL int GLAD_VK_KHR_surface;
3424 #define VK_KHR_swapchain 1
3425 GLAD_API_CALL int GLAD_VK_KHR_swapchain;
3426
3427
3428 typedef VkResult (GLAD_API_PTR *PFN_vkAcquireNextImage2KHR)(VkDevice device, const
       VkAcquireNextImageInfoKHR * pAcquireInfo, uint32_t * pImageIndex);
3429 typedef VkResult (GLAD_API_PTR *PFN_vkAcquireNextImageKHR)(VkDevice device, VkSwapchainKHR swapchain,
       uint64_t timeout, VkSemaphore semaphore, VkFence fence, uint32_t * pImageIndex);
3430 typedef VkResult (GLAD_API_PTR *PFN_vkAllocateCommandBuffers)(VkDevice device, const
       VkCommandBufferAllocateInfo * pAllocateInfo, VkCommandBuffer * pCommandBuffers);
3431 typedef VkResult (GLAD_API_PTR *PFN_vkAllocateDescriptorSets)(VkDevice device, const
       VkDescriptorSetAllocateInfo * pAllocateInfo, VkDescriptorSet * pDescriptorSets);
3432 typedef VkResult (GLAD_API_PTR *PFN_vkAllocateMemory)(VkDevice device, const VkMemoryAllocateInfo *
       pAllocateInfo, const VkAllocationCallbacks * pAllocator, VkDeviceMemory * pMemory);
3433 typedef VkResult (GLAD_API_PTR *PFN_vkBeginCommandBuffer)(VkCommandBuffer commandBuffer, const
       VkCommandBufferBeginInfo * pBeginInfo);
3434 typedef VkResult (GLAD_API_PTR *PFN_vkBindBufferMemory)(VkDevice device, VkBuffer buffer,
       VkDeviceMemory memory, VkDeviceSize memoryOffset);
3435 typedef VkResult (GLAD_API_PTR *PFN_vkBindBufferMemory2)(VkDevice device, uint32_t bindInfoCount, const
       VkBindBufferMemoryInfo * pBindInfos);
3436 typedef VkResult (GLAD_API_PTR *PFN_vkBindImageMemory)(VkDevice device, VkImage image, VkDeviceMemory
       memory, VkDeviceSize memoryOffset);
3437 typedef VkResult (GLAD_API_PTR *PFN_vkBindImageMemory2)(VkDevice device, uint32_t bindInfoCount, const
       VkBindImageMemoryInfo * pBindInfos);
3438 typedef void (GLAD_API_PTR *PFN_vkCmdBeginQuery)(VkCommandBuffer commandBuffer, VkQueryPool queryPool,
       uint32_t query, VkQueryControlFlags flags);
3439 typedef void (GLAD_API_PTR *PFN_vkCmdBeginRenderPass)(VkCommandBuffer commandBuffer, const
       VkRenderPassBeginInfo * pRenderPassBegin, VkSubpassContents contents);
3440 typedef void (GLAD_API_PTR *PFN_vkCmdBindDescriptorSets)(VkCommandBuffer commandBuffer,
       VkPipelineBindPoint pipelineBindPoint, VkPipelineLayout layout, uint32_t firstSet, uint32_t
       descriptorSetCount, const VkDescriptorSet * pDescriptorSets, uint32_t dynamicOffsetCount, const
       uint32_t * pDynamicOffsets);
3441 typedef void (GLAD_API_PTR *PFN_vkCmdBindIndexBuffer)(VkCommandBuffer commandBuffer, VkBuffer buffer,
       VkDeviceSize offset, VkIndexType indexType);
3442 typedef void (GLAD_API_PTR *PFN_vkCmdBindPipeline)(VkCommandBuffer commandBuffer, VkPipelineBindPoint
       pipelineBindPoint, VkPipeline pipeline);
3443 typedef void (GLAD_API_PTR *PFN_vkCmdBindVertexBuffers)(VkCommandBuffer commandBuffer, uint32_t
       firstBinding, uint32_t bindingCount, const VkBuffer * pBuffers, const VkDeviceSize * pOffsets);
3444 typedef void (GLAD_API_PTR *PFN_vkCmdBlitImage)(VkCommandBuffer commandBuffer, VkImage srcImage,
       VkImageLayout srcImageLayout, VkImage dstImage, VkImageLayout dstImageLayout, uint32_t regionCount,
       const VkImageBlit * pRegions, VkFilter filter);
3445 typedef void (GLAD_API_PTR *PFN_vkCmdClearAttachments)(VkCommandBuffer commandBuffer, uint32_t
       attachmentCount, const VkClearAttachment * pAttachments, uint32_t rectCount, const VkClearRect *
       pRects);
3446 typedef void (GLAD_API_PTR *PFN_vkCmdClearColorImage)(VkCommandBuffer commandBuffer, VkImage image,
       VkImageLayout imageLayout, const VkClearColorValue * pColor, uint32_t rangeCount, const
       VkImageSubresourceRange * pRanges);
3447 typedef void (GLAD_API_PTR *PFN_vkCmdClearDepthStencilImage)(VkCommandBuffer commandBuffer, VkImage
       image, VkImageLayout imageLayout, const VkClearDepthStencilValue * pDepthStencil, uint32_t
       rangeCount, const VkImageSubresourceRange * pRanges);
3448 typedef void (GLAD_API_PTR *PFN_vkCmdCopyBuffer)(VkCommandBuffer commandBuffer, VkBuffer srcBuffer,
       VkBuffer dstBuffer, uint32_t regionCount, const VkBufferCopy * pRegions);
3449 typedef void (GLAD_API_PTR *PFN_vkCmdCopyBufferToImage)(VkCommandBuffer commandBuffer, VkBuffer
       srcBuffer, VkImage dstImage, VkImageLayout dstImageLayout, uint32_t regionCount, const
       VkBufferImageCopy * pRegions);
3450 typedef void (GLAD_API_PTR *PFN_vkCmdCopyImage)(VkCommandBuffer commandBuffer, VkImage srcImage,
       VkImageLayout srcImageLayout, VkImage dstImage, VkImageLayout dstImageLayout, uint32_t regionCount,
       const VkImageCopy * pRegions);
3451 typedef void (GLAD_API_PTR *PFN_vkCmdCopyImageToBuffer)(VkCommandBuffer commandBuffer, VkImage
       srcImage, VkImageLayout srcImageLayout, VkBuffer dstBuffer, uint32_t regionCount, const
       VkBufferImageCopy * pRegions);
3452 typedef void (GLAD_API_PTR *PFN_vkCmdCopyQueryPoolResults)(VkCommandBuffer commandBuffer, VkQueryPool
       queryPool, uint32_t firstQuery, uint32_t queryCount, VkBuffer dstBuffer, VkDeviceSize dstOffset,
       VkDeviceSize stride, VkQueryResultFlags flags);
3453 typedef void (GLAD_API_PTR *PFN_vkCmdDispatch)(VkCommandBuffer commandBuffer, uint32_t groupCountX,
       uint32_t groupCountY, uint32_t groupCountZ);
3454 typedef void (GLAD_API_PTR *PFN_vkCmdDispatchBase)(VkCommandBuffer commandBuffer, uint32_t baseGroupX,
       uint32_t baseGroupY, uint32_t baseGroupZ, uint32_t groupCountX, uint32_t groupCountY, uint32_t
       groupCountZ);
```

```
3455 typedef void (GLAD_API_PTR *PFN_vkCmdDispatchIndirect)(VkCommandBuffer commandBuffer, VkBuffer buffer,
       VkDeviceSize offset);
3456 typedef void (GLAD_API_PTR *PFN_vkCmdDraw)(VkCommandBuffer commandBuffer, uint32_t vertexCount,
       uint32_t instanceCount, uint32_t firstVertex, uint32_t firstInstance);
3457 typedef void (GLAD_API_PTR *PFN_vkCmdDrawIndexed)(VkCommandBuffer commandBuffer, uint32_t indexCount,
       uint32_t instanceCount, uint32_t firstIndex, int32_t vertexOffset, uint32_t firstInstance);
3458 typedef void (GLAD_API_PTR *PFN_vkCmdDrawIndexedIndirect)(VkCommandBuffer commandBuffer, VkBuffer
       buffer, VkDeviceSize offset, uint32_t drawCount, uint32_t stride);
3459 typedef void (GLAD_API_PTR *PFN_vkCmdDrawIndirect)(VkCommandBuffer commandBuffer, VkBuffer buffer,
       VkDeviceSize offset, uint32_t drawCount, uint32_t stride);
3460 typedef void (GLAD_API_PTR *PFN_vkCmdEndQuery)(VkCommandBuffer commandBuffer, VkQueryPool queryPool,
       uint32_t query);
3461 typedef void (GLAD_API_PTR *PFN_vkCmdEndRenderPass)(VkCommandBuffer commandBuffer);
3462 typedef void (GLAD_API_PTR *PFN_vkCmdExecuteCommands)(VkCommandBuffer commandBuffer, uint32_t
       commandBufferCount, const VkCommandBuffer * pCommandBuffers);
3463 typedef void (GLAD_API_PTR *PFN_vkCmdFillBuffer)(VkCommandBuffer commandBuffer, VkBuffer dstBuffer,
       VkDeviceSize dstOffset, VkDeviceSize size, uint32_t data);
3464 typedef void (GLAD_API_PTR *PFN_vkCmdNextSubpass)(VkCommandBuffer commandBuffer, VkSubpassContents
       contents);
3465 typedef void (GLAD_API_PTR *PFN_vkCmdPipelineBarrier)(VkCommandBuffer commandBuffer,
       VkPipelineStageFlags srcStageMask, VkPipelineStageFlags dstStageMask, VkDependencyFlags
       dependencyFlags, uint32_t memoryBarrierCount, const VkMemoryBarrier * pMemoryBarriers, uint32_t
       bufferMemoryBarrierCount, const VkBufferMemoryBarrier * pBufferMemoryBarriers, uint32_t
       imageMemoryBarrierCount, const VkImageMemoryBarrier * pImageMemoryBarriers);
3466 typedef void (GLAD_API_PTR *PFN_vkCmdPushConstants)(VkCommandBuffer commandBuffer, VkPipelineLayout
       layout, VkShaderStageFlags stageFlags, uint32_t offset, uint32_t size, const void * pValues);
3467 typedef void (GLAD_API_PTR *PFN_vkCmdResetEvent)(VkCommandBuffer commandBuffer, VkEvent event,
       VkPipelineStageFlags stageMask);
3468 typedef void (GLAD_API_PTR *PFN_vkCmdResetQueryPool)(VkCommandBuffer commandBuffer, VkQueryPool
       queryPool, uint32_t firstQuery, uint32_t queryCount);
3469 typedef void (GLAD_API_PTR *PFN_vkCmdResolveImage)(VkCommandBuffer commandBuffer, VkImage srcImage,
       VkImageLayout srcImageLayout, VkImage dstImage, VkImageLayout dstImageLayout, uint32_t regionCount,
       const VkImageResolve * pRegions);
3470 typedef void (GLAD_API_PTR *PFN_vkCmdSetBlendConstants)(VkCommandBuffer commandBuffer, const float
       blendConstants [4]);
3471 typedef void (GLAD_API_PTR *PFN_vkCmdSetDepthBias)(VkCommandBuffer commandBuffer, float
       depthBiasConstantFactor, float depthBiasClamp, float depthBiasSlopeFactor);
3472 typedef void (GLAD_API_PTR *PFN_vkCmdSetDepthBounds)(VkCommandBuffer commandBuffer, float
       minDepthBounds, float maxDepthBounds);
3473 typedef void (GLAD_API_PTR *PFN_vkCmdSetDeviceMask)(VkCommandBuffer commandBuffer, uint32_t
       deviceMask);
3474 typedef void (GLAD_API_PTR *PFN_vkCmdSetEvent)(VkCommandBuffer commandBuffer, VkEvent event,
       VkPipelineStageFlags stageMask);
3475 typedef void (GLAD_API_PTR *PFN_vkCmdSetLineWidth)(VkCommandBuffer commandBuffer, float lineWidth);
3476 typedef void (GLAD_API_PTR *PFN_vkCmdSetScissor)(VkCommandBuffer commandBuffer, uint32_t firstScissor,
       uint32_t scissorCount, const VkRect2D * pScissors);
3477 typedef void (GLAD_API_PTR *PFN_vkCmdSetStencilCompareMask)(VkCommandBuffer commandBuffer,
       VkStencilFaceFlags faceMask, uint32_t compareMask);
3478 typedef void (GLAD_API_PTR *PFN_vkCmdSetStencilReference)(VkCommandBuffer commandBuffer,
       VkStencilFaceFlags faceMask, uint32_t reference);
3479 typedef void (GLAD_API_PTR *PFN_vkCmdSetStencilWriteMask)(VkCommandBuffer commandBuffer,
       VkStencilFaceFlags faceMask, uint32_t writeMask);
3480 typedef void (GLAD_API_PTR *PFN_vkCmdSetViewport)(VkCommandBuffer commandBuffer, uint32_t
       firstViewport, uint32_t viewportCount, const VkViewport * pViewports);
3481 typedef void (GLAD_API_PTR *PFN_vkCmdUpdateBuffer)(VkCommandBuffer commandBuffer, VkBuffer dstBuffer,
       VkDeviceSize dstOffset, VkDeviceSize dataSize, const void * pData);
3482 typedef void (GLAD_API_PTR *PFN_vkCmdWaitEvents)(VkCommandBuffer commandBuffer, uint32_t eventCount,
       const VkEvent * pEvents, VkPipelineStageFlags srcStageMask, VkPipelineStageFlags dstStageMask,
       uint32_t memoryBarrierCount, const VkMemoryBarrier * pMemoryBarriers, uint32_t
       bufferMemoryBarrierCount, const VkBufferMemoryBarrier * pBufferMemoryBarriers, uint32_t
       imageMemoryBarrierCount, const VkImageMemoryBarrier * pImageMemoryBarriers);
3483 typedef void (GLAD_API_PTR *PFN_vkCmdWriteTimestamp)(VkCommandBuffer commandBuffer,
       VkPipelineStageFlagBits pipelineStage, VkQueryPool queryPool, uint32_t query);
3484 typedef VkResult (GLAD_API_PTR *PFN_vkCreateBuffer)(VkDevice device, const VkBufferCreateInfo *
       pCreateInfo, const VkAllocationCallbacks * pAllocator, VkBuffer * pBuffer);
3485 typedef VkResult (GLAD_API_PTR *PFN_vkCreateBufferView)(VkDevice device, const VkBufferViewCreateInfo *
       pCreateInfo, const VkAllocationCallbacks * pAllocator, VkBufferView * pView);
3486 typedef VkResult (GLAD_API_PTR *PFN_vkCreateCommandPool)(VkDevice device, const VkCommandPoolCreateInfo
       * pCreateInfo, const VkAllocationCallbacks * pAllocator, VkCommandPool * pCommandPool);
3487 typedef VkResult (GLAD_API_PTR *PFN_vkCreateComputePipelines)(VkDevice device, VkPipelineCache
       pipelineCache, uint32_t createInfoCount, const VkComputePipelineCreateInfo * pCreateInfos, const
       VkAllocationCallbacks * pAllocator, VkPipeline * pPipelines);
3488 typedef VkResult (GLAD_API_PTR *PFN_vkCreateDebugReportCallbackEXT)(VkInstance instance, const
       VkDebugReportCallbackCreateInfoEXT * pCreateInfo, const VkAllocationCallbacks * pAllocator,
       VkDebugReportCallbackEXT * pCallback);
3489 typedef VkResult (GLAD_API_PTR *PFN_vkCreateDescriptorPool)(VkDevice device, const
       VkDescriptorPoolCreateInfo * pCreateInfo, const VkAllocationCallbacks * pAllocator, VkDescriptorPool
       * pDescriptorPool);
3490 typedef VkResult (GLAD_API_PTR *PFN_vkCreateDescriptorSetLayout)(VkDevice device, const
       VkDescriptorSetLayoutCreateInfo * pCreateInfo, const VkAllocationCallbacks * pAllocator,
       VkDescriptorSetLayout * pSetLayout);
3491 typedef VkResult (GLAD_API_PTR *PFN_vkCreateDescriptorUpdateTemplate)(VkDevice device, const
       VkDescriptorUpdateTemplateCreateInfo * pCreateInfo, const VkAllocationCallbacks * pAllocator,
       VkDescriptorUpdateTemplate * pDescriptorUpdateTemplate);
3492 typedef VkResult (GLAD_API_PTR *PFN_vkCreateDevice)(VkPhysicalDevice physicalDevice, const
       VkDeviceCreateInfo * pCreateInfo, const VkAllocationCallbacks * pAllocator, VkDevice * pDevice);
3493 typedef VkResult (GLAD_API_PTR *PFN_vkCreateEvent)(VkDevice device, const VkEventCreateInfo *
```

```
    pCreateInfo, const VkAllocationCallbacks * pAllocator, VkEvent * pEvent);
3494 typedef VkResult (GLAD_API_PTR *PFN_vkCreateFence)(VkDevice device, const VkFenceCreateInfo *
    pCreateInfo, const VkAllocationCallbacks * pAllocator, VkFence * pFence);
3495 typedef VkResult (GLAD_API_PTR *PFN_vkCreateFramebuffer)(VkDevice device, const VkFramebufferCreateInfo
    * pCreateInfo, const VkAllocationCallbacks * pAllocator, VkFramebuffer * pFramebuffer);
3496 typedef VkResult (GLAD_API_PTR *PFN_vkCreateGraphicsPipelines)(VkDevice device, VkPipelineCache
    pipelineCache, uint32_t createInfoCount, const VkGraphicsPipelineCreateInfo * pCreateInfos, const
    VkAllocationCallbacks * pAllocator, VkPipeline * pPipelines);
3497 typedef VkResult (GLAD_API_PTR *PFN_vkCreateImage)(VkDevice device, const VkImageCreateInfo *
    pCreateInfo, const VkAllocationCallbacks * pAllocator, VkImage * pImage);
3498 typedef VkResult (GLAD_API_PTR *PFN_vkCreateImageView)(VkDevice device, const VkImageViewCreateInfo *
    pCreateInfo, const VkAllocationCallbacks * pAllocator, VkImageView * pView);
3499 typedef VkResult (GLAD_API_PTR *PFN_vkCreateInstance)(const VkInstanceCreateInfo * pCreateInfo, const
    VkAllocationCallbacks * pAllocator, VkInstance * pInstance);
3500 typedef VkResult (GLAD_API_PTR *PFN_vkCreatePipelineCache)(VkDevice device, const
    VkPipelineCacheCreateInfo * pCreateInfo, const VkAllocationCallbacks * pAllocator, VkPipelineCache *
    pPipelineCache);
3501 typedef VkResult (GLAD_API_PTR *PFN_vkCreatePipelineLayout)(VkDevice device, const
    VkPipelineLayoutCreateInfo * pCreateInfo, const VkAllocationCallbacks * pAllocator, VkPipelineLayout
    * pPipelineLayout);
3502 typedef VkResult (GLAD_API_PTR *PFN_vkCreateQueryPool)(VkDevice device, const VkQueryPoolCreateInfo *
    pCreateInfo, const VkAllocationCallbacks * pAllocator, VkQueryPool * pQueryPool);
3503 typedef VkResult (GLAD_API_PTR *PFN_vkCreateRenderPass)(VkDevice device, const VkRenderPassCreateInfo *
    pCreateInfo, const VkAllocationCallbacks * pAllocator, VkRenderPass * pRenderPass);
3504 typedef VkResult (GLAD_API_PTR *PFN_vkCreateSampler)(VkDevice device, const VkSamplerCreateInfo *
    pCreateInfo, const VkAllocationCallbacks * pAllocator, VkSampler * pSampler);
3505 typedef VkResult (GLAD_API_PTR *PFN_vkCreateSamplerYcbcrConversion)(VkDevice device, const
    VkSamplerYcbcrConversionCreateInfo * pCreateInfo, const VkAllocationCallbacks * pAllocator,
    VkSamplerYcbcrConversion * pYcbcrConversion);
3506 typedef VkResult (GLAD_API_PTR *PFN_vkCreateSemaphore)(VkDevice device, const VkSemaphoreCreateInfo *
    pCreateInfo, const VkAllocationCallbacks * pAllocator, VkSemaphore * pSemaphore);
3507 typedef VkResult (GLAD_API_PTR *PFN_vkCreateShaderModule)(VkDevice device, const
    VkShaderModuleCreateInfo * pCreateInfo, const VkAllocationCallbacks * pAllocator, VkShaderModule *
    pShaderModule);
3508 typedef VkResult (GLAD_API_PTR *PFN_vkCreateSwapchainKHR)(VkDevice device, const
    VkSwapchainCreateInfoKHR * pCreateInfo, const VkAllocationCallbacks * pAllocator, VkSwapchainKHR *
    pSwapchain);
3509 typedef void (GLAD_API_PTR *PFN_vkDebugReportMessageEXT)(VkInstance instance, VkDebugReportFlagsEXT
    flags, VkDebugReportObjectTypeEXT objectType, uint64_t object, size_t location, int32_t messageCode,
    const char * pLayerPrefix, const char * pMessage);
3510 typedef void (GLAD_API_PTR *PFN_vkDestroyBuffer)(VkDevice device, VkBuffer buffer, const
    VkAllocationCallbacks * pAllocator);
3511 typedef void (GLAD_API_PTR *PFN_vkDestroyBufferView)(VkDevice device, VkBufferView bufferView, const
    VkAllocationCallbacks * pAllocator);
3512 typedef void (GLAD_API_PTR *PFN_vkDestroyCommandPool)(VkDevice device, VkCommandPool commandPool, const
    VkAllocationCallbacks * pAllocator);
3513 typedef void (GLAD_API_PTR *PFN_vkDestroyDebugReportCallbackEXT)(VkInstance instance,
    VkDebugReportCallbackEXT callback, const VkAllocationCallbacks * pAllocator);
3514 typedef void (GLAD_API_PTR *PFN_vkDestroyDescriptorPool)(VkDevice device, VkDescriptorPool
    descriptorPool, const VkAllocationCallbacks * pAllocator);
3515 typedef void (GLAD_API_PTR *PFN_vkDestroyDescriptorSetLayout)(VkDevice device, VkDescriptorSetLayout
    descriptorSetLayout, const VkAllocationCallbacks * pAllocator);
3516 typedef void (GLAD_API_PTR *PFN_vkDestroyDescriptorUpdateTemplate)(VkDevice device,
    VkDescriptorUpdateTemplate descriptorUpdateTemplate, const VkAllocationCallbacks * pAllocator);
3517 typedef void (GLAD_API_PTR *PFN_vkDestroyDevice)(VkDevice device, const VkAllocationCallbacks *
    pAllocator);
3518 typedef void (GLAD_API_PTR *PFN_vkDestroyEvent)(VkDevice device, VkEvent event, const
    VkAllocationCallbacks * pAllocator);
3519 typedef void (GLAD_API_PTR *PFN_vkDestroyFence)(VkDevice device, VkFence fence, const
    VkAllocationCallbacks * pAllocator);
3520 typedef void (GLAD_API_PTR *PFN_vkDestroyFramebuffer)(VkDevice device, VkFramebuffer framebuffer, const
    VkAllocationCallbacks * pAllocator);
3521 typedef void (GLAD_API_PTR *PFN_vkDestroyImage)(VkDevice device, VkImage image, const
    VkAllocationCallbacks * pAllocator);
3522 typedef void (GLAD_API_PTR *PFN_vkDestroyImageView)(VkDevice device, VkImageView imageView, const
    VkAllocationCallbacks * pAllocator);
3523 typedef void (GLAD_API_PTR *PFN_vkDestroyInstance)(VkInstance instance, const VkAllocationCallbacks *
    pAllocator);
3524 typedef void (GLAD_API_PTR *PFN_vkDestroyPipeline)(VkDevice device, VkPipeline pipeline, const
    VkAllocationCallbacks * pAllocator);
3525 typedef void (GLAD_API_PTR *PFN_vkDestroyPipelineCache)(VkDevice device, VkPipelineCache pipelineCache,
    const VkAllocationCallbacks * pAllocator);
3526 typedef void (GLAD_API_PTR *PFN_vkDestroyPipelineLayout)(VkDevice device, VkPipelineLayout
    pipelineLayout, const VkAllocationCallbacks * pAllocator);
3527 typedef void (GLAD_API_PTR *PFN_vkDestroyQueryPool)(VkDevice device, VkQueryPool queryPool, const
    VkAllocationCallbacks * pAllocator);
3528 typedef void (GLAD_API_PTR *PFN_vkDestroyRenderPass)(VkDevice device, VkRenderPass renderPass, const
    VkAllocationCallbacks * pAllocator);
3529 typedef void (GLAD_API_PTR *PFN_vkDestroySampler)(VkDevice device, VkSampler sampler, const
    VkAllocationCallbacks * pAllocator);
3530 typedef void (GLAD_API_PTR *PFN_vkDestroySamplerYcbcrConversion)(VkDevice device,
    VkSamplerYcbcrConversion ycbcrConversion, const VkAllocationCallbacks * pAllocator);
3531 typedef void (GLAD_API_PTR *PFN_vkDestroySemaphore)(VkDevice device, VkSemaphore semaphore, const
    VkAllocationCallbacks * pAllocator);
3532 typedef void (GLAD_API_PTR *PFN_vkDestroyShaderModule)(VkDevice device, VkShaderModule shaderModule,
    const VkAllocationCallbacks * pAllocator);
3533 typedef void (GLAD_API_PTR *PFN_vkDestroySurfaceKHR)(VkInstance instance, VkSurfaceKHR surface, const
```

```
      VkAllocationCallbacks * pAllocator);
3534 typedef void (GLAD_API_PTR *PFN_vkDestroySwapchainKHR)(VkDevice device, VkSwapchainKHR swapchain, const
      VkAllocationCallbacks * pAllocator);
3535 typedef VkResult (GLAD_API_PTR *PFN_vkDeviceWaitIdle)(VkDevice device);
3536 typedef VkResult (GLAD_API_PTR *PFN_vkEndCommandBuffer)(VkCommandBuffer commandBuffer);
3537 typedef VkResult (GLAD_API_PTR *PFN_vkEnumerateDeviceExtensionProperties)(VkPhysicalDevice
      physicalDevice, const char * pLayerName, uint32_t * pPropertyCount, VkExtensionProperties *
      pProperties);
3538 typedef VkResult (GLAD_API_PTR *PFN_vkEnumerateDeviceLayerProperties)(VkPhysicalDevice physicalDevice,
      uint32_t * pPropertyCount, VkLayerProperties * pProperties);
3539 typedef VkResult (GLAD_API_PTR *PFN_vkEnumerateInstanceExtensionProperties)(const char * pLayerName,
      uint32_t * pPropertyCount, VkExtensionProperties * pProperties);
3540 typedef VkResult (GLAD_API_PTR *PFN_vkEnumerateInstanceLayerProperties)(uint32_t * pPropertyCount,
      VkLayerProperties * pProperties);
3541 typedef VkResult (GLAD_API_PTR *PFN_vkEnumerateInstanceVersion)(uint32_t * pApiVersion);
3542 typedef VkResult (GLAD_API_PTR *PFN_vkEnumeratePhysicalDeviceGroups)(VkInstance instance, uint32_t *
      pPhysicalDeviceGroupCount, VkPhysicalDeviceGroupProperties * pPhysicalDeviceGroupProperties);
3543 typedef VkResult (GLAD_API_PTR *PFN_vkEnumeratePhysicalDevices)(VkInstance instance, uint32_t *
      pPhysicalDeviceCount, VkPhysicalDevice * pPhysicalDevices);
3544 typedef VkResult (GLAD_API_PTR *PFN_vkFlushMappedMemoryRanges)(VkDevice device, uint32_t
      memoryRangeCount, const VkMappedMemoryRange * pMemoryRanges);
3545 typedef void (GLAD_API_PTR *PFN_vkFreeCommandBuffers)(VkDevice device, VkCommandPool commandPool,
      uint32_t commandBufferCount, const VkCommandBuffer * pCommandBuffers);
3546 typedef VkResult (GLAD_API_PTR *PFN_vkFreeDescriptorSets)(VkDevice device, VkDescriptorPool
      descriptorPool, uint32_t descriptorSetCount, const VkDescriptorSet * pDescriptorSets);
3547 typedef void (GLAD_API_PTR *PFN_vkFreeMemory)(VkDevice device, VkDeviceMemory memory, const
      VkAllocationCallbacks * pAllocator);
3548 typedef void (GLAD_API_PTR *PFN_vkGetBufferMemoryRequirements)(VkDevice device, VkBuffer buffer,
      VkMemoryRequirements * pMemoryRequirements);
3549 typedef void (GLAD_API_PTR *PFN_vkGetBufferMemoryRequirements2)(VkDevice device, const
      VkBufferMemoryRequirementsInfo2 * pInfo, VkMemoryRequirements2 * pMemoryRequirements);
3550 typedef void (GLAD_API_PTR *PFN_vkGetDescriptorSetLayoutSupport)(VkDevice device, const
      VkDescriptorSetLayoutCreateInfo * pCreateInfo, VkDescriptorSetLayoutSupport * pSupport);
3551 typedef void (GLAD_API_PTR *PFN_vkGetDeviceGroupPeerMemoryFeatures)(VkDevice device, uint32_t
      heapIndex, uint32_t localDeviceIndex, uint32_t remoteDeviceIndex, VkPeerMemoryFeatureFlags *
      pPeerMemoryFeatures);
3552 typedef VkResult (GLAD_API_PTR *PFN_vkGetDeviceGroupPresentCapabilitiesKHR)(VkDevice device,
      VkDeviceGroupPresentCapabilitiesKHR * pDeviceGroupPresentCapabilities);
3553 typedef VkResult (GLAD_API_PTR *PFN_vkGetDeviceGroupSurfacePresentModesKHR)(VkDevice device,
      VkSurfaceKHR surface, VkDeviceGroupPresentModeFlagsKHR * pModes);
3554 typedef void (GLAD_API_PTR *PFN_vkGetDeviceMemoryCommitment)(VkDevice device, VkDeviceMemory memory,
      VkDeviceSize * pCommittedMemoryInBytes);
3555 typedef PFN_vkVoidFunction (GLAD_API_PTR *PFN_vkGetDeviceProcAddr)(VkDevice device, const char *
      pName);
3556 typedef void (GLAD_API_PTR *PFN_vkGetDeviceQueue)(VkDevice device, uint32_t queueFamilyIndex, uint32_t
      queueIndex, VkQueue * pQueue);
3557 typedef void (GLAD_API_PTR *PFN_vkGetDeviceQueue2)(VkDevice device, const VkDeviceQueueInfo2 *
      pQueueInfo, VkQueue * pQueue);
3558 typedef VkResult (GLAD_API_PTR *PFN_vkGetEventStatus)(VkDevice device, VkEvent event);
3559 typedef VkResult (GLAD_API_PTR *PFN_vkGetFenceStatus)(VkDevice device, VkFence fence);
3560 typedef void (GLAD_API_PTR *PFN_vkGetImageMemoryRequirements)(VkDevice device, VkImage image,
      VkMemoryRequirements * pMemoryRequirements);
3561 typedef void (GLAD_API_PTR *PFN_vkGetImageMemoryRequirements2)(VkDevice device, const
      VkImageMemoryRequirementsInfo2 * pInfo, VkMemoryRequirements2 * pMemoryRequirements);
3562 typedef void (GLAD_API_PTR *PFN_vkGetImageSparseMemoryRequirements)(VkDevice device, VkImage image,
      uint32_t * pSparseMemoryRequirementCount, VkSparseImageMemoryRequirements *
      pSparseMemoryRequirements);
3563 typedef void (GLAD_API_PTR *PFN_vkGetImageSparseMemoryRequirements2)(VkDevice device, const
      VkImageSparseMemoryRequirementsInfo2 * pInfo, uint32_t * pSparseMemoryRequirementCount,
      VkSparseImageMemoryRequirements2 * pSparseMemoryRequirements);
3564 typedef void (GLAD_API_PTR *PFN_vkGetImageSubresourceLayout)(VkDevice device, VkImage image, const
      VkImageSubresource * pSubresource, VkSubresourceLayout * pLayout);
3565 typedef PFN_vkVoidFunction (GLAD_API_PTR *PFN_vkGetInstanceProcAddr)(VkInstance instance, const char *
      pName);
3566 typedef void (GLAD_API_PTR *PFN_vkGetPhysicalDeviceExternalBufferProperties)(VkPhysicalDevice
      physicalDevice, const VkPhysicalDeviceExternalBufferInfo * pExternalBufferInfo,
      VkExternalBufferProperties * pExternalBufferProperties);
3567 typedef void (GLAD_API_PTR *PFN_vkGetPhysicalDeviceExternalFenceProperties)(VkPhysicalDevice
      physicalDevice, const VkPhysicalDeviceExternalFenceInfo * pExternalFenceInfo,
      VkExternalFenceProperties * pExternalFenceProperties);
3568 typedef void (GLAD_API_PTR *PFN_vkGetPhysicalDeviceExternalSemaphoreProperties)(VkPhysicalDevice
      physicalDevice, const VkPhysicalDeviceExternalSemaphoreInfo * pExternalSemaphoreInfo,
      VkExternalSemaphoreProperties * pExternalSemaphoreProperties);
3569 typedef void (GLAD_API_PTR *PFN_vkGetPhysicalDeviceFeatures)(VkPhysicalDevice physicalDevice,
      VkPhysicalDeviceFeatures * pFeatures);
3570 typedef void (GLAD_API_PTR *PFN_vkGetPhysicalDeviceFeatures2)(VkPhysicalDevice physicalDevice,
      VkPhysicalDeviceFeatures2 * pFeatures);
3571 typedef void (GLAD_API_PTR *PFN_vkGetPhysicalDeviceFormatProperties)(VkPhysicalDevice physicalDevice,
      VkFormat format, VkFormatProperties * pFormatProperties);
3572 typedef void (GLAD_API_PTR *PFN_vkGetPhysicalDeviceFormatProperties2)(VkPhysicalDevice physicalDevice,
      VkFormat format, VkFormatProperties2 * pFormatProperties);
3573 typedef VkResult (GLAD_API_PTR *PFN_vkGetPhysicalDeviceImageFormatProperties)(VkPhysicalDevice
      physicalDevice, VkFormat format, VkImageType type, VkImageTiling tiling, VkImageUsageFlags usage,
      VkImageCreateFlags flags, VkImageFormatProperties * pImageFormatProperties);
3574 typedef VkResult (GLAD_API_PTR *PFN_vkGetPhysicalDeviceImageFormatProperties2)(VkPhysicalDevice
      physicalDevice, const VkPhysicalDeviceImageFormatInfo2 * pImageFormatInfo, VkImageFormatProperties2 *
      pImageFormatProperties);
```

```
3575 typedef void (GLAD_API_PTR *PFN_vkGetPhysicalDeviceMemoryProperties)(VkPhysicalDevice physicalDevice,
       VkPhysicalDeviceMemoryProperties * pMemoryProperties);
3576 typedef void (GLAD_API_PTR *PFN_vkGetPhysicalDeviceMemoryProperties2)(VkPhysicalDevice physicalDevice,
       VkPhysicalDeviceMemoryProperties2 * pMemoryProperties);
3577 typedef VkResult (GLAD_API_PTR *PFN_vkGetPhysicalDevicePresentRectanglesKHR)(VkPhysicalDevice
       physicalDevice, VkSurfaceKHR surface, uint32_t * pRectCount, VkRect2D * pRects);
3578 typedef void (GLAD_API_PTR *PFN_vkGetPhysicalDeviceProperties)(VkPhysicalDevice physicalDevice,
       VkPhysicalDeviceProperties * pProperties);
3579 typedef void (GLAD_API_PTR *PFN_vkGetPhysicalDeviceProperties2)(VkPhysicalDevice physicalDevice,
       VkPhysicalDeviceProperties2 * pProperties);
3580 typedef void (GLAD_API_PTR *PFN_vkGetPhysicalDeviceQueueFamilyProperties)(VkPhysicalDevice
       physicalDevice, uint32_t * pQueueFamilyPropertyCount, VkQueueFamilyProperties *
       pQueueFamilyProperties);
3581 typedef void (GLAD_API_PTR *PFN_vkGetPhysicalDeviceQueueFamilyProperties2)(VkPhysicalDevice
       physicalDevice, uint32_t * pQueueFamilyPropertyCount, VkQueueFamilyProperties2 *
       pQueueFamilyProperties);
3582 typedef void (GLAD_API_PTR *PFN_vkGetPhysicalDeviceSparseImageFormatProperties)(VkPhysicalDevice
       physicalDevice, VkFormat format, VkImageType type, VkSampleCountFlagBits samples, VkImageUsageFlags
       usage, VkImageTiling tiling, uint32_t * pPropertyCount, VkSparseImageFormatProperties * pProperties);
3583 typedef void (GLAD_API_PTR *PFN_vkGetPhysicalDeviceSparseImageFormatProperties2)(VkPhysicalDevice
       physicalDevice, const VkPhysicalDeviceSparseImageFormatInfo2 * pFormatInfo, uint32_t *
       pPropertyCount, VkSparseImageFormatProperties2 * pProperties);
3584 typedef VkResult (GLAD_API_PTR *PFN_vkGetPhysicalDeviceSurfaceCapabilitiesKHR)(VkPhysicalDevice
       physicalDevice, VkSurfaceKHR surface, VkSurfaceCapabilitiesKHR * pSurfaceCapabilities);
3585 typedef VkResult (GLAD_API_PTR *PFN_vkGetPhysicalDeviceSurfaceFormatsKHR)(VkPhysicalDevice
       physicalDevice, VkSurfaceKHR surface, uint32_t * pSurfaceFormatCount, VkSurfaceFormatKHR *
       pSurfaceFormats);
3586 typedef VkResult (GLAD_API_PTR *PFN_vkGetPhysicalDeviceSurfacePresentModesKHR)(VkPhysicalDevice
       physicalDevice, VkSurfaceKHR surface, uint32_t * pPresentModeCount, VkPresentModeKHR *
       pPresentModes);
3587 typedef VkResult (GLAD_API_PTR *PFN_vkGetPhysicalDeviceSurfaceSupportKHR)(VkPhysicalDevice
       physicalDevice, uint32_t queueFamilyIndex, VkSurfaceKHR surface, VkBool32 * pSupported);
3588 typedef VkResult (GLAD_API_PTR *PFN_vkGetPipelineCacheData)(VkDevice device, VkPipelineCache
       pipelineCache, size_t * pDataSize, void * pData);
3589 typedef VkResult (GLAD_API_PTR *PFN_vkGetQueryPoolResults)(VkDevice device, VkQueryPool queryPool,
       uint32_t firstQuery, uint32_t queryCount, size_t dataSize, void * pData, VkDeviceSize stride,
       VkQueryResultFlags flags);
3590 typedef void (GLAD_API_PTR *PFN_vkGetRenderAreaGranularity)(VkDevice device, VkRenderPass renderPass,
       VkExtent2D * pGranularity);
3591 typedef VkResult (GLAD_API_PTR *PFN_vkGetSwapchainImagesKHR)(VkDevice device, VkSwapchainKHR swapchain,
       uint32_t * pSwapchainImageCount, VkImage * pSwapchainImages);
3592 typedef VkResult (GLAD_API_PTR *PFN_vkInvalidateMappedMemoryRanges)(VkDevice device, uint32_t
       memoryRangeCount, const VkMappedMemoryRange * pMemoryRanges);
3593 typedef VkResult (GLAD_API_PTR *PFN_vkMapMemory)(VkDevice device, VkDeviceMemory memory, VkDeviceSize
       offset, VkDeviceSize size, VkMemoryMapFlags flags, void ** ppData);
3594 typedef VkResult (GLAD_API_PTR *PFN_vkMergePipelineCaches)(VkDevice device, VkPipelineCache dstCache,
       uint32_t srcCacheCount, const VkPipelineCache * pSrcCaches);
3595 typedef VkResult (GLAD_API_PTR *PFN_vkQueueBindSparse)(VkQueue queue, uint32_t bindInfoCount, const
       VkBindSparseInfo * pBindInfo, VkFence fence);
3596 typedef VkResult (GLAD_API_PTR *PFN_vkQueuePresentKHR)(VkQueue queue, const VkPresentInfoKHR *
       pPresentInfo);
3597 typedef VkResult (GLAD_API_PTR *PFN_vkQueueSubmit)(VkQueue queue, uint32_t submitCount, const
       VkSubmitInfo * pSubmits, VkFence fence);
3598 typedef VkResult (GLAD_API_PTR *PFN_vkQueueWaitIdle)(VkQueue queue);
3599 typedef VkResult (GLAD_API_PTR *PFN_vkResetCommandBuffer)(VkCommandBuffer commandBuffer,
       VkCommandBufferResetFlags flags);
3600 typedef VkResult (GLAD_API_PTR *PFN_vkResetCommandPool)(VkDevice device, VkCommandPool commandPool,
       VkCommandPoolResetFlags flags);
3601 typedef VkResult (GLAD_API_PTR *PFN_vkResetDescriptorPool)(VkDevice device, VkDescriptorPool
       descriptorPool, VkDescriptorPoolResetFlags flags);
3602 typedef VkResult (GLAD_API_PTR *PFN_vkResetEvent)(VkDevice device, VkEvent event);
3603 typedef VkResult (GLAD_API_PTR *PFN_vkResetFences)(VkDevice device, uint32_t fenceCount, const VkFence
       * pFences);
3604 typedef VkResult (GLAD_API_PTR *PFN_vkSetEvent)(VkDevice device, VkEvent event);
3605 typedef void (GLAD_API_PTR *PFN_vkTrimCommandPool)(VkDevice device, VkCommandPool commandPool,
       VkCommandPoolTrimFlags flags);
3606 typedef void (GLAD_API_PTR *PFN_vkUnmapMemory)(VkDevice device, VkDeviceMemory memory);
3607 typedef void (GLAD_API_PTR *PFN_vkUpdateDescriptorSetWithTemplate)(VkDevice device, VkDescriptorSet
       descriptorSet, VkDescriptorUpdateTemplate descriptorUpdateTemplate, const void * pData);
3608 typedef void (GLAD_API_PTR *PFN_vkUpdateDescriptorSets)(VkDevice device, uint32_t descriptorWriteCount,
       const VkWriteDescriptorSet * pDescriptorWrites, uint32_t descriptorCopyCount, const
       VkCopyDescriptorSet * pDescriptorCopies);
3609 typedef VkResult (GLAD_API_PTR *PFN_vkWaitForFences)(VkDevice device, uint32_t fenceCount, const
       VkFence * pFences, VkBool32 waitAll, uint64_t timeout);
3610
3611 GLAD_API_CALL PFN_vkAcquireNextImage2KHR glad_vkAcquireNextImage2KHR;
3612 #define vkAcquireNextImage2KHR glad_vkAcquireNextImage2KHR
3613 GLAD_API_CALL PFN_vkAcquireNextImageKHR glad_vkAcquireNextImageKHR;
3614 #define vkAcquireNextImageKHR glad_vkAcquireNextImageKHR
3615 GLAD_API_CALL PFN_vkAllocateCommandBuffers glad_vkAllocateCommandBuffers;
3616 #define vkAllocateCommandBuffers glad_vkAllocateCommandBuffers
3617 GLAD_API_CALL PFN_vkAllocateDescriptorSets glad_vkAllocateDescriptorSets;
3618 #define vkAllocateDescriptorSets glad_vkAllocateDescriptorSets
3619 GLAD_API_CALL PFN_vkAllocateMemory glad_vkAllocateMemory;
3620 #define vkAllocateMemory glad_vkAllocateMemory
3621 GLAD_API_CALL PFN_vkBeginCommandBuffer glad_vkBeginCommandBuffer;
3622 #define vkBeginCommandBuffer glad_vkBeginCommandBuffer
```

```
3623 GLAD_API_CALL PFN_vkBindBufferMemory glad_vkBindBufferMemory;
3624 #define vkBindBufferMemory glad_vkBindBufferMemory
3625 GLAD_API_CALL PFN_vkBindBufferMemory2 glad_vkBindBufferMemory2;
3626 #define vkBindBufferMemory2 glad_vkBindBufferMemory2
3627 GLAD_API_CALL PFN_vkBindImageMemory glad_vkBindImageMemory;
3628 #define vkBindImageMemory glad_vkBindImageMemory
3629 GLAD_API_CALL PFN_vkBindImageMemory2 glad_vkBindImageMemory2;
3630 #define vkBindImageMemory2 glad_vkBindImageMemory2
3631 GLAD_API_CALL PFN_vkCmdBeginQuery glad_vkCmdBeginQuery;
3632 #define vkCmdBeginQuery glad_vkCmdBeginQuery
3633 GLAD_API_CALL PFN_vkCmdBeginRenderPass glad_vkCmdBeginRenderPass;
3634 #define vkCmdBeginRenderPass glad_vkCmdBeginRenderPass
3635 GLAD_API_CALL PFN_vkCmdBindDescriptorSets glad_vkCmdBindDescriptorSets;
3636 #define vkCmdBindDescriptorSets glad_vkCmdBindDescriptorSets
3637 GLAD_API_CALL PFN_vkCmdBindIndexBuffer glad_vkCmdBindIndexBuffer;
3638 #define vkCmdBindIndexBuffer glad_vkCmdBindIndexBuffer
3639 GLAD_API_CALL PFN_vkCmdBindPipeline glad_vkCmdBindPipeline;
3640 #define vkCmdBindPipeline glad_vkCmdBindPipeline
3641 GLAD_API_CALL PFN_vkCmdBindVertexBuffers glad_vkCmdBindVertexBuffers;
3642 #define vkCmdBindVertexBuffers glad_vkCmdBindVertexBuffers
3643 GLAD_API_CALL PFN_vkCmdBlitImage glad_vkCmdBlitImage;
3644 #define vkCmdBlitImage glad_vkCmdBlitImage
3645 GLAD_API_CALL PFN_vkCmdClearAttachments glad_vkCmdClearAttachments;
3646 #define vkCmdClearAttachments glad_vkCmdClearAttachments
3647 GLAD_API_CALL PFN_vkCmdClearColorImage glad_vkCmdClearColorImage;
3648 #define vkCmdClearColorImage glad_vkCmdClearColorImage
3649 GLAD_API_CALL PFN_vkCmdClearDepthStencilImage glad_vkCmdClearDepthStencilImage;
3650 #define vkCmdClearDepthStencilImage glad_vkCmdClearDepthStencilImage
3651 GLAD_API_CALL PFN_vkCmdCopyBuffer glad_vkCmdCopyBuffer;
3652 #define vkCmdCopyBuffer glad_vkCmdCopyBuffer
3653 GLAD_API_CALL PFN_vkCmdCopyBufferToImage glad_vkCmdCopyBufferToImage;
3654 #define vkCmdCopyBufferToImage glad_vkCmdCopyBufferToImage
3655 GLAD_API_CALL PFN_vkCmdCopyImage glad_vkCmdCopyImage;
3656 #define vkCmdCopyImage glad_vkCmdCopyImage
3657 GLAD_API_CALL PFN_vkCmdCopyImageToBuffer glad_vkCmdCopyImageToBuffer;
3658 #define vkCmdCopyImageToBuffer glad_vkCmdCopyImageToBuffer
3659 GLAD_API_CALL PFN_vkCmdCopyQueryPoolResults glad_vkCmdCopyQueryPoolResults;
3660 #define vkCmdCopyQueryPoolResults glad_vkCmdCopyQueryPoolResults
3661 GLAD_API_CALL PFN_vkCmdDispatch glad_vkCmdDispatch;
3662 #define vkCmdDispatch glad_vkCmdDispatch
3663 GLAD_API_CALL PFN_vkCmdDispatchBase glad_vkCmdDispatchBase;
3664 #define vkCmdDispatchBase glad_vkCmdDispatchBase
3665 GLAD_API_CALL PFN_vkCmdDispatchIndirect glad_vkCmdDispatchIndirect;
3666 #define vkCmdDispatchIndirect glad_vkCmdDispatchIndirect
3667 GLAD_API_CALL PFN_vkCmdDraw glad_vkCmdDraw;
3668 #define vkCmdDraw glad_vkCmdDraw
3669 GLAD_API_CALL PFN_vkCmdDrawIndexed glad_vkCmdDrawIndexed;
3670 #define vkCmdDrawIndexed glad_vkCmdDrawIndexed
3671 GLAD_API_CALL PFN_vkCmdDrawIndexedIndirect glad_vkCmdDrawIndexedIndirect;
3672 #define vkCmdDrawIndexedIndirect glad_vkCmdDrawIndexedIndirect
3673 GLAD_API_CALL PFN_vkCmdDrawIndirect glad_vkCmdDrawIndirect;
3674 #define vkCmdDrawIndirect glad_vkCmdDrawIndirect
3675 GLAD_API_CALL PFN_vkCmdEndQuery glad_vkCmdEndQuery;
3676 #define vkCmdEndQuery glad_vkCmdEndQuery
3677 GLAD_API_CALL PFN_vkCmdEndRenderPass glad_vkCmdEndRenderPass;
3678 #define vkCmdEndRenderPass glad_vkCmdEndRenderPass
3679 GLAD_API_CALL PFN_vkCmdExecuteCommands glad_vkCmdExecuteCommands;
3680 #define vkCmdExecuteCommands glad_vkCmdExecuteCommands
3681 GLAD_API_CALL PFN_vkCmdFillBuffer glad_vkCmdFillBuffer;
3682 #define vkCmdFillBuffer glad_vkCmdFillBuffer
3683 GLAD_API_CALL PFN_vkCmdNextSubpass glad_vkCmdNextSubpass;
3684 #define vkCmdNextSubpass glad_vkCmdNextSubpass
3685 GLAD_API_CALL PFN_vkCmdPipelineBarrier glad_vkCmdPipelineBarrier;
3686 #define vkCmdPipelineBarrier glad_vkCmdPipelineBarrier
3687 GLAD_API_CALL PFN_vkCmdPushConstants glad_vkCmdPushConstants;
3688 #define vkCmdPushConstants glad_vkCmdPushConstants
3689 GLAD_API_CALL PFN_vkCmdResetEvent glad_vkCmdResetEvent;
3690 #define vkCmdResetEvent glad_vkCmdResetEvent
3691 GLAD_API_CALL PFN_vkCmdResetQueryPool glad_vkCmdResetQueryPool;
3692 #define vkCmdResetQueryPool glad_vkCmdResetQueryPool
3693 GLAD_API_CALL PFN_vkCmdResolveImage glad_vkCmdResolveImage;
3694 #define vkCmdResolveImage glad_vkCmdResolveImage
3695 GLAD_API_CALL PFN_vkCmdSetBlendConstants glad_vkCmdSetBlendConstants;
3696 #define vkCmdSetBlendConstants glad_vkCmdSetBlendConstants
3697 GLAD_API_CALL PFN_vkCmdSetDepthBias glad_vkCmdSetDepthBias;
3698 #define vkCmdSetDepthBias glad_vkCmdSetDepthBias
3699 GLAD_API_CALL PFN_vkCmdSetDepthBounds glad_vkCmdSetDepthBounds;
3700 #define vkCmdSetDepthBounds glad_vkCmdSetDepthBounds
3701 GLAD_API_CALL PFN_vkCmdSetDeviceMask glad_vkCmdSetDeviceMask;
3702 #define vkCmdSetDeviceMask glad_vkCmdSetDeviceMask
3703 GLAD_API_CALL PFN_vkCmdSetEvent glad_vkCmdSetEvent;
3704 #define vkCmdSetEvent glad_vkCmdSetEvent
3705 GLAD_API_CALL PFN_vkCmdSetLineWidth glad_vkCmdSetLineWidth;
3706 #define vkCmdSetLineWidth glad_vkCmdSetLineWidth
3707 GLAD_API_CALL PFN_vkCmdSetScissor glad_vkCmdSetScissor;
3708 #define vkCmdSetScissor glad_vkCmdSetScissor
3709 GLAD_API_CALL PFN_vkCmdSetStencilCompareMask glad_vkCmdSetStencilCompareMask;
```

```
3710 #define vkCmdSetStencilCompareMask glad_vkCmdSetStencilCompareMask
3711 GLAD_API_CALL PFN_vkCmdSetStencilReference glad_vkCmdSetStencilReference;
3712 #define vkCmdSetStencilReference glad_vkCmdSetStencilReference
3713 GLAD_API_CALL PFN_vkCmdSetStencilWriteMask glad_vkCmdSetStencilWriteMask;
3714 #define vkCmdSetStencilWriteMask glad_vkCmdSetStencilWriteMask
3715 GLAD_API_CALL PFN_vkCmdSetViewport glad_vkCmdSetViewport;
3716 #define vkCmdSetViewport glad_vkCmdSetViewport
3717 GLAD_API_CALL PFN_vkCmdUpdateBuffer glad_vkCmdUpdateBuffer;
3718 #define vkCmdUpdateBuffer glad_vkCmdUpdateBuffer
3719 GLAD_API_CALL PFN_vkCmdWaitEvents glad_vkCmdWaitEvents;
3720 #define vkCmdWaitEvents glad_vkCmdWaitEvents
3721 GLAD_API_CALL PFN_vkCmdWriteTimestamp glad_vkCmdWriteTimestamp;
3722 #define vkCmdWriteTimestamp glad_vkCmdWriteTimestamp
3723 GLAD_API_CALL PFN_vkCreateBuffer glad_vkCreateBuffer;
3724 #define vkCreateBuffer glad_vkCreateBuffer
3725 GLAD_API_CALL PFN_vkCreateBufferView glad_vkCreateBufferView;
3726 #define vkCreateBufferView glad_vkCreateBufferView
3727 GLAD_API_CALL PFN_vkCreateCommandPool glad_vkCreateCommandPool;
3728 #define vkCreateCommandPool glad_vkCreateCommandPool
3729 GLAD_API_CALL PFN_vkCreateComputePipelines glad_vkCreateComputePipelines;
3730 #define vkCreateComputePipelines glad_vkCreateComputePipelines
3731 GLAD_API_CALL PFN_vkCreateDebugReportCallbackEXT glad_vkCreateDebugReportCallbackEXT;
3732 #define vkCreateDebugReportCallbackEXT glad_vkCreateDebugReportCallbackEXT
3733 GLAD_API_CALL PFN_vkCreateDescriptorPool glad_vkCreateDescriptorPool;
3734 #define vkCreateDescriptorPool glad_vkCreateDescriptorPool
3735 GLAD_API_CALL PFN_vkCreateDescriptorSetLayout glad_vkCreateDescriptorSetLayout;
3736 #define vkCreateDescriptorSetLayout glad_vkCreateDescriptorSetLayout
3737 GLAD_API_CALL PFN_vkCreateDescriptorUpdateTemplate glad_vkCreateDescriptorUpdateTemplate;
3738 #define vkCreateDescriptorUpdateTemplate glad_vkCreateDescriptorUpdateTemplate
3739 GLAD_API_CALL PFN_vkCreateDevice glad_vkCreateDevice;
3740 #define vkCreateDevice glad_vkCreateDevice
3741 GLAD_API_CALL PFN_vkCreateEvent glad_vkCreateEvent;
3742 #define vkCreateEvent glad_vkCreateEvent
3743 GLAD_API_CALL PFN_vkCreateFence glad_vkCreateFence;
3744 #define vkCreateFence glad_vkCreateFence
3745 GLAD_API_CALL PFN_vkCreateFramebuffer glad_vkCreateFramebuffer;
3746 #define vkCreateFramebuffer glad_vkCreateFramebuffer
3747 GLAD_API_CALL PFN_vkCreateGraphicsPipelines glad_vkCreateGraphicsPipelines;
3748 #define vkCreateGraphicsPipelines glad_vkCreateGraphicsPipelines
3749 GLAD_API_CALL PFN_vkCreateImage glad_vkCreateImage;
3750 #define vkCreateImage glad_vkCreateImage
3751 GLAD_API_CALL PFN_vkCreateImageView glad_vkCreateImageView;
3752 #define vkCreateImageView glad_vkCreateImageView
3753 GLAD_API_CALL PFN_vkCreateInstance glad_vkCreateInstance;
3754 #define vkCreateInstance glad_vkCreateInstance
3755 GLAD_API_CALL PFN_vkCreatePipelineCache glad_vkCreatePipelineCache;
3756 #define vkCreatePipelineCache glad_vkCreatePipelineCache
3757 GLAD_API_CALL PFN_vkCreatePipelineLayout glad_vkCreatePipelineLayout;
3758 #define vkCreatePipelineLayout glad_vkCreatePipelineLayout
3759 GLAD_API_CALL PFN_vkCreateQueryPool glad_vkCreateQueryPool;
3760 #define vkCreateQueryPool glad_vkCreateQueryPool
3761 GLAD_API_CALL PFN_vkCreateRenderPass glad_vkCreateRenderPass;
3762 #define vkCreateRenderPass glad_vkCreateRenderPass
3763 GLAD_API_CALL PFN_vkCreateSampler glad_vkCreateSampler;
3764 #define vkCreateSampler glad_vkCreateSampler
3765 GLAD_API_CALL PFN_vkCreateSamplerYcbcrConversion glad_vkCreateSamplerYcbcrConversion;
3766 #define vkCreateSamplerYcbcrConversion glad_vkCreateSamplerYcbcrConversion
3767 GLAD_API_CALL PFN_vkCreateSemaphore glad_vkCreateSemaphore;
3768 #define vkCreateSemaphore glad_vkCreateSemaphore
3769 GLAD_API_CALL PFN_vkCreateShaderModule glad_vkCreateShaderModule;
3770 #define vkCreateShaderModule glad_vkCreateShaderModule
3771 GLAD_API_CALL PFN_vkCreateSwapchainKHR glad_vkCreateSwapchainKHR;
3772 #define vkCreateSwapchainKHR glad_vkCreateSwapchainKHR
3773 GLAD_API_CALL PFN_vkDebugReportMessageEXT glad_vkDebugReportMessageEXT;
3774 #define vkDebugReportMessageEXT glad_vkDebugReportMessageEXT
3775 GLAD_API_CALL PFN_vkDestroyBuffer glad_vkDestroyBuffer;
3776 #define vkDestroyBuffer glad_vkDestroyBuffer
3777 GLAD_API_CALL PFN_vkDestroyBufferView glad_vkDestroyBufferView;
3778 #define vkDestroyBufferView glad_vkDestroyBufferView
3779 GLAD_API_CALL PFN_vkDestroyCommandPool glad_vkDestroyCommandPool;
3780 #define vkDestroyCommandPool glad_vkDestroyCommandPool
3781 GLAD_API_CALL PFN_vkDestroyDebugReportCallbackEXT glad_vkDestroyDebugReportCallbackEXT;
3782 #define vkDestroyDebugReportCallbackEXT glad_vkDestroyDebugReportCallbackEXT
3783 GLAD_API_CALL PFN_vkDestroyDescriptorPool glad_vkDestroyDescriptorPool;
3784 #define vkDestroyDescriptorPool glad_vkDestroyDescriptorPool
3785 GLAD_API_CALL PFN_vkDestroyDescriptorSetLayout glad_vkDestroyDescriptorSetLayout;
3786 #define vkDestroyDescriptorSetLayout glad_vkDestroyDescriptorSetLayout
3787 GLAD_API_CALL PFN_vkDestroyDescriptorUpdateTemplate glad_vkDestroyDescriptorUpdateTemplate;
3788 #define vkDestroyDescriptorUpdateTemplate glad_vkDestroyDescriptorUpdateTemplate
3789 GLAD_API_CALL PFN_vkDestroyDevice glad_vkDestroyDevice;
3790 #define vkDestroyDevice glad_vkDestroyDevice
3791 GLAD_API_CALL PFN_vkDestroyEvent glad_vkDestroyEvent;
3792 #define vkDestroyEvent glad_vkDestroyEvent
3793 GLAD_API_CALL PFN_vkDestroyFence glad_vkDestroyFence;
3794 #define vkDestroyFence glad_vkDestroyFence
3795 GLAD_API_CALL PFN_vkDestroyFramebuffer glad_vkDestroyFramebuffer;
3796 #define vkDestroyFramebuffer glad_vkDestroyFramebuffer
```

```
3797 GLAD_API_CALL PFN_vkDestroyImage glad_vkDestroyImage;
3798 #define vkDestroyImage glad_vkDestroyImage
3799 GLAD_API_CALL PFN_vkDestroyImageView glad_vkDestroyImageView;
3800 #define vkDestroyImageView glad_vkDestroyImageView
3801 GLAD_API_CALL PFN_vkDestroyInstance glad_vkDestroyInstance;
3802 #define vkDestroyInstance glad_vkDestroyInstance
3803 GLAD_API_CALL PFN_vkDestroyPipeline glad_vkDestroyPipeline;
3804 #define vkDestroyPipeline glad_vkDestroyPipeline
3805 GLAD_API_CALL PFN_vkDestroyPipelineCache glad_vkDestroyPipelineCache;
3806 #define vkDestroyPipelineCache glad_vkDestroyPipelineCache
3807 GLAD_API_CALL PFN_vkDestroyPipelineLayout glad_vkDestroyPipelineLayout;
3808 #define vkDestroyPipelineLayout glad_vkDestroyPipelineLayout
3809 GLAD_API_CALL PFN_vkDestroyQueryPool glad_vkDestroyQueryPool;
3810 #define vkDestroyQueryPool glad_vkDestroyQueryPool
3811 GLAD_API_CALL PFN_vkDestroyRenderPass glad_vkDestroyRenderPass;
3812 #define vkDestroyRenderPass glad_vkDestroyRenderPass
3813 GLAD_API_CALL PFN_vkDestroySampler glad_vkDestroySampler;
3814 #define vkDestroySampler glad_vkDestroySampler
3815 GLAD_API_CALL PFN_vkDestroySamplerYcbcrConversion glad_vkDestroySamplerYcbcrConversion;
3816 #define vkDestroySamplerYcbcrConversion glad_vkDestroySamplerYcbcrConversion
3817 GLAD_API_CALL PFN_vkDestroySemaphore glad_vkDestroySemaphore;
3818 #define vkDestroySemaphore glad_vkDestroySemaphore
3819 GLAD_API_CALL PFN_vkDestroyShaderModule glad_vkDestroyShaderModule;
3820 #define vkDestroyShaderModule glad_vkDestroyShaderModule
3821 GLAD_API_CALL PFN_vkDestroySurfaceKHR glad_vkDestroySurfaceKHR;
3822 #define vkDestroySurfaceKHR glad_vkDestroySurfaceKHR
3823 GLAD_API_CALL PFN_vkDestroySwapchainKHR glad_vkDestroySwapchainKHR;
3824 #define vkDestroySwapchainKHR glad_vkDestroySwapchainKHR
3825 GLAD_API_CALL PFN_vkDeviceWaitIdle glad_vkDeviceWaitIdle;
3826 #define vkDeviceWaitIdle glad_vkDeviceWaitIdle
3827 GLAD_API_CALL PFN_vkEndCommandBuffer glad_vkEndCommandBuffer;
3828 #define vkEndCommandBuffer glad_vkEndCommandBuffer
3829 GLAD_API_CALL PFN_vkEnumerateDeviceExtensionProperties glad_vkEnumerateDeviceExtensionProperties;
3830 #define vkEnumerateDeviceExtensionProperties glad_vkEnumerateDeviceExtensionProperties
3831 GLAD_API_CALL PFN_vkEnumerateDeviceLayerProperties glad_vkEnumerateDeviceLayerProperties;
3832 #define vkEnumerateDeviceLayerProperties glad_vkEnumerateDeviceLayerProperties
3833 GLAD_API_CALL PFN_vkEnumerateInstanceExtensionProperties glad_vkEnumerateInstanceExtensionProperties;
3834 #define vkEnumerateInstanceExtensionProperties glad_vkEnumerateInstanceExtensionProperties
3835 GLAD_API_CALL PFN_vkEnumerateInstanceLayerProperties glad_vkEnumerateInstanceLayerProperties;
3836 #define vkEnumerateInstanceLayerProperties glad_vkEnumerateInstanceLayerProperties
3837 GLAD_API_CALL PFN_vkEnumerateInstanceVersion glad_vkEnumerateInstanceVersion;
3838 #define vkEnumerateInstanceVersion glad_vkEnumerateInstanceVersion
3839 GLAD_API_CALL PFN_vkEnumeratePhysicalDeviceGroups glad_vkEnumeratePhysicalDeviceGroups;
3840 #define vkEnumeratePhysicalDeviceGroups glad_vkEnumeratePhysicalDeviceGroups
3841 GLAD_API_CALL PFN_vkEnumeratePhysicalDevices glad_vkEnumeratePhysicalDevices;
3842 #define vkEnumeratePhysicalDevices glad_vkEnumeratePhysicalDevices
3843 GLAD_API_CALL PFN_vkFlushMappedMemoryRanges glad_vkFlushMappedMemoryRanges;
3844 #define vkFlushMappedMemoryRanges glad_vkFlushMappedMemoryRanges
3845 GLAD_API_CALL PFN_vkFreeCommandBuffers glad_vkFreeCommandBuffers;
3846 #define vkFreeCommandBuffers glad_vkFreeCommandBuffers
3847 GLAD_API_CALL PFN_vkFreeDescriptorSets glad_vkFreeDescriptorSets;
3848 #define vkFreeDescriptorSets glad_vkFreeDescriptorSets
3849 GLAD_API_CALL PFN_vkFreeMemory glad_vkFreeMemory;
3850 #define vkFreeMemory glad_vkFreeMemory
3851 GLAD_API_CALL PFN_vkGetBufferMemoryRequirements glad_vkGetBufferMemoryRequirements;
3852 #define vkGetBufferMemoryRequirements glad_vkGetBufferMemoryRequirements
3853 GLAD_API_CALL PFN_vkGetBufferMemoryRequirements2 glad_vkGetBufferMemoryRequirements2;
3854 #define vkGetBufferMemoryRequirements2 glad_vkGetBufferMemoryRequirements2
3855 GLAD_API_CALL PFN_vkGetDescriptorSetLayoutSupport glad_vkGetDescriptorSetLayoutSupport;
3856 #define vkGetDescriptorSetLayoutSupport glad_vkGetDescriptorSetLayoutSupport
3857 GLAD_API_CALL PFN_vkGetDeviceGroupPeerMemoryFeatures glad_vkGetDeviceGroupPeerMemoryFeatures;
3858 #define vkGetDeviceGroupPeerMemoryFeatures glad_vkGetDeviceGroupPeerMemoryFeatures
3859 GLAD_API_CALL PFN_vkGetDeviceGroupPresentCapabilitiesKHR glad_vkGetDeviceGroupPresentCapabilitiesKHR;
3860 #define vkGetDeviceGroupPresentCapabilitiesKHR glad_vkGetDeviceGroupPresentCapabilitiesKHR
3861 GLAD_API_CALL PFN_vkGetDeviceGroupSurfacePresentModesKHR glad_vkGetDeviceGroupSurfacePresentModesKHR;
3862 #define vkGetDeviceGroupSurfacePresentModesKHR glad_vkGetDeviceGroupSurfacePresentModesKHR
3863 GLAD_API_CALL PFN_vkGetDeviceMemoryCommitment glad_vkGetDeviceMemoryCommitment;
3864 #define vkGetDeviceMemoryCommitment glad_vkGetDeviceMemoryCommitment
3865 GLAD_API_CALL PFN_vkGetDeviceProcAddr glad_vkGetDeviceProcAddr;
3866 #define vkGetDeviceProcAddr glad_vkGetDeviceProcAddr
3867 GLAD_API_CALL PFN_vkGetDeviceQueue glad_vkGetDeviceQueue;
3868 #define vkGetDeviceQueue glad_vkGetDeviceQueue
3869 GLAD_API_CALL PFN_vkGetDeviceQueue2 glad_vkGetDeviceQueue2;
3870 #define vkGetDeviceQueue2 glad_vkGetDeviceQueue2
3871 GLAD_API_CALL PFN_vkGetEventStatus glad_vkGetEventStatus;
3872 #define vkGetEventStatus glad_vkGetEventStatus
3873 GLAD_API_CALL PFN_vkGetFenceStatus glad_vkGetFenceStatus;
3874 #define vkGetFenceStatus glad_vkGetFenceStatus
3875 GLAD_API_CALL PFN_vkGetImageMemoryRequirements glad_vkGetImageMemoryRequirements;
3876 #define vkGetImageMemoryRequirements glad_vkGetImageMemoryRequirements
3877 GLAD_API_CALL PFN_vkGetImageMemoryRequirements2 glad_vkGetImageMemoryRequirements2;
3878 #define vkGetImageMemoryRequirements2 glad_vkGetImageMemoryRequirements2
3879 GLAD_API_CALL PFN_vkGetImageSparseMemoryRequirements glad_vkGetImageSparseMemoryRequirements;
3880 #define vkGetImageSparseMemoryRequirements glad_vkGetImageSparseMemoryRequirements
3881 GLAD_API_CALL PFN_vkGetImageSparseMemoryRequirements2 glad_vkGetImageSparseMemoryRequirements2;
3882 #define vkGetImageSparseMemoryRequirements2 glad_vkGetImageSparseMemoryRequirements2
3883 GLAD_API_CALL PFN_vkGetImageSubresourceLayout glad_vkGetImageSubresourceLayout;
```

```
3884 #define vkGetImageSubresourceLayout glad_vkGetImageSubresourceLayout
3885 GLAD_API_CALL PFN_vkGetInstanceProcAddr glad_vkGetInstanceProcAddr;
3886 #define vkGetInstanceProcAddr glad_vkGetInstanceProcAddr
3887 GLAD_API_CALL PFN_vkGetPhysicalDeviceExternalBufferProperties
       glad_vkGetPhysicalDeviceExternalBufferProperties;
3888 #define vkGetPhysicalDeviceExternalBufferProperties glad_vkGetPhysicalDeviceExternalBufferProperties
3889 GLAD_API_CALL PFN_vkGetPhysicalDeviceExternalFenceProperties
       glad_vkGetPhysicalDeviceExternalFenceProperties;
3890 #define vkGetPhysicalDeviceExternalFenceProperties glad_vkGetPhysicalDeviceExternalFenceProperties
3891 GLAD_API_CALL PFN_vkGetPhysicalDeviceExternalSemaphoreProperties
       glad_vkGetPhysicalDeviceExternalSemaphoreProperties;
3892 #define vkGetPhysicalDeviceExternalSemaphoreProperties
       glad_vkGetPhysicalDeviceExternalSemaphoreProperties
3893 GLAD_API_CALL PFN_vkGetPhysicalDeviceFeatures glad_vkGetPhysicalDeviceFeatures;
3894 #define vkGetPhysicalDeviceFeatures glad_vkGetPhysicalDeviceFeatures
3895 GLAD_API_CALL PFN_vkGetPhysicalDeviceFeatures2 glad_vkGetPhysicalDeviceFeatures2;
3896 #define vkGetPhysicalDeviceFeatures2 glad_vkGetPhysicalDeviceFeatures2
3897 GLAD_API_CALL PFN_vkGetPhysicalDeviceFormatProperties glad_vkGetPhysicalDeviceFormatProperties;
3898 #define vkGetPhysicalDeviceFormatProperties glad_vkGetPhysicalDeviceFormatProperties
3899 GLAD_API_CALL PFN_vkGetPhysicalDeviceFormatProperties2 glad_vkGetPhysicalDeviceFormatProperties2;
3900 #define vkGetPhysicalDeviceFormatProperties2 glad_vkGetPhysicalDeviceFormatProperties2
3901 GLAD_API_CALL PFN_vkGetPhysicalDeviceImageFormatProperties
       glad_vkGetPhysicalDeviceImageFormatProperties;
3902 #define vkGetPhysicalDeviceImageFormatProperties glad_vkGetPhysicalDeviceImageFormatProperties
3903 GLAD_API_CALL PFN_vkGetPhysicalDeviceImageFormatProperties2
       glad_vkGetPhysicalDeviceImageFormatProperties2;
3904 #define vkGetPhysicalDeviceImageFormatProperties2 glad_vkGetPhysicalDeviceImageFormatProperties2
3905 GLAD_API_CALL PFN_vkGetPhysicalDeviceMemoryProperties glad_vkGetPhysicalDeviceMemoryProperties;
3906 #define vkGetPhysicalDeviceMemoryProperties glad_vkGetPhysicalDeviceMemoryProperties
3907 GLAD_API_CALL PFN_vkGetPhysicalDeviceMemoryProperties2 glad_vkGetPhysicalDeviceMemoryProperties2;
3908 #define vkGetPhysicalDeviceMemoryProperties2 glad_vkGetPhysicalDeviceMemoryProperties2
3909 GLAD_API_CALL PFN_vkGetPhysicalDevicePresentRectanglesKHR glad_vkGetPhysicalDevicePresentRectanglesKHR;
3910 #define vkGetPhysicalDevicePresentRectanglesKHR glad_vkGetPhysicalDevicePresentRectanglesKHR
3911 GLAD_API_CALL PFN_vkGetPhysicalDeviceProperties glad_vkGetPhysicalDeviceProperties;
3912 #define vkGetPhysicalDeviceProperties glad_vkGetPhysicalDeviceProperties
3913 GLAD_API_CALL PFN_vkGetPhysicalDeviceProperties2 glad_vkGetPhysicalDeviceProperties2;
3914 #define vkGetPhysicalDeviceProperties2 glad_vkGetPhysicalDeviceProperties2
3915 GLAD_API_CALL PFN_vkGetPhysicalDeviceQueueFamilyProperties
       glad_vkGetPhysicalDeviceQueueFamilyProperties;
3916 #define vkGetPhysicalDeviceQueueFamilyProperties glad_vkGetPhysicalDeviceQueueFamilyProperties
3917 GLAD_API_CALL PFN_vkGetPhysicalDeviceQueueFamilyProperties2
       glad_vkGetPhysicalDeviceQueueFamilyProperties2;
3918 #define vkGetPhysicalDeviceQueueFamilyProperties2 glad_vkGetPhysicalDeviceQueueFamilyProperties2
3919 GLAD_API_CALL PFN_vkGetPhysicalDeviceSparseImageFormatProperties
       glad_vkGetPhysicalDeviceSparseImageFormatProperties;
3920 #define vkGetPhysicalDeviceSparseImageFormatProperties
       glad_vkGetPhysicalDeviceSparseImageFormatProperties
3921 GLAD_API_CALL PFN_vkGetPhysicalDeviceSparseImageFormatProperties2
       glad_vkGetPhysicalDeviceSparseImageFormatProperties2;
3922 #define vkGetPhysicalDeviceSparseImageFormatProperties2
       glad_vkGetPhysicalDeviceSparseImageFormatProperties2
3923 GLAD_API_CALL PFN_vkGetPhysicalDeviceSurfaceCapabilitiesKHR
       glad_vkGetPhysicalDeviceSurfaceCapabilitiesKHR;
3924 #define vkGetPhysicalDeviceSurfaceCapabilitiesKHR glad_vkGetPhysicalDeviceSurfaceCapabilitiesKHR
3925 GLAD_API_CALL PFN_vkGetPhysicalDeviceSurfaceFormatsKHR glad_vkGetPhysicalDeviceSurfaceFormatsKHR;
3926 #define vkGetPhysicalDeviceSurfaceFormatsKHR glad_vkGetPhysicalDeviceSurfaceFormatsKHR
3927 GLAD_API_CALL PFN_vkGetPhysicalDeviceSurfacePresentModesKHR
       glad_vkGetPhysicalDeviceSurfacePresentModesKHR;
3928 #define vkGetPhysicalDeviceSurfacePresentModesKHR glad_vkGetPhysicalDeviceSurfacePresentModesKHR
3929 GLAD_API_CALL PFN_vkGetPhysicalDeviceSurfaceSupportKHR glad_vkGetPhysicalDeviceSurfaceSupportKHR;
3930 #define vkGetPhysicalDeviceSurfaceSupportKHR glad_vkGetPhysicalDeviceSurfaceSupportKHR
3931 GLAD_API_CALL PFN_vkGetPipelineCacheData glad_vkGetPipelineCacheData;
3932 #define vkGetPipelineCacheData glad_vkGetPipelineCacheData
3933 GLAD_API_CALL PFN_vkGetQueryPoolResults glad_vkGetQueryPoolResults;
3934 #define vkGetQueryPoolResults glad_vkGetQueryPoolResults
3935 GLAD_API_CALL PFN_vkGetRenderAreaGranularity glad_vkGetRenderAreaGranularity;
3936 #define vkGetRenderAreaGranularity glad_vkGetRenderAreaGranularity
3937 GLAD_API_CALL PFN_vkGetSwapchainImagesKHR glad_vkGetSwapchainImagesKHR;
3938 #define vkGetSwapchainImagesKHR glad_vkGetSwapchainImagesKHR
3939 GLAD_API_CALL PFN_vkInvalidateMappedMemoryRanges glad_vkInvalidateMappedMemoryRanges;
3940 #define vkInvalidateMappedMemoryRanges glad_vkInvalidateMappedMemoryRanges
3941 GLAD_API_CALL PFN_vkMapMemory glad_vkMapMemory;
3942 #define vkMapMemory glad_vkMapMemory
3943 GLAD_API_CALL PFN_vkMergePipelineCaches glad_vkMergePipelineCaches;
3944 #define vkMergePipelineCaches glad_vkMergePipelineCaches
3945 GLAD_API_CALL PFN_vkQueueBindSparse glad_vkQueueBindSparse;
3946 #define vkQueueBindSparse glad_vkQueueBindSparse
3947 GLAD_API_CALL PFN_vkQueuePresentKHR glad_vkQueuePresentKHR;
3948 #define vkQueuePresentKHR glad_vkQueuePresentKHR
3949 GLAD_API_CALL PFN_vkQueueSubmit glad_vkQueueSubmit;
3950 #define vkQueueSubmit glad_vkQueueSubmit
3951 GLAD_API_CALL PFN_vkQueueWaitIdle glad_vkQueueWaitIdle;
3952 #define vkQueueWaitIdle glad_vkQueueWaitIdle
3953 GLAD_API_CALL PFN_vkResetCommandBuffer glad_vkResetCommandBuffer;
3954 #define vkResetCommandBuffer glad_vkResetCommandBuffer
3955 GLAD_API_CALL PFN_vkResetCommandPool glad_vkResetCommandPool;
3956 #define vkResetCommandPool glad_vkResetCommandPool
```

```
3957 GLAD_API_CALL PFN_vkResetDescriptorPool glad_vkResetDescriptorPool;
3958 #define vkResetDescriptorPool glad_vkResetDescriptorPool
3959 GLAD_API_CALL PFN_vkResetEvent glad_vkResetEvent;
3960 #define vkResetEvent glad_vkResetEvent
3961 GLAD_API_CALL PFN_vkResetFences glad_vkResetFences;
3962 #define vkResetFences glad_vkResetFences
3963 GLAD_API_CALL PFN_vkSetEvent glad_vkSetEvent;
3964 #define vkSetEvent glad_vkSetEvent
3965 GLAD_API_CALL PFN_vkTrimCommandPool glad_vkTrimCommandPool;
3966 #define vkTrimCommandPool glad_vkTrimCommandPool
3967 GLAD_API_CALL PFN_vkUnmapMemory glad_vkUnmapMemory;
3968 #define vkUnmapMemory glad_vkUnmapMemory
3969 GLAD_API_CALL PFN_vkUpdateDescriptorSetWithTemplate glad_vkUpdateDescriptorSetWithTemplate;
3970 #define vkUpdateDescriptorSetWithTemplate glad_vkUpdateDescriptorSetWithTemplate
3971 GLAD_API_CALL PFN_vkUpdateDescriptorSets glad_vkUpdateDescriptorSets;
3972 #define vkUpdateDescriptorSets glad_vkUpdateDescriptorSets
3973 GLAD_API_CALL PFN_vkWaitForFences glad_vkWaitForFences;
3974 #define vkWaitForFences glad_vkWaitForFences
3975
3976
3977
3978
3979
3980 GLAD_API_CALL int gladLoadVulkanUserPtr( VkPhysicalDevice physical_device, GLADuserptrloadfunc load,
       void *userptr);
3981 GLAD_API_CALL int gladLoadVulkan( VkPhysicalDevice physical_device, GLADloadfunc load);
3982
3983
3984
3985 #ifdef __cplusplus
3986 }
3987 #endif
3988 #endif
3989
3990 /* Source */
3991 #ifdef GLAD_VULKAN_IMPLEMENTATION
3992 #include <stdio.h>
3993 #include <stdlib.h>
3994 #include <string.h>
3995
3996 #ifndef GLAD_IMPL_UTIL_C_
3997 #define GLAD_IMPL_UTIL_C_
3998
3999 #ifdef _MSC_VER
4000 #define GLAD_IMPL_UTIL_SSCANF sscanf_s
4001 #else
4002 #define GLAD_IMPL_UTIL_SSCANF sscanf
4003 #endif
4004
4005 #endif /* GLAD_IMPL_UTIL_C_ */
4006
4007 #ifdef __cplusplus
4008 extern "C" {
4009 #endif
4010
4011
4012
4013 int GLAD_VK_VERSION_1_0 = 0;
4014 int GLAD_VK_VERSION_1_1 = 0;
4015 int GLAD_VK_EXT_debug_report = 0;
4016 int GLAD_VK_KHR_surface = 0;
4017 int GLAD_VK_KHR_swapchain = 0;
4018
4019
4020
4021 PFN_vkAcquireNextImage2KHR glad_vkAcquireNextImage2KHR = NULL;
4022 PFN_vkAcquireNextImageKHR glad_vkAcquireNextImageKHR = NULL;
4023 PFN_vkAllocateCommandBuffers glad_vkAllocateCommandBuffers = NULL;
4024 PFN_vkAllocateDescriptorSets glad_vkAllocateDescriptorSets = NULL;
4025 PFN_vkAllocateMemory glad_vkAllocateMemory = NULL;
4026 PFN_vkBeginCommandBuffer glad_vkBeginCommandBuffer = NULL;
4027 PFN_vkBindBufferMemory glad_vkBindBufferMemory = NULL;
4028 PFN_vkBindBufferMemory2 glad_vkBindBufferMemory2 = NULL;
4029 PFN_vkBindImageMemory glad_vkBindImageMemory = NULL;
4030 PFN_vkBindImageMemory2 glad_vkBindImageMemory2 = NULL;
4031 PFN_vkCmdBeginQuery glad_vkCmdBeginQuery = NULL;
4032 PFN_vkCmdBeginRenderPass glad_vkCmdBeginRenderPass = NULL;
4033 PFN_vkCmdBindDescriptorSets glad_vkCmdBindDescriptorSets = NULL;
4034 PFN_vkCmdBindIndexBuffer glad_vkCmdBindIndexBuffer = NULL;
4035 PFN_vkCmdBindPipeline glad_vkCmdBindPipeline = NULL;
4036 PFN_vkCmdBindVertexBuffers glad_vkCmdBindVertexBuffers = NULL;
4037 PFN_vkCmdBlitImage glad_vkCmdBlitImage = NULL;
4038 PFN_vkCmdClearAttachments glad_vkCmdClearAttachments = NULL;
4039 PFN_vkCmdClearColorImage glad_vkCmdClearColorImage = NULL;
4040 PFN_vkCmdClearDepthStencilImage glad_vkCmdClearDepthStencilImage = NULL;
4041 PFN_vkCmdCopyBuffer glad_vkCmdCopyBuffer = NULL;
4042 PFN_vkCmdCopyBufferToImage glad_vkCmdCopyBufferToImage = NULL;
```

```
4043 PFN_vkCmdCopyImage glad_vkCmdCopyImage = NULL;
4044 PFN_vkCmdCopyImageToBuffer glad_vkCmdCopyImageToBuffer = NULL;
4045 PFN_vkCmdCopyQueryPoolResults glad_vkCmdCopyQueryPoolResults = NULL;
4046 PFN_vkCmdDispatch glad_vkCmdDispatch = NULL;
4047 PFN_vkCmdDispatchBase glad_vkCmdDispatchBase = NULL;
4048 PFN_vkCmdDispatchIndirect glad_vkCmdDispatchIndirect = NULL;
4049 PFN_vkCmdDraw glad_vkCmdDraw = NULL;
4050 PFN_vkCmdDrawIndexed glad_vkCmdDrawIndexed = NULL;
4051 PFN_vkCmdDrawIndexedIndirect glad_vkCmdDrawIndexedIndirect = NULL;
4052 PFN_vkCmdDrawIndirect glad_vkCmdDrawIndirect = NULL;
4053 PFN_vkCmdEndQuery glad_vkCmdEndQuery = NULL;
4054 PFN_vkCmdEndRenderPass glad_vkCmdEndRenderPass = NULL;
4055 PFN_vkCmdExecuteCommands glad_vkCmdExecuteCommands = NULL;
4056 PFN_vkCmdFillBuffer glad_vkCmdFillBuffer = NULL;
4057 PFN_vkCmdNextSubpass glad_vkCmdNextSubpass = NULL;
4058 PFN_vkCmdPipelineBarrier glad_vkCmdPipelineBarrier = NULL;
4059 PFN_vkCmdPushConstants glad_vkCmdPushConstants = NULL;
4060 PFN_vkCmdResetEvent glad_vkCmdResetEvent = NULL;
4061 PFN_vkCmdResetQueryPool glad_vkCmdResetQueryPool = NULL;
4062 PFN_vkCmdResolveImage glad_vkCmdResolveImage = NULL;
4063 PFN_vkCmdSetBlendConstants glad_vkCmdSetBlendConstants = NULL;
4064 PFN_vkCmdSetDepthBias glad_vkCmdSetDepthBias = NULL;
4065 PFN_vkCmdSetDepthBounds glad_vkCmdSetDepthBounds = NULL;
4066 PFN_vkCmdSetDeviceMask glad_vkCmdSetDeviceMask = NULL;
4067 PFN_vkCmdSetEvent glad_vkCmdSetEvent = NULL;
4068 PFN_vkCmdSetLineWidth glad_vkCmdSetLineWidth = NULL;
4069 PFN_vkCmdSetScissor glad_vkCmdSetScissor = NULL;
4070 PFN_vkCmdSetStencilCompareMask glad_vkCmdSetStencilCompareMask = NULL;
4071 PFN_vkCmdSetStencilReference glad_vkCmdSetStencilReference = NULL;
4072 PFN_vkCmdSetStencilWriteMask glad_vkCmdSetStencilWriteMask = NULL;
4073 PFN_vkCmdSetViewport glad_vkCmdSetViewport = NULL;
4074 PFN_vkCmdUpdateBuffer glad_vkCmdUpdateBuffer = NULL;
4075 PFN_vkCmdWaitEvents glad_vkCmdWaitEvents = NULL;
4076 PFN_vkCmdWriteTimestamp glad_vkCmdWriteTimestamp = NULL;
4077 PFN_vkCreateBuffer glad_vkCreateBuffer = NULL;
4078 PFN_vkCreateBufferView glad_vkCreateBufferView = NULL;
4079 PFN_vkCreateCommandPool glad_vkCreateCommandPool = NULL;
4080 PFN_vkCreateComputePipelines glad_vkCreateComputePipelines = NULL;
4081 PFN_vkCreateDebugReportCallbackEXT glad_vkCreateDebugReportCallbackEXT = NULL;
4082 PFN_vkCreateDescriptorPool glad_vkCreateDescriptorPool = NULL;
4083 PFN_vkCreateDescriptorSetLayout glad_vkCreateDescriptorSetLayout = NULL;
4084 PFN_vkCreateDescriptorUpdateTemplate glad_vkCreateDescriptorUpdateTemplate = NULL;
4085 PFN_vkCreateDevice glad_vkCreateDevice = NULL;
4086 PFN_vkCreateEvent glad_vkCreateEvent = NULL;
4087 PFN_vkCreateFence glad_vkCreateFence = NULL;
4088 PFN_vkCreateFramebuffer glad_vkCreateFramebuffer = NULL;
4089 PFN_vkCreateGraphicsPipelines glad_vkCreateGraphicsPipelines = NULL;
4090 PFN_vkCreateImage glad_vkCreateImage = NULL;
4091 PFN_vkCreateImageView glad_vkCreateImageView = NULL;
4092 PFN_vkCreateInstance glad_vkCreateInstance = NULL;
4093 PFN_vkCreatePipelineCache glad_vkCreatePipelineCache = NULL;
4094 PFN_vkCreatePipelineLayout glad_vkCreatePipelineLayout = NULL;
4095 PFN_vkCreateQueryPool glad_vkCreateQueryPool = NULL;
4096 PFN_vkCreateRenderPass glad_vkCreateRenderPass = NULL;
4097 PFN_vkCreateSampler glad_vkCreateSampler = NULL;
4098 PFN_vkCreateSamplerYcbcrConversion glad_vkCreateSamplerYcbcrConversion = NULL;
4099 PFN_vkCreateSemaphore glad_vkCreateSemaphore = NULL;
4100 PFN_vkCreateShaderModule glad_vkCreateShaderModule = NULL;
4101 PFN_vkCreateSwapchainKHR glad_vkCreateSwapchainKHR = NULL;
4102 PFN_vkDebugReportMessageEXT glad_vkDebugReportMessageEXT = NULL;
4103 PFN_vkDestroyBuffer glad_vkDestroyBuffer = NULL;
4104 PFN_vkDestroyBufferView glad_vkDestroyBufferView = NULL;
4105 PFN_vkDestroyCommandPool glad_vkDestroyCommandPool = NULL;
4106 PFN_vkDestroyDebugReportCallbackEXT glad_vkDestroyDebugReportCallbackEXT = NULL;
4107 PFN_vkDestroyDescriptorPool glad_vkDestroyDescriptorPool = NULL;
4108 PFN_vkDestroyDescriptorSetLayout glad_vkDestroyDescriptorSetLayout = NULL;
4109 PFN_vkDestroyDescriptorUpdateTemplate glad_vkDestroyDescriptorUpdateTemplate = NULL;
4110 PFN_vkDestroyDevice glad_vkDestroyDevice = NULL;
4111 PFN_vkDestroyEvent glad_vkDestroyEvent = NULL;
4112 PFN_vkDestroyFence glad_vkDestroyFence = NULL;
4113 PFN_vkDestroyFramebuffer glad_vkDestroyFramebuffer = NULL;
4114 PFN_vkDestroyImage glad_vkDestroyImage = NULL;
4115 PFN_vkDestroyImageView glad_vkDestroyImageView = NULL;
4116 PFN_vkDestroyInstance glad_vkDestroyInstance = NULL;
4117 PFN_vkDestroyPipeline glad_vkDestroyPipeline = NULL;
4118 PFN_vkDestroyPipelineCache glad_vkDestroyPipelineCache = NULL;
4119 PFN_vkDestroyPipelineLayout glad_vkDestroyPipelineLayout = NULL;
4120 PFN_vkDestroyQueryPool glad_vkDestroyQueryPool = NULL;
4121 PFN_vkDestroyRenderPass glad_vkDestroyRenderPass = NULL;
4122 PFN_vkDestroySampler glad_vkDestroySampler = NULL;
4123 PFN_vkDestroySamplerYcbcrConversion glad_vkDestroySamplerYcbcrConversion = NULL;
4124 PFN_vkDestroySemaphore glad_vkDestroySemaphore = NULL;
4125 PFN_vkDestroyShaderModule glad_vkDestroyShaderModule = NULL;
4126 PFN_vkDestroySurfaceKHR glad_vkDestroySurfaceKHR = NULL;
4127 PFN_vkDestroySwapchainKHR glad_vkDestroySwapchainKHR = NULL;
4128 PFN_vkDeviceWaitIdle glad_vkDeviceWaitIdle = NULL;
4129 PFN_vkEndCommandBuffer glad_vkEndCommandBuffer = NULL;
```

```
4130 PFN_vkEnumerateDeviceExtensionProperties glad_vkEnumerateDeviceExtensionProperties = NULL;
4131 PFN_vkEnumerateDeviceLayerProperties glad_vkEnumerateDeviceLayerProperties = NULL;
4132 PFN_vkEnumerateInstanceExtensionProperties glad_vkEnumerateInstanceExtensionProperties = NULL;
4133 PFN_vkEnumerateInstanceLayerProperties glad_vkEnumerateInstanceLayerProperties = NULL;
4134 PFN_vkEnumerateInstanceVersion glad_vkEnumerateInstanceVersion = NULL;
4135 PFN_vkEnumeratePhysicalDeviceGroups glad_vkEnumeratePhysicalDeviceGroups = NULL;
4136 PFN_vkEnumeratePhysicalDevices glad_vkEnumeratePhysicalDevices = NULL;
4137 PFN_vkFlushMappedMemoryRanges glad_vkFlushMappedMemoryRanges = NULL;
4138 PFN_vkFreeCommandBuffers glad_vkFreeCommandBuffers = NULL;
4139 PFN_vkFreeDescriptorSets glad_vkFreeDescriptorSets = NULL;
4140 PFN_vkFreeMemory glad_vkFreeMemory = NULL;
4141 PFN_vkGetBufferMemoryRequirements glad_vkGetBufferMemoryRequirements = NULL;
4142 PFN_vkGetBufferMemoryRequirements2 glad_vkGetBufferMemoryRequirements2 = NULL;
4143 PFN_vkGetDescriptorSetLayoutSupport glad_vkGetDescriptorSetLayoutSupport = NULL;
4144 PFN_vkGetDeviceGroupPeerMemoryFeatures glad_vkGetDeviceGroupPeerMemoryFeatures = NULL;
4145 PFN_vkGetDeviceGroupPresentCapabilitiesKHR glad_vkGetDeviceGroupPresentCapabilitiesKHR = NULL;
4146 PFN_vkGetDeviceGroupSurfacePresentModesKHR glad_vkGetDeviceGroupSurfacePresentModesKHR = NULL;
4147 PFN_vkGetDeviceMemoryCommitment glad_vkGetDeviceMemoryCommitment = NULL;
4148 PFN_vkGetDeviceProcAddr glad_vkGetDeviceProcAddr = NULL;
4149 PFN_vkGetDeviceQueue glad_vkGetDeviceQueue = NULL;
4150 PFN_vkGetDeviceQueue2 glad_vkGetDeviceQueue2 = NULL;
4151 PFN_vkGetEventStatus glad_vkGetEventStatus = NULL;
4152 PFN_vkGetFenceStatus glad_vkGetFenceStatus = NULL;
4153 PFN_vkGetImageMemoryRequirements glad_vkGetImageMemoryRequirements = NULL;
4154 PFN_vkGetImageMemoryRequirements2 glad_vkGetImageMemoryRequirements2 = NULL;
4155 PFN_vkGetImageSparseMemoryRequirements glad_vkGetImageSparseMemoryRequirements = NULL;
4156 PFN_vkGetImageSparseMemoryRequirements2 glad_vkGetImageSparseMemoryRequirements2 = NULL;
4157 PFN_vkGetImageSubresourceLayout glad_vkGetImageSubresourceLayout = NULL;
4158 PFN_vkGetInstanceProcAddr glad_vkGetInstanceProcAddr = NULL;
4159 PFN_vkGetPhysicalDeviceExternalBufferProperties glad_vkGetPhysicalDeviceExternalBufferProperties =
       NULL;
4160 PFN_vkGetPhysicalDeviceExternalFenceProperties glad_vkGetPhysicalDeviceExternalFenceProperties = NULL;
4161 PFN_vkGetPhysicalDeviceExternalSemaphoreProperties glad_vkGetPhysicalDeviceExternalSemaphoreProperties
       = NULL;
4162 PFN_vkGetPhysicalDeviceFeatures glad_vkGetPhysicalDeviceFeatures = NULL;
4163 PFN_vkGetPhysicalDeviceFeatures2 glad_vkGetPhysicalDeviceFeatures2 = NULL;
4164 PFN_vkGetPhysicalDeviceFormatProperties glad_vkGetPhysicalDeviceFormatProperties = NULL;
4165 PFN_vkGetPhysicalDeviceFormatProperties2 glad_vkGetPhysicalDeviceFormatProperties2 = NULL;
4166 PFN_vkGetPhysicalDeviceImageFormatProperties glad_vkGetPhysicalDeviceImageFormatProperties = NULL;
4167 PFN_vkGetPhysicalDeviceImageFormatProperties2 glad_vkGetPhysicalDeviceImageFormatProperties2 = NULL;
4168 PFN_vkGetPhysicalDeviceMemoryProperties glad_vkGetPhysicalDeviceMemoryProperties = NULL;
4169 PFN_vkGetPhysicalDeviceMemoryProperties2 glad_vkGetPhysicalDeviceMemoryProperties2 = NULL;
4170 PFN_vkGetPhysicalDevicePresentRectanglesKHR glad_vkGetPhysicalDevicePresentRectanglesKHR = NULL;
4171 PFN_vkGetPhysicalDeviceProperties glad_vkGetPhysicalDeviceProperties = NULL;
4172 PFN_vkGetPhysicalDeviceProperties2 glad_vkGetPhysicalDeviceProperties2 = NULL;
4173 PFN_vkGetPhysicalDeviceQueueFamilyProperties glad_vkGetPhysicalDeviceQueueFamilyProperties = NULL;
4174 PFN_vkGetPhysicalDeviceQueueFamilyProperties2 glad_vkGetPhysicalDeviceQueueFamilyProperties2 = NULL;
4175 PFN_vkGetPhysicalDeviceSparseImageFormatProperties glad_vkGetPhysicalDeviceSparseImageFormatProperties
       = NULL;
4176 PFN_vkGetPhysicalDeviceSparseImageFormatProperties2
       glad_vkGetPhysicalDeviceSparseImageFormatProperties2 = NULL;
4177 PFN_vkGetPhysicalDeviceSurfaceCapabilitiesKHR glad_vkGetPhysicalDeviceSurfaceCapabilitiesKHR = NULL;
4178 PFN_vkGetPhysicalDeviceSurfaceFormatsKHR glad_vkGetPhysicalDeviceSurfaceFormatsKHR = NULL;
4179 PFN_vkGetPhysicalDeviceSurfacePresentModesKHR glad_vkGetPhysicalDeviceSurfacePresentModesKHR = NULL;
4180 PFN_vkGetPhysicalDeviceSurfaceSupportKHR glad_vkGetPhysicalDeviceSurfaceSupportKHR = NULL;
4181 PFN_vkGetPipelineCacheData glad_vkGetPipelineCacheData = NULL;
4182 PFN_vkGetQueryPoolResults glad_vkGetQueryPoolResults = NULL;
4183 PFN_vkGetRenderAreaGranularity glad_vkGetRenderAreaGranularity = NULL;
4184 PFN_vkGetSwapchainImagesKHR glad_vkGetSwapchainImagesKHR = NULL;
4185 PFN_vkInvalidateMappedMemoryRanges glad_vkInvalidateMappedMemoryRanges = NULL;
4186 PFN_vkMapMemory glad_vkMapMemory = NULL;
4187 PFN_vkMergePipelineCaches glad_vkMergePipelineCaches = NULL;
4188 PFN_vkQueueBindSparse glad_vkQueueBindSparse = NULL;
4189 PFN_vkQueuePresentKHR glad_vkQueuePresentKHR = NULL;
4190 PFN_vkQueueSubmit glad_vkQueueSubmit = NULL;
4191 PFN_vkQueueWaitIdle glad_vkQueueWaitIdle = NULL;
4192 PFN_vkResetCommandBuffer glad_vkResetCommandBuffer = NULL;
4193 PFN_vkResetCommandPool glad_vkResetCommandPool = NULL;
4194 PFN_vkResetDescriptorPool glad_vkResetDescriptorPool = NULL;
4195 PFN_vkResetEvent glad_vkResetEvent = NULL;
4196 PFN_vkResetFences glad_vkResetFences = NULL;
4197 PFN_vkSetEvent glad_vkSetEvent = NULL;
4198 PFN_vkTrimCommandPool glad_vkTrimCommandPool = NULL;
4199 PFN_vkUnmapMemory glad_vkUnmapMemory = NULL;
4200 PFN_vkUpdateDescriptorSetWithTemplate glad_vkUpdateDescriptorSetWithTemplate = NULL;
4201 PFN_vkUpdateDescriptorSets glad_vkUpdateDescriptorSets = NULL;
4202 PFN_vkWaitForFences glad_vkWaitForFences = NULL;
4203
4204
4205 static void glad_vk_load_VK_VERSION_1_0( GLADuserptrloadfunc load, void* userptr) {
4206     if(!GLAD_VK_VERSION_1_0) return;
4207     glad_vkAllocateCommandBuffers = (PFN_vkAllocateCommandBuffers) load(userptr,
       "vkAllocateCommandBuffers");
4208     glad_vkAllocateDescriptorSets = (PFN_vkAllocateDescriptorSets) load(userptr,
       "vkAllocateDescriptorSets");
4209     glad_vkAllocateMemory = (PFN_vkAllocateMemory) load(userptr, "vkAllocateMemory");
4210     glad_vkBeginCommandBuffer = (PFN_vkBeginCommandBuffer) load(userptr, "vkBeginCommandBuffer");
```

```
4211     glad_vkBindBufferMemory = (PFN_vkBindBufferMemory) load(userptr, "vkBindBufferMemory");
4212     glad_vkBindImageMemory = (PFN_vkBindImageMemory) load(userptr, "vkBindImageMemory");
4213     glad_vkCmdBeginQuery = (PFN_vkCmdBeginQuery) load(userptr, "vkCmdBeginQuery");
4214     glad_vkCmdBeginRenderPass = (PFN_vkCmdBeginRenderPass) load(userptr, "vkCmdBeginRenderPass");
4215     glad_vkCmdBindDescriptorSets = (PFN_vkCmdBindDescriptorSets) load(userptr,
    "vkCmdBindDescriptorSets");
4216     glad_vkCmdBindIndexBuffer = (PFN_vkCmdBindIndexBuffer) load(userptr, "vkCmdBindIndexBuffer");
4217     glad_vkCmdBindPipeline = (PFN_vkCmdBindPipeline) load(userptr, "vkCmdBindPipeline");
4218     glad_vkCmdBindVertexBuffers = (PFN_vkCmdBindVertexBuffers) load(userptr, "vkCmdBindVertexBuffers");
4219     glad_vkCmdBlitImage = (PFN_vkCmdBlitImage) load(userptr, "vkCmdBlitImage");
4220     glad_vkCmdClearAttachments = (PFN_vkCmdClearAttachments) load(userptr, "vkCmdClearAttachments");
4221     glad_vkCmdClearColorImage = (PFN_vkCmdClearColorImage) load(userptr, "vkCmdClearColorImage");
4222     glad_vkCmdClearDepthStencilImage = (PFN_vkCmdClearDepthStencilImage) load(userptr,
    "vkCmdClearDepthStencilImage");
4223     glad_vkCmdCopyBuffer = (PFN_vkCmdCopyBuffer) load(userptr, "vkCmdCopyBuffer");
4224     glad_vkCmdCopyBufferToImage = (PFN_vkCmdCopyBufferToImage) load(userptr, "vkCmdCopyBufferToImage");
4225     glad_vkCmdCopyImage = (PFN_vkCmdCopyImage) load(userptr, "vkCmdCopyImage");
4226     glad_vkCmdCopyImageToBuffer = (PFN_vkCmdCopyImageToBuffer) load(userptr, "vkCmdCopyImageToBuffer");
4227     glad_vkCmdCopyQueryPoolResults = (PFN_vkCmdCopyQueryPoolResults) load(userptr,
    "vkCmdCopyQueryPoolResults");
4228     glad_vkCmdDispatch = (PFN_vkCmdDispatch) load(userptr, "vkCmdDispatch");
4229     glad_vkCmdDispatchIndirect = (PFN_vkCmdDispatchIndirect) load(userptr, "vkCmdDispatchIndirect");
4230     glad_vkCmdDraw = (PFN_vkCmdDraw) load(userptr, "vkCmdDraw");
4231     glad_vkCmdDrawIndexed = (PFN_vkCmdDrawIndexed) load(userptr, "vkCmdDrawIndexed");
4232     glad_vkCmdDrawIndexedIndirect = (PFN_vkCmdDrawIndexedIndirect) load(userptr,
    "vkCmdDrawIndexedIndirect");
4233     glad_vkCmdDrawIndirect = (PFN_vkCmdDrawIndirect) load(userptr, "vkCmdDrawIndirect");
4234     glad_vkCmdEndQuery = (PFN_vkCmdEndQuery) load(userptr, "vkCmdEndQuery");
4235     glad_vkCmdEndRenderPass = (PFN_vkCmdEndRenderPass) load(userptr, "vkCmdEndRenderPass");
4236     glad_vkCmdExecuteCommands = (PFN_vkCmdExecuteCommands) load(userptr, "vkCmdExecuteCommands");
4237     glad_vkCmdFillBuffer = (PFN_vkCmdFillBuffer) load(userptr, "vkCmdFillBuffer");
4238     glad_vkCmdNextSubpass = (PFN_vkCmdNextSubpass) load(userptr, "vkCmdNextSubpass");
4239     glad_vkCmdPipelineBarrier = (PFN_vkCmdPipelineBarrier) load(userptr, "vkCmdPipelineBarrier");
4240     glad_vkCmdPushConstants = (PFN_vkCmdPushConstants) load(userptr, "vkCmdPushConstants");
4241     glad_vkCmdResetEvent = (PFN_vkCmdResetEvent) load(userptr, "vkCmdResetEvent");
4242     glad_vkCmdResetQueryPool = (PFN_vkCmdResetQueryPool) load(userptr, "vkCmdResetQueryPool");
4243     glad_vkCmdResolveImage = (PFN_vkCmdResolveImage) load(userptr, "vkCmdResolveImage");
4244     glad_vkCmdSetBlendConstants = (PFN_vkCmdSetBlendConstants) load(userptr, "vkCmdSetBlendConstants");
4245     glad_vkCmdSetDepthBias = (PFN_vkCmdSetDepthBias) load(userptr, "vkCmdSetDepthBias");
4246     glad_vkCmdSetDepthBounds = (PFN_vkCmdSetDepthBounds) load(userptr, "vkCmdSetDepthBounds");
4247     glad_vkCmdSetEvent = (PFN_vkCmdSetEvent) load(userptr, "vkCmdSetEvent");
4248     glad_vkCmdSetLineWidth = (PFN_vkCmdSetLineWidth) load(userptr, "vkCmdSetLineWidth");
4249     glad_vkCmdSetScissor = (PFN_vkCmdSetScissor) load(userptr, "vkCmdSetScissor");
4250     glad_vkCmdSetStencilCompareMask = (PFN_vkCmdSetStencilCompareMask) load(userptr,
    "vkCmdSetStencilCompareMask");
4251     glad_vkCmdSetStencilReference = (PFN_vkCmdSetStencilReference) load(userptr,
    "vkCmdSetStencilReference");
4252     glad_vkCmdSetStencilWriteMask = (PFN_vkCmdSetStencilWriteMask) load(userptr,
    "vkCmdSetStencilWriteMask");
4253     glad_vkCmdSetViewport = (PFN_vkCmdSetViewport) load(userptr, "vkCmdSetViewport");
4254     glad_vkCmdUpdateBuffer = (PFN_vkCmdUpdateBuffer) load(userptr, "vkCmdUpdateBuffer");
4255     glad_vkCmdWaitEvents = (PFN_vkCmdWaitEvents) load(userptr, "vkCmdWaitEvents");
4256     glad_vkCmdWriteTimestamp = (PFN_vkCmdWriteTimestamp) load(userptr, "vkCmdWriteTimestamp");
4257     glad_vkCreateBuffer = (PFN_vkCreateBuffer) load(userptr, "vkCreateBuffer");
4258     glad_vkCreateBufferView = (PFN_vkCreateBufferView) load(userptr, "vkCreateBufferView");
4259     glad_vkCreateCommandPool = (PFN_vkCreateCommandPool) load(userptr, "vkCreateCommandPool");
4260     glad_vkCreateComputePipelines = (PFN_vkCreateComputePipelines) load(userptr,
    "vkCreateComputePipelines");
4261     glad_vkCreateDescriptorPool = (PFN_vkCreateDescriptorPool) load(userptr, "vkCreateDescriptorPool");
4262     glad_vkCreateDescriptorSetLayout = (PFN_vkCreateDescriptorSetLayout) load(userptr,
    "vkCreateDescriptorSetLayout");
4263     glad_vkCreateDevice = (PFN_vkCreateDevice) load(userptr, "vkCreateDevice");
4264     glad_vkCreateEvent = (PFN_vkCreateEvent) load(userptr, "vkCreateEvent");
4265     glad_vkCreateFence = (PFN_vkCreateFence) load(userptr, "vkCreateFence");
4266     glad_vkCreateFramebuffer = (PFN_vkCreateFramebuffer) load(userptr, "vkCreateFramebuffer");
4267     glad_vkCreateGraphicsPipelines = (PFN_vkCreateGraphicsPipelines) load(userptr,
    "vkCreateGraphicsPipelines");
4268     glad_vkCreateImage = (PFN_vkCreateImage) load(userptr, "vkCreateImage");
4269     glad_vkCreateImageView = (PFN_vkCreateImageView) load(userptr, "vkCreateImageView");
4270     glad_vkCreateInstance = (PFN_vkCreateInstance) load(userptr, "vkCreateInstance");
4271     glad_vkCreatePipelineCache = (PFN_vkCreatePipelineCache) load(userptr, "vkCreatePipelineCache");
4272     glad_vkCreatePipelineLayout = (PFN_vkCreatePipelineLayout) load(userptr, "vkCreatePipelineLayout");
4273     glad_vkCreateQueryPool = (PFN_vkCreateQueryPool) load(userptr, "vkCreateQueryPool");
4274     glad_vkCreateRenderPass = (PFN_vkCreateRenderPass) load(userptr, "vkCreateRenderPass");
4275     glad_vkCreateSampler = (PFN_vkCreateSampler) load(userptr, "vkCreateSampler");
4276     glad_vkCreateSemaphore = (PFN_vkCreateSemaphore) load(userptr, "vkCreateSemaphore");
4277     glad_vkCreateShaderModule = (PFN_vkCreateShaderModule) load(userptr, "vkCreateShaderModule");
4278     glad_vkDestroyBuffer = (PFN_vkDestroyBuffer) load(userptr, "vkDestroyBuffer");
4279     glad_vkDestroyBufferView = (PFN_vkDestroyBufferView) load(userptr, "vkDestroyBufferView");
4280     glad_vkDestroyCommandPool = (PFN_vkDestroyCommandPool) load(userptr, "vkDestroyCommandPool");
4281     glad_vkDestroyDescriptorPool = (PFN_vkDestroyDescriptorPool) load(userptr,
    "vkDestroyDescriptorPool");
4282     glad_vkDestroyDescriptorSetLayout = (PFN_vkDestroyDescriptorSetLayout) load(userptr,
    "vkDestroyDescriptorSetLayout");
4283     glad_vkDestroyDevice = (PFN_vkDestroyDevice) load(userptr, "vkDestroyDevice");
4284     glad_vkDestroyEvent = (PFN_vkDestroyEvent) load(userptr, "vkDestroyEvent");
4285     glad_vkDestroyFence = (PFN_vkDestroyFence) load(userptr, "vkDestroyFence");
```

```
4286     glad_vkDestroyFramebuffer = (PFN_vkDestroyFramebuffer) load(userptr, "vkDestroyFramebuffer");
4287     glad_vkDestroyImage = (PFN_vkDestroyImage) load(userptr, "vkDestroyImage");
4288     glad_vkDestroyImageView = (PFN_vkDestroyImageView) load(userptr, "vkDestroyImageView");
4289     glad_vkDestroyInstance = (PFN_vkDestroyInstance) load(userptr, "vkDestroyInstance");
4290     glad_vkDestroyPipeline = (PFN_vkDestroyPipeline) load(userptr, "vkDestroyPipeline");
4291     glad_vkDestroyPipelineCache = (PFN_vkDestroyPipelineCache) load(userptr, "vkDestroyPipelineCache");
4292     glad_vkDestroyPipelineLayout = (PFN_vkDestroyPipelineLayout) load(userptr,
     "vkDestroyPipelineLayout");
4293     glad_vkDestroyQueryPool = (PFN_vkDestroyQueryPool) load(userptr, "vkDestroyQueryPool");
4294     glad_vkDestroyRenderPass = (PFN_vkDestroyRenderPass) load(userptr, "vkDestroyRenderPass");
4295     glad_vkDestroySampler = (PFN_vkDestroySampler) load(userptr, "vkDestroySampler");
4296     glad_vkDestroySemaphore = (PFN_vkDestroySemaphore) load(userptr, "vkDestroySemaphore");
4297     glad_vkDestroyShaderModule = (PFN_vkDestroyShaderModule) load(userptr, "vkDestroyShaderModule");
4298     glad_vkDeviceWaitIdle = (PFN_vkDeviceWaitIdle) load(userptr, "vkDeviceWaitIdle");
4299     glad_vkEndCommandBuffer = (PFN_vkEndCommandBuffer) load(userptr, "vkEndCommandBuffer");
4300     glad_vkEnumerateDeviceExtensionProperties = (PFN_vkEnumerateDeviceExtensionProperties)
     load(userptr, "vkEnumerateDeviceExtensionProperties");
4301     glad_vkEnumerateDeviceLayerProperties = (PFN_vkEnumerateDeviceLayerProperties) load(userptr,
     "vkEnumerateDeviceLayerProperties");
4302     glad_vkEnumerateInstanceExtensionProperties = (PFN_vkEnumerateInstanceExtensionProperties)
     load(userptr, "vkEnumerateInstanceExtensionProperties");
4303     glad_vkEnumerateInstanceLayerProperties = (PFN_vkEnumerateInstanceLayerProperties) load(userptr,
     "vkEnumerateInstanceLayerProperties");
4304     glad_vkEnumeratePhysicalDevices = (PFN_vkEnumeratePhysicalDevices) load(userptr,
     "vkEnumeratePhysicalDevices");
4305     glad_vkFlushMappedMemoryRanges = (PFN_vkFlushMappedMemoryRanges) load(userptr,
     "vkFlushMappedMemoryRanges");
4306     glad_vkFreeCommandBuffers = (PFN_vkFreeCommandBuffers) load(userptr, "vkFreeCommandBuffers");
4307     glad_vkFreeDescriptorSets = (PFN_vkFreeDescriptorSets) load(userptr, "vkFreeDescriptorSets");
4308     glad_vkFreeMemory = (PFN_vkFreeMemory) load(userptr, "vkFreeMemory");
4309     glad_vkGetBufferMemoryRequirements = (PFN_vkGetBufferMemoryRequirements) load(userptr,
     "vkGetBufferMemoryRequirements");
4310     glad_vkGetDeviceMemoryCommitment = (PFN_vkGetDeviceMemoryCommitment) load(userptr,
     "vkGetDeviceMemoryCommitment");
4311     glad_vkGetDeviceProcAddr = (PFN_vkGetDeviceProcAddr) load(userptr, "vkGetDeviceProcAddr");
4312     glad_vkGetDeviceQueue = (PFN_vkGetDeviceQueue) load(userptr, "vkGetDeviceQueue");
4313     glad_vkGetEventStatus = (PFN_vkGetEventStatus) load(userptr, "vkGetEventStatus");
4314     glad_vkGetFenceStatus = (PFN_vkGetFenceStatus) load(userptr, "vkGetFenceStatus");
4315     glad_vkGetImageMemoryRequirements = (PFN_vkGetImageMemoryRequirements) load(userptr,
     "vkGetImageMemoryRequirements");
4316     glad_vkGetImageSparseMemoryRequirements = (PFN_vkGetImageSparseMemoryRequirements) load(userptr,
     "vkGetImageSparseMemoryRequirements");
4317     glad_vkGetImageSubresourceLayout = (PFN_vkGetImageSubresourceLayout) load(userptr,
     "vkGetImageSubresourceLayout");
4318     glad_vkGetInstanceProcAddr = (PFN_vkGetInstanceProcAddr) load(userptr, "vkGetInstanceProcAddr");
4319     glad_vkGetPhysicalDeviceFeatures = (PFN_vkGetPhysicalDeviceFeatures) load(userptr,
     "vkGetPhysicalDeviceFeatures");
4320     glad_vkGetPhysicalDeviceFormatProperties = (PFN_vkGetPhysicalDeviceFormatProperties) load(userptr,
     "vkGetPhysicalDeviceFormatProperties");
4321     glad_vkGetPhysicalDeviceImageFormatProperties = (PFN_vkGetPhysicalDeviceImageFormatProperties)
     load(userptr, "vkGetPhysicalDeviceImageFormatProperties");
4322     glad_vkGetPhysicalDeviceMemoryProperties = (PFN_vkGetPhysicalDeviceMemoryProperties) load(userptr,
     "vkGetPhysicalDeviceMemoryProperties");
4323     glad_vkGetPhysicalDeviceProperties = (PFN_vkGetPhysicalDeviceProperties) load(userptr,
     "vkGetPhysicalDeviceProperties");
4324     glad_vkGetPhysicalDeviceQueueFamilyProperties = (PFN_vkGetPhysicalDeviceQueueFamilyProperties)
     load(userptr, "vkGetPhysicalDeviceQueueFamilyProperties");
4325     glad_vkGetPhysicalDeviceSparseImageFormatProperties =
     (PFN_vkGetPhysicalDeviceSparseImageFormatProperties) load(userptr,
     "vkGetPhysicalDeviceSparseImageFormatProperties");
4326     glad_vkGetPipelineCacheData = (PFN_vkGetPipelineCacheData) load(userptr, "vkGetPipelineCacheData");
4327     glad_vkGetQueryPoolResults = (PFN_vkGetQueryPoolResults) load(userptr, "vkGetQueryPoolResults");
4328     glad_vkGetRenderAreaGranularity = (PFN_vkGetRenderAreaGranularity) load(userptr,
     "vkGetRenderAreaGranularity");
4329     glad_vkInvalidateMappedMemoryRanges = (PFN_vkInvalidateMappedMemoryRanges) load(userptr,
     "vkInvalidateMappedMemoryRanges");
4330     glad_vkMapMemory = (PFN_vkMapMemory) load(userptr, "vkMapMemory");
4331     glad_vkMergePipelineCaches = (PFN_vkMergePipelineCaches) load(userptr, "vkMergePipelineCaches");
4332     glad_vkQueueBindSparse = (PFN_vkQueueBindSparse) load(userptr, "vkQueueBindSparse");
4333     glad_vkQueueSubmit = (PFN_vkQueueSubmit) load(userptr, "vkQueueSubmit");
4334     glad_vkQueueWaitIdle = (PFN_vkQueueWaitIdle) load(userptr, "vkQueueWaitIdle");
4335     glad_vkResetCommandBuffer = (PFN_vkResetCommandBuffer) load(userptr, "vkResetCommandBuffer");
4336     glad_vkResetCommandPool = (PFN_vkResetCommandPool) load(userptr, "vkResetCommandPool");
4337     glad_vkResetDescriptorPool = (PFN_vkResetDescriptorPool) load(userptr, "vkResetDescriptorPool");
4338     glad_vkResetEvent = (PFN_vkResetEvent) load(userptr, "vkResetEvent");
4339     glad_vkResetFences = (PFN_vkResetFences) load(userptr, "vkResetFences");
4340     glad_vkSetEvent = (PFN_vkSetEvent) load(userptr, "vkSetEvent");
4341     glad_vkUnmapMemory = (PFN_vkUnmapMemory) load(userptr, "vkUnmapMemory");
4342     glad_vkUpdateDescriptorSets = (PFN_vkUpdateDescriptorSets) load(userptr, "vkUpdateDescriptorSets");
4343     glad_vkWaitForFences = (PFN_vkWaitForFences) load(userptr, "vkWaitForFences");
4344 }
4345 static void glad_vk_load_VK_VERSION_1_1( GLADuserptrloadfunc load, void* userptr) {
4346     if(!GLAD_VK_VERSION_1_1) return;
4347     glad_vkBindBufferMemory2 = (PFN_vkBindBufferMemory2) load(userptr, "vkBindBufferMemory2");
4348     glad_vkBindImageMemory2 = (PFN_vkBindImageMemory2) load(userptr, "vkBindImageMemory2");
4349     glad_vkCmdDispatchBase = (PFN_vkCmdDispatchBase) load(userptr, "vkCmdDispatchBase");
4350     glad_vkCmdSetDeviceMask = (PFN_vkCmdSetDeviceMask) load(userptr, "vkCmdSetDeviceMask");
```

```
4351     glad_vkCreateDescriptorUpdateTemplate = (PFN_vkCreateDescriptorUpdateTemplate) load(userptr,
      "vkCreateDescriptorUpdateTemplate");
4352     glad_vkCreateSamplerYcbcrConversion = (PFN_vkCreateSamplerYcbcrConversion) load(userptr,
      "vkCreateSamplerYcbcrConversion");
4353     glad_vkDestroyDescriptorUpdateTemplate = (PFN_vkDestroyDescriptorUpdateTemplate) load(userptr,
      "vkDestroyDescriptorUpdateTemplate");
4354     glad_vkDestroySamplerYcbcrConversion = (PFN_vkDestroySamplerYcbcrConversion) load(userptr,
      "vkDestroySamplerYcbcrConversion");
4355     glad_vkEnumerateInstanceVersion = (PFN_vkEnumerateInstanceVersion) load(userptr,
      "vkEnumerateInstanceVersion");
4356     glad_vkEnumeratePhysicalDeviceGroups = (PFN_vkEnumeratePhysicalDeviceGroups) load(userptr,
      "vkEnumeratePhysicalDeviceGroups");
4357     glad_vkGetBufferMemoryRequirements2 = (PFN_vkGetBufferMemoryRequirements2) load(userptr,
      "vkGetBufferMemoryRequirements2");
4358     glad_vkGetDescriptorSetLayoutSupport = (PFN_vkGetDescriptorSetLayoutSupport) load(userptr,
      "vkGetDescriptorSetLayoutSupport");
4359     glad_vkGetDeviceGroupPeerMemoryFeatures = (PFN_vkGetDeviceGroupPeerMemoryFeatures) load(userptr,
      "vkGetDeviceGroupPeerMemoryFeatures");
4360     glad_vkGetDeviceQueue2 = (PFN_vkGetDeviceQueue2) load(userptr, "vkGetDeviceQueue2");
4361     glad_vkGetImageMemoryRequirements2 = (PFN_vkGetImageMemoryRequirements2) load(userptr,
      "vkGetImageMemoryRequirements2");
4362     glad_vkGetImageSparseMemoryRequirements2 = (PFN_vkGetImageSparseMemoryRequirements2) load(userptr,
      "vkGetImageSparseMemoryRequirements2");
4363     glad_vkGetPhysicalDeviceExternalBufferProperties =
      (PFN_vkGetPhysicalDeviceExternalBufferProperties) load(userptr,
      "vkGetPhysicalDeviceExternalBufferProperties");
4364     glad_vkGetPhysicalDeviceExternalFenceProperties = (PFN_vkGetPhysicalDeviceExternalFenceProperties)
      load(userptr, "vkGetPhysicalDeviceExternalFenceProperties");
4365     glad_vkGetPhysicalDeviceExternalSemaphoreProperties =
      (PFN_vkGetPhysicalDeviceExternalSemaphoreProperties) load(userptr,
      "vkGetPhysicalDeviceExternalSemaphoreProperties");
4366     glad_vkGetPhysicalDeviceFeatures2 = (PFN_vkGetPhysicalDeviceFeatures2) load(userptr,
      "vkGetPhysicalDeviceFeatures2");
4367     glad_vkGetPhysicalDeviceFormatProperties2 = (PFN_vkGetPhysicalDeviceFormatProperties2)
      load(userptr, "vkGetPhysicalDeviceFormatProperties2");
4368     glad_vkGetPhysicalDeviceImageFormatProperties2 = (PFN_vkGetPhysicalDeviceImageFormatProperties2)
      load(userptr, "vkGetPhysicalDeviceImageFormatProperties2");
4369     glad_vkGetPhysicalDeviceMemoryProperties2 = (PFN_vkGetPhysicalDeviceMemoryProperties2)
      load(userptr, "vkGetPhysicalDeviceMemoryProperties2");
4370     glad_vkGetPhysicalDeviceProperties2 = (PFN_vkGetPhysicalDeviceProperties2) load(userptr,
      "vkGetPhysicalDeviceProperties2");
4371     glad_vkGetPhysicalDeviceQueueFamilyProperties2 = (PFN_vkGetPhysicalDeviceQueueFamilyProperties2)
      load(userptr, "vkGetPhysicalDeviceQueueFamilyProperties2");
4372     glad_vkGetPhysicalDeviceSparseImageFormatProperties2 =
      (PFN_vkGetPhysicalDeviceSparseImageFormatProperties2) load(userptr,
      "vkGetPhysicalDeviceSparseImageFormatProperties2");
4373     glad_vkTrimCommandPool = (PFN_vkTrimCommandPool) load(userptr, "vkTrimCommandPool");
4374     glad_vkUpdateDescriptorSetWithTemplate = (PFN_vkUpdateDescriptorSetWithTemplate) load(userptr,
      "vkUpdateDescriptorSetWithTemplate");
4375 }
4376 static void glad_vk_load_VK_EXT_debug_report( GLADuserptrloadfunc load, void* userptr) {
4377     if(!GLAD_VK_EXT_debug_report) return;
4378     glad_vkCreateDebugReportCallbackEXT = (PFN_vkCreateDebugReportCallbackEXT) load(userptr,
      "vkCreateDebugReportCallbackEXT");
4379     glad_vkDebugReportMessageEXT = (PFN_vkDebugReportMessageEXT) load(userptr,
      "vkDebugReportMessageEXT");
4380     glad_vkDestroyDebugReportCallbackEXT = (PFN_vkDestroyDebugReportCallbackEXT) load(userptr,
      "vkDestroyDebugReportCallbackEXT");
4381 }
4382 static void glad_vk_load_VK_KHR_surface( GLADuserptrloadfunc load, void* userptr) {
4383     if(!GLAD_VK_KHR_surface) return;
4384     glad_vkDestroySurfaceKHR = (PFN_vkDestroySurfaceKHR) load(userptr, "vkDestroySurfaceKHR");
4385     glad_vkGetPhysicalDeviceSurfaceCapabilitiesKHR = (PFN_vkGetPhysicalDeviceSurfaceCapabilitiesKHR)
      load(userptr, "vkGetPhysicalDeviceSurfaceCapabilitiesKHR");
4386     glad_vkGetPhysicalDeviceSurfaceFormatsKHR = (PFN_vkGetPhysicalDeviceSurfaceFormatsKHR)
      load(userptr, "vkGetPhysicalDeviceSurfaceFormatsKHR");
4387     glad_vkGetPhysicalDeviceSurfacePresentModesKHR = (PFN_vkGetPhysicalDeviceSurfacePresentModesKHR)
      load(userptr, "vkGetPhysicalDeviceSurfacePresentModesKHR");
4388     glad_vkGetPhysicalDeviceSurfaceSupportKHR = (PFN_vkGetPhysicalDeviceSurfaceSupportKHR)
      load(userptr, "vkGetPhysicalDeviceSurfaceSupportKHR");
4389 }
4390 static void glad_vk_load_VK_KHR_swapchain( GLADuserptrloadfunc load, void* userptr) {
4391     if(!GLAD_VK_KHR_swapchain) return;
4392     glad_vkAcquireNextImage2KHR = (PFN_vkAcquireNextImage2KHR) load(userptr, "vkAcquireNextImage2KHR");
4393     glad_vkAcquireNextImageKHR = (PFN_vkAcquireNextImageKHR) load(userptr, "vkAcquireNextImageKHR");
4394     glad_vkCreateSwapchainKHR = (PFN_vkCreateSwapchainKHR) load(userptr, "vkCreateSwapchainKHR");
4395     glad_vkDestroySwapchainKHR = (PFN_vkDestroySwapchainKHR) load(userptr, "vkDestroySwapchainKHR");
4396     glad_vkGetDeviceGroupPresentCapabilitiesKHR = (PFN_vkGetDeviceGroupPresentCapabilitiesKHR)
      load(userptr, "vkGetDeviceGroupPresentCapabilitiesKHR");
4397     glad_vkGetDeviceGroupSurfacePresentModesKHR = (PFN_vkGetDeviceGroupSurfacePresentModesKHR)
      load(userptr, "vkGetDeviceGroupSurfacePresentModesKHR");
4398     glad_vkGetPhysicalDevicePresentRectanglesKHR = (PFN_vkGetPhysicalDevicePresentRectanglesKHR)
      load(userptr, "vkGetPhysicalDevicePresentRectanglesKHR");
4399     glad_vkGetSwapchainImagesKHR = (PFN_vkGetSwapchainImagesKHR) load(userptr,
      "vkGetSwapchainImagesKHR");
4400     glad_vkQueuePresentKHR = (PFN_vkQueuePresentKHR) load(userptr, "vkQueuePresentKHR");
4401 }
```

```
4402
4403
4404
4405 static int glad_vk_get_extensions( VkPhysicalDevice physical_device, uint32_t *out_extension_count,
     char ***out_extensions) {
4406     uint32_t i;
4407     uint32_t instance_extension_count = 0;
4408     uint32_t device_extension_count = 0;
4409     uint32_t max_extension_count = 0;
4410     uint32_t total_extension_count = 0;
4411     char **extensions = NULL;
4412     VkExtensionProperties *ext_properties = NULL;
4413     VkResult result;
4414
4415     if (glad_vkEnumerateInstanceExtensionProperties == NULL || (physical_device != NULL &&
     glad_vkEnumerateDeviceExtensionProperties == NULL)) {
4416         return 0;
4417     }
4418
4419     result = glad_vkEnumerateInstanceExtensionProperties(NULL, &instance_extension_count, NULL);
4420     if (result != VK_SUCCESS) {
4421         return 0;
4422     }
4423
4424     if (physical_device != NULL) {
4425         result = glad_vkEnumerateDeviceExtensionProperties(physical_device, NULL,
     &device_extension_count, NULL);
4426         if (result != VK_SUCCESS) {
4427             return 0;
4428         }
4429     }
4430
4431     total_extension_count = instance_extension_count + device_extension_count;
4432     if (total_extension_count <= 0) {
4433         return 0;
4434     }
4435
4436     max_extension_count = instance_extension_count > device_extension_count
4437         ? instance_extension_count : device_extension_count;
4438
4439     ext_properties = (VkExtensionProperties*) malloc(max_extension_count *
     sizeof(VkExtensionProperties));
4440     if (ext_properties == NULL) {
4441         goto glad_vk_get_extensions_error;
4442     }
4443
4444     result = glad_vkEnumerateInstanceExtensionProperties(NULL, &instance_extension_count,
     ext_properties);
4445     if (result != VK_SUCCESS) {
4446         goto glad_vk_get_extensions_error;
4447     }
4448
4449     extensions = (char**) calloc(total_extension_count, sizeof(char*));
4450     if (extensions == NULL) {
4451         goto glad_vk_get_extensions_error;
4452     }
4453
4454     for (i = 0; i < instance_extension_count; ++i) {
4455         VkExtensionProperties ext = ext_properties[i];
4456
4457         size_t extension_name_length = strlen(ext.extensionName) + 1;
4458         extensions[i] = (char*) malloc(extension_name_length * sizeof(char));
4459         if (extensions[i] == NULL) {
4460             goto glad_vk_get_extensions_error;
4461         }
4462         memcpy(extensions[i], ext.extensionName, extension_name_length * sizeof(char));
4463     }
4464
4465     if (physical_device != NULL) {
4466         result = glad_vkEnumerateDeviceExtensionProperties(physical_device, NULL,
     &device_extension_count, ext_properties);
4467         if (result != VK_SUCCESS) {
4468             goto glad_vk_get_extensions_error;
4469         }
4470
4471         for (i = 0; i < device_extension_count; ++i) {
4472             VkExtensionProperties ext = ext_properties[i];
4473
4474             size_t extension_name_length = strlen(ext.extensionName) + 1;
4475             extensions[instance_extension_count + i] = (char*) malloc(extension_name_length *
     sizeof(char));
4476             if (extensions[instance_extension_count + i] == NULL) {
4477                 goto glad_vk_get_extensions_error;
4478             }
4479             memcpy(extensions[instance_extension_count + i], ext.extensionName, extension_name_length *
     sizeof(char));
4480         }
```

```
4481     }
4482
4483     free((void*) ext_properties);
4484
4485     *out_extension_count = total_extension_count;
4486     *out_extensions = extensions;
4487
4488     return 1;
4489
4490 glad_vk_get_extensions_error:
4491     free((void*) ext_properties);
4492     if (extensions != NULL) {
4493         for (i = 0; i < total_extension_count; ++i) {
4494             free((void*) extensions[i]);
4495         }
4496         free(extensions);
4497     }
4498     return 0;
4499 }
4500
4501 static void glad_vk_free_extensions(uint32_t extension_count, char **extensions) {
4502     uint32_t i;
4503
4504     for(i = 0; i < extension_count ; ++i) {
4505         free((void*) (extensions[i]));
4506     }
4507
4508     free((void*) extensions);
4509 }
4510
4511 static int glad_vk_has_extension(const char *name, uint32_t extension_count, char **extensions) {
4512     uint32_t i;
4513
4514     for (i = 0; i < extension_count; ++i) {
4515         if(extensions[i] != NULL && strcmp(name, extensions[i]) == 0) {
4516             return 1;
4517         }
4518     }
4519
4520     return 0;
4521 }
4522
4523 static GLADapiproc glad_vk_get_proc_from_userptr(void *userptr, const char* name) {
4524     return (GLAD_GNUC_EXTENSION (GLADapiproc (*)(const char *name)) userptr)(name);
4525 }
4526
4527 static int glad_vk_find_extensions_vulkan( VkPhysicalDevice physical_device) {
4528     uint32_t extension_count = 0;
4529     char **extensions = NULL;
4530     if (!glad_vk_get_extensions(physical_device, &extension_count, &extensions)) return 0;
4531
4532     GLAD_VK_EXT_debug_report = glad_vk_has_extension("VK_EXT_debug_report", extension_count,
      extensions);
4533     GLAD_VK_KHR_surface = glad_vk_has_extension("VK_KHR_surface", extension_count, extensions);
4534     GLAD_VK_KHR_swapchain = glad_vk_has_extension("VK_KHR_swapchain", extension_count, extensions);
4535
4536     (void) glad_vk_has_extension;
4537
4538     glad_vk_free_extensions(extension_count, extensions);
4539
4540     return 1;
4541 }
4542
4543 static int glad_vk_find_core_vulkan( VkPhysicalDevice physical_device) {
4544     int major = 1;
4545     int minor = 0;
4546
4547 #ifdef VK_VERSION_1_1
4548     if (glad_vkEnumerateInstanceVersion != NULL) {
4549         uint32_t version;
4550         VkResult result;
4551
4552         result = glad_vkEnumerateInstanceVersion(&version);
4553         if (result == VK_SUCCESS) {
4554             major = (int) VK_VERSION_MAJOR(version);
4555             minor = (int) VK_VERSION_MINOR(version);
4556         }
4557     }
4558 #endif
4559
4560     if (physical_device != NULL && glad_vkGetPhysicalDeviceProperties != NULL) {
4561         VkPhysicalDeviceProperties properties;
4562         glad_vkGetPhysicalDeviceProperties(physical_device, &properties);
4563
4564         major = (int) VK_VERSION_MAJOR(properties.apiVersion);
4565         minor = (int) VK_VERSION_MINOR(properties.apiVersion);
4566     }
```

```
4567
4568     GLAD_VK_VERSION_1_0 = (major == 1 && minor >= 0) || major > 1;
4569     GLAD_VK_VERSION_1_1 = (major == 1 && minor >= 1) || major > 1;
4570
4571     return GLAD_MAKE_VERSION(major, minor);
4572 }
4573
4574 int gladLoadVulkanUserPtr( VkPhysicalDevice physical_device, GLADuserptrloadfunc load, void *userptr) {
4575     int version;
4576
4577 #ifdef VK_VERSION_1_1
4578     glad_vkEnumerateInstanceVersion  = (PFN_vkEnumerateInstanceVersion) load(userptr,
      "vkEnumerateInstanceVersion");
4579 #endif
4580     version = glad_vk_find_core_vulkan( physical_device);
4581     if (!version) {
4582         return 0;
4583     }
4584
4585     glad_vk_load_VK_VERSION_1_0(load, userptr);
4586     glad_vk_load_VK_VERSION_1_1(load, userptr);
4587
4588     if (!glad_vk_find_extensions_vulkan( physical_device)) return 0;
4589     glad_vk_load_VK_EXT_debug_report(load, userptr);
4590     glad_vk_load_VK_KHR_surface(load, userptr);
4591     glad_vk_load_VK_KHR_swapchain(load, userptr);
4592
4593
4594     return version;
4595 }
4596
4597
4598 int gladLoadVulkan( VkPhysicalDevice physical_device, GLADloadfunc load) {
4599     return gladLoadVulkanUserPtr( physical_device, glad_vk_get_proc_from_userptr, GLAD_GNUC_EXTENSION
      (void*) load);
4600 }
4601
4602
4603
4604
4605
4606
4607 #ifdef __cplusplus
4608 }
4609 #endif
4610
4611 #endif /* GLAD_VULKAN_IMPLEMENTATION */
4612
```

27.5 linmath.h

```
1 #ifndef LINMATH_H
2 #define LINMATH_H
3
4 #include <string.h>
5 #include <math.h>
6 #include <string.h>
7
8 /* 2021-03-21 Camilla Löwy <elmindreda@elmindreda.org>
9  * - Replaced double constants with float equivalents
10  */
11
12 #ifdef LINMATH_NO_INLINE
13 #define LINMATH_H_FUNC static
14 #else
15 #define LINMATH_H_FUNC static inline
16 #endif
17
18 #define LINMATH_H_DEFINE_VEC(n) \
19 typedef float vec##n[n]; \
20 LINMATH_H_FUNC void vec##n##_add(vec##n r, vec##n const a, vec##n const b) \
21 { \
22     int i; \
23     for(i=0; i<n; ++i) \
24         r[i] = a[i] + b[i]; \
25 } \
26 LINMATH_H_FUNC void vec##n##_sub(vec##n r, vec##n const a, vec##n const b) \
27 { \
28     int i; \
29     for(i=0; i<n; ++i) \
30         r[i] = a[i] - b[i]; \
31 } \
32 LINMATH_H_FUNC void vec##n##_scale(vec##n r, vec##n const v, float const s) \
```

```
33 { \
34     int i; \
35     for(i=0; i<n; ++i) \
36         r[i] = v[i] * s; \
37 } \
38 LINMATH_H_FUNC float vec##n##_mul_inner(vec##n const a, vec##n const b) \
39 { \
40     float p = 0.f; \
41     int i; \
42     for(i=0; i<n; ++i) \
43         p += b[i]*a[i]; \
44     return p; \
45 } \
46 LINMATH_H_FUNC float vec##n##_len(vec##n const v) \
47 { \
48     return sqrtf(vec##n##_mul_inner(v,v)); \
49 } \
50 LINMATH_H_FUNC void vec##n##_norm(vec##n r, vec##n const v) \
51 { \
52     float k = 1.f / vec##n##_len(v); \
53     vec##n##_scale(r, v, k); \
54 } \
55 LINMATH_H_FUNC void vec##n##_min(vec##n r, vec##n const a, vec##n const b) \
56 { \
57     int i; \
58     for(i=0; i<n; ++i) \
59         r[i] = a[i]<b[i] ? a[i] : b[i]; \
60 } \
61 LINMATH_H_FUNC void vec##n##_max(vec##n r, vec##n const a, vec##n const b) \
62 { \
63     int i; \
64     for(i=0; i<n; ++i) \
65         r[i] = a[i]>b[i] ? a[i] : b[i]; \
66 } \
67 LINMATH_H_FUNC void vec##n##_dup(vec##n r, vec##n const src) \
68 { \
69     int i; \
70     for(i=0; i<n; ++i) \
71         r[i] = src[i]; \
72 }
73
74 LINMATH_H_DEFINE_VEC(2)
75 LINMATH_H_DEFINE_VEC(3)
76 LINMATH_H_DEFINE_VEC(4)
77
78 LINMATH_H_FUNC void vec3_mul_cross(vec3 r, vec3 const a, vec3 const b)
79 {
80     r[0] = a[1]*b[2] - a[2]*b[1];
81     r[1] = a[2]*b[0] - a[0]*b[2];
82     r[2] = a[0]*b[1] - a[1]*b[0];
83 }
84
85 LINMATH_H_FUNC void vec3_reflect(vec3 r, vec3 const v, vec3 const n)
86 {
87     float p = 2.f * vec3_mul_inner(v, n);
88     int i;
89     for(i=0;i<3;++i)
90         r[i] = v[i] - p*n[i];
91 }
92
93 LINMATH_H_FUNC void vec4_mul_cross(vec4 r, vec4 const a, vec4 const b)
94 {
95     r[0] = a[1]*b[2] - a[2]*b[1];
96     r[1] = a[2]*b[0] - a[0]*b[2];
97     r[2] = a[0]*b[1] - a[1]*b[0];
98     r[3] = 1.f;
99 }
100
101 LINMATH_H_FUNC void vec4_reflect(vec4 r, vec4 const v, vec4 const n)
102 {
103     float p  = 2.f*vec4_mul_inner(v, n);
104     int i;
105     for(i=0;i<4;++i)
106         r[i] = v[i] - p*n[i];
107 }
108
109 typedef vec4 mat4x4[4];
110 LINMATH_H_FUNC void mat4x4_identity(mat4x4 M)
111 {
112     int i, j;
113     for(i=0; i<4; ++i)
114         for(j=0; j<4; ++j)
115             M[i][j] = i==j ? 1.f : 0.f;
116 }
117 LINMATH_H_FUNC void mat4x4_dup(mat4x4 M, mat4x4 const N)
118 {
119     int i;
```

```
120     for(i=0; i<4; ++i)
121         vec4_dup(M[i], N[i]);
122 }
123 LINMATH_H_FUNC void mat4x4_row(vec4 r, mat4x4 const M, int i)
124 {
125     int k;
126     for(k=0; k<4; ++k)
127         r[k] = M[k][i];
128 }
129 LINMATH_H_FUNC void mat4x4_col(vec4 r, mat4x4 const M, int i)
130 {
131     int k;
132     for(k=0; k<4; ++k)
133         r[k] = M[i][k];
134 }
135 LINMATH_H_FUNC void mat4x4_transpose(mat4x4 M, mat4x4 const N)
136 {
137     // Note: if M and N are the same, the user has to
138     // explicitly make a copy of M and set it to N.
139     int i, j;
140     for(j=0; j<4; ++j)
141         for(i=0; i<4; ++i)
142             M[i][j] = N[j][i];
143 }
144 LINMATH_H_FUNC void mat4x4_add(mat4x4 M, mat4x4 const a, mat4x4 const b)
145 {
146     int i;
147     for(i=0; i<4; ++i)
148         vec4_add(M[i], a[i], b[i]);
149 }
150 LINMATH_H_FUNC void mat4x4_sub(mat4x4 M, mat4x4 const a, mat4x4 const b)
151 {
152     int i;
153     for(i=0; i<4; ++i)
154         vec4_sub(M[i], a[i], b[i]);
155 }
156 LINMATH_H_FUNC void mat4x4_scale(mat4x4 M, mat4x4 const a, float k)
157 {
158     int i;
159     for(i=0; i<4; ++i)
160         vec4_scale(M[i], a[i], k);
161 }
162 LINMATH_H_FUNC void mat4x4_scale_aniso(mat4x4 M, mat4x4 const a, float x, float y, float z)
163 {
164     vec4_scale(M[0], a[0], x);
165     vec4_scale(M[1], a[1], y);
166     vec4_scale(M[2], a[2], z);
167     vec4_dup(M[3], a[3]);
168 }
169 LINMATH_H_FUNC void mat4x4_mul(mat4x4 M, mat4x4 const a, mat4x4 const b)
170 {
171     mat4x4 temp;
172     int k, r, c;
173     for(c=0; c<4; ++c) for(r=0; r<4; ++r) {
174         temp[c][r] = 0.f;
175         for(k=0; k<4; ++k)
176             temp[c][r] += a[k][r] * b[c][k];
177     }
178     mat4x4_dup(M, temp);
179 }
180 LINMATH_H_FUNC void mat4x4_mul_vec4(vec4 r, mat4x4 const M, vec4 const v)
181 {
182     int i, j;
183     for(j=0; j<4; ++j) {
184         r[j] = 0.f;
185         for(i=0; i<4; ++i)
186             r[j] += M[i][j] * v[i];
187     }
188 }
189 LINMATH_H_FUNC void mat4x4_translate(mat4x4 T, float x, float y, float z)
190 {
191     mat4x4_identity(T);
192     T[3][0] = x;
193     T[3][1] = y;
194     T[3][2] = z;
195 }
196 LINMATH_H_FUNC void mat4x4_translate_in_place(mat4x4 M, float x, float y, float z)
197 {
198     vec4 t = {x, y, z, 0};
199     vec4 r;
200     int i;
201     for (i = 0; i < 4; ++i) {
202         mat4x4_row(r, M, i);
203         M[3][i] += vec4_mul_inner(r, t);
204     }
205 }
206 LINMATH_H_FUNC void mat4x4_from_vec3_mul_outer(mat4x4 M, vec3 const a, vec3 const b)
```

```
207 {
208     int i, j;
209     for(i=0; i<4; ++i) for(j=0; j<4; ++j)
210         M[i][j] = i<3 && j<3 ? a[i] * b[j] : 0.f;
211 }
212 LINMATH_H_FUNC void mat4x4_rotate(mat4x4 R, mat4x4 const M, float x, float y, float z, float angle)
213 {
214     float s = sinf(angle);
215     float c = cosf(angle);
216     vec3 u = {x, y, z};
217 
218     if(vec3_len(u) > 1e-4) {
219         vec3_norm(u, u);
220         mat4x4 T;
221         mat4x4_from_vec3_mul_outer(T, u, u);
222 
223         mat4x4 S = {
224             {    0,  u[2], -u[1], 0},
225             {-u[2],     0,  u[0], 0},
226             { u[1], -u[0],     0, 0},
227             {    0,     0,     0, 0}
228         };
229         mat4x4_scale(S, S, s);
230 
231         mat4x4 C;
232         mat4x4_identity(C);
233         mat4x4_sub(C, C, T);
234 
235         mat4x4_scale(C, C, c);
236 
237         mat4x4_add(T, T, C);
238         mat4x4_add(T, T, S);
239 
240         T[3][3] = 1.f;
241         mat4x4_mul(R, M, T);
242     } else {
243         mat4x4_dup(R, M);
244     }
245 }
246 LINMATH_H_FUNC void mat4x4_rotate_X(mat4x4 Q, mat4x4 const M, float angle)
247 {
248     float s = sinf(angle);
249     float c = cosf(angle);
250     mat4x4 R = {
251         {1.f, 0.f, 0.f, 0.f},
252         {0.f,   c,   s, 0.f},
253         {0.f,  -s,   c, 0.f},
254         {0.f, 0.f, 0.f, 1.f}
255     };
256     mat4x4_mul(Q, M, R);
257 }
258 LINMATH_H_FUNC void mat4x4_rotate_Y(mat4x4 Q, mat4x4 const M, float angle)
259 {
260     float s = sinf(angle);
261     float c = cosf(angle);
262     mat4x4 R = {
263         {   c, 0.f,  -s, 0.f},
264         { 0.f, 1.f, 0.f, 0.f},
265         {   s, 0.f,   c, 0.f},
266         { 0.f, 0.f, 0.f, 1.f}
267     };
268     mat4x4_mul(Q, M, R);
269 }
270 LINMATH_H_FUNC void mat4x4_rotate_Z(mat4x4 Q, mat4x4 const M, float angle)
271 {
272     float s = sinf(angle);
273     float c = cosf(angle);
274     mat4x4 R = {
275         {   c,   s, 0.f, 0.f},
276         {  -s,   c, 0.f, 0.f},
277         { 0.f, 0.f, 1.f, 0.f},
278         { 0.f, 0.f, 0.f, 1.f}
279     };
280     mat4x4_mul(Q, M, R);
281 }
282 LINMATH_H_FUNC void mat4x4_invert(mat4x4 T, mat4x4 const M)
283 {
284     float s[6];
285     float c[6];
286     s[0] = M[0][0]*M[1][1] - M[1][0]*M[0][1];
287     s[1] = M[0][0]*M[1][2] - M[1][0]*M[0][2];
288     s[2] = M[0][0]*M[1][3] - M[1][0]*M[0][3];
289     s[3] = M[0][1]*M[1][2] - M[1][1]*M[0][2];
290     s[4] = M[0][1]*M[1][3] - M[1][1]*M[0][3];
291     s[5] = M[0][2]*M[1][3] - M[1][2]*M[0][3];
292 
293     c[0] = M[2][0]*M[3][1] - M[3][0]*M[2][1];
```

```
294     c[1] = M[2][0]*M[3][2] - M[3][0]*M[2][2];
295     c[2] = M[2][0]*M[3][3] - M[3][0]*M[2][3];
296     c[3] = M[2][1]*M[3][2] - M[3][1]*M[2][2];
297     c[4] = M[2][1]*M[3][3] - M[3][1]*M[2][3];
298     c[5] = M[2][2]*M[3][3] - M[3][2]*M[2][3];
299
300     /* Assumes it is invertible */
301     float idet = 1.0f/( s[0]*c[5]-s[1]*c[4]+s[2]*c[3]+s[3]*c[2]-s[4]*c[1]+s[5]*c[0] );
302
303     T[0][0] = ( M[1][1] * c[5] - M[1][2] * c[4] + M[1][3] * c[3]) * idet;
304     T[0][1] = (-M[0][1] * c[5] + M[0][2] * c[4] - M[0][3] * c[3]) * idet;
305     T[0][2] = ( M[3][1] * s[5] - M[3][2] * s[4] + M[3][3] * s[3]) * idet;
306     T[0][3] = (-M[2][1] * s[5] + M[2][2] * s[4] - M[2][3] * s[3]) * idet;
307
308     T[1][0] = (-M[1][0] * c[5] + M[1][2] * c[2] - M[1][3] * c[1]) * idet;
309     T[1][1] = ( M[0][0] * c[5] - M[0][2] * c[2] + M[0][3] * c[1]) * idet;
310     T[1][2] = (-M[3][0] * s[5] + M[3][2] * s[2] - M[3][3] * s[1]) * idet;
311     T[1][3] = ( M[2][0] * s[5] - M[2][2] * s[2] + M[2][3] * s[1]) * idet;
312
313     T[2][0] = ( M[1][0] * c[4] - M[1][1] * c[2] + M[1][3] * c[0]) * idet;
314     T[2][1] = (-M[0][0] * c[4] + M[0][1] * c[2] - M[0][3] * c[0]) * idet;
315     T[2][2] = ( M[3][0] * s[4] - M[3][1] * s[2] + M[3][3] * s[0]) * idet;
316     T[2][3] = (-M[2][0] * s[4] + M[2][1] * s[2] - M[2][3] * s[0]) * idet;
317
318     T[3][0] = (-M[1][0] * c[3] + M[1][1] * c[1] - M[1][2] * c[0]) * idet;
319     T[3][1] = ( M[0][0] * c[3] - M[0][1] * c[1] + M[0][2] * c[0]) * idet;
320     T[3][2] = (-M[3][0] * s[3] + M[3][1] * s[1] - M[3][2] * s[0]) * idet;
321     T[3][3] = ( M[2][0] * s[3] - M[2][1] * s[1] + M[2][2] * s[0]) * idet;
322 }
323 LINMATH_H_FUNC void mat4x4_orthonormalize(mat4x4 R, mat4x4 const M)
324 {
325     mat4x4_dup(R, M);
326     float s = 1.f;
327     vec3 h;
328
329     vec3_norm(R[2], R[2]);
330
331     s = vec3_mul_inner(R[1], R[2]);
332     vec3_scale(h, R[2], s);
333     vec3_sub(R[1], R[1], h);
334     vec3_norm(R[1], R[1]);
335
336     s = vec3_mul_inner(R[0], R[2]);
337     vec3_scale(h, R[2], s);
338     vec3_sub(R[0], R[0], h);
339
340     s = vec3_mul_inner(R[0], R[1]);
341     vec3_scale(h, R[1], s);
342     vec3_sub(R[0], R[0], h);
343     vec3_norm(R[0], R[0]);
344 }
345
346 LINMATH_H_FUNC void mat4x4_frustum(mat4x4 M, float l, float r, float b, float t, float n, float f)
347 {
348     M[0][0] = 2.f*n/(r-l);
349     M[0][1] = M[0][2] = M[0][3] = 0.f;
350
351     M[1][1] = 2.f*n/(t-b);
352     M[1][0] = M[1][2] = M[1][3] = 0.f;
353
354     M[2][0] = (r+l)/(r-l);
355     M[2][1] = (t+b)/(t-b);
356     M[2][2] = -(f+n)/(f-n);
357     M[2][3] = -1.f;
358
359     M[3][2] = -2.f*(f*n)/(f-n);
360     M[3][0] = M[3][1] = M[3][3] = 0.f;
361 }
362 LINMATH_H_FUNC void mat4x4_ortho(mat4x4 M, float l, float r, float b, float t, float n, float f)
363 {
364     M[0][0] = 2.f/(r-l);
365     M[0][1] = M[0][2] = M[0][3] = 0.f;
366
367     M[1][1] = 2.f/(t-b);
368     M[1][0] = M[1][2] = M[1][3] = 0.f;
369
370     M[2][2] = -2.f/(f-n);
371     M[2][0] = M[2][1] = M[2][3] = 0.f;
372
373     M[3][0] = -(r+l)/(r-l);
374     M[3][1] = -(t+b)/(t-b);
375     M[3][2] = -(f+n)/(f-n);
376     M[3][3] = 1.f;
377 }
378 LINMATH_H_FUNC void mat4x4_perspective(mat4x4 m, float y_fov, float aspect, float n, float f)
379 {
380     /* NOTE: Degrees are an unhandy unit to work with.
```

```
381      * linmath.h uses radians for everything! */
382     float const a = 1.f / tanf(y_fov / 2.f);
383
384     m[0][0] = a / aspect;
385     m[0][1] = 0.f;
386     m[0][2] = 0.f;
387     m[0][3] = 0.f;
388
389     m[1][0] = 0.f;
390     m[1][1] = a;
391     m[1][2] = 0.f;
392     m[1][3] = 0.f;
393
394     m[2][0] = 0.f;
395     m[2][1] = 0.f;
396     m[2][2] = -((f + n) / (f - n));
397     m[2][3] = -1.f;
398
399     m[3][0] = 0.f;
400     m[3][1] = 0.f;
401     m[3][2] = -((2.f * f * n) / (f - n));
402     m[3][3] = 0.f;
403 }
404 LINMATH_H_FUNC void mat4x4_look_at(mat4x4 m, vec3 const eye, vec3 const center, vec3 const up)
405 {
406     /* Adapted from Android's OpenGL Matrix.java.                        */
407     /* See the OpenGL GLUT documentation for gluLookAt for a description */
408     /* of the algorithm. We implement it in a straightforward way:       */
409
410     /* TODO: The negation of of can be spared by swapping the order of
411      *       operands in the following cross products in the right way. */
412     vec3 f;
413     vec3_sub(f, center, eye);
414     vec3_norm(f, f);
415
416     vec3 s;
417     vec3_mul_cross(s, f, up);
418     vec3_norm(s, s);
419
420     vec3 t;
421     vec3_mul_cross(t, s, f);
422
423     m[0][0] =  s[0];
424     m[0][1] =  t[0];
425     m[0][2] = -f[0];
426     m[0][3] =   0.f;
427
428     m[1][0] =  s[1];
429     m[1][1] =  t[1];
430     m[1][2] = -f[1];
431     m[1][3] =   0.f;
432
433     m[2][0] =  s[2];
434     m[2][1] =  t[2];
435     m[2][2] = -f[2];
436     m[2][3] =   0.f;
437
438     m[3][0] =  0.f;
439     m[3][1] =  0.f;
440     m[3][2] =  0.f;
441     m[3][3] =  1.f;
442
443     mat4x4_translate_in_place(m, -eye[0], -eye[1], -eye[2]);
444 }
445
446 typedef float quat[4];
447 #define quat_add vec4_add
448 #define quat_sub vec4_sub
449 #define quat_norm vec4_norm
450 #define quat_scale vec4_scale
451 #define quat_mul_inner vec4_mul_inner
452
453 LINMATH_H_FUNC void quat_identity(quat q)
454 {
455     q[0] = q[1] = q[2] = 0.f;
456     q[3] = 1.f;
457 }
458 LINMATH_H_FUNC void quat_mul(quat r, quat const p, quat const q)
459 {
460     vec3 w;
461     vec3_mul_cross(r, p, q);
462     vec3_scale(w, p, q[3]);
463     vec3_add(r, r, w);
464     vec3_scale(w, q, p[3]);
465     vec3_add(r, r, w);
466     r[3] = p[3]*q[3] - vec3_mul_inner(p, q);
467 }
```

```
468 LINMATH_H_FUNC void quat_conj(quat r, quat const q)
469 {
470     int i;
471     for(i=0; i<3; ++i)
472         r[i] = -q[i];
473     r[3] = q[3];
474 }
475 LINMATH_H_FUNC void quat_rotate(quat r, float angle, vec3 const axis) {
476     vec3 axis_norm;
477     vec3_norm(axis_norm, axis);
478     float s = sinf(angle / 2);
479     float c = cosf(angle / 2);
480     vec3_scale(r, axis_norm, s);
481     r[3] = c;
482 }
483 LINMATH_H_FUNC void quat_mul_vec3(vec3 r, quat const q, vec3 const v)
484 {
485 /*
486  * Method by Fabian 'ryg' Giessen (of Farbrausch)
487 t = 2 * cross(q.xyz, v)
488 v' = v + q.w * t + cross(q.xyz, t)
489  */
490     vec3 t;
491     vec3 q_xyz = {q[0], q[1], q[2]};
492     vec3 u = {q[0], q[1], q[2]};
493
494     vec3_mul_cross(t, q_xyz, v);
495     vec3_scale(t, t, 2);
496
497     vec3_mul_cross(u, q_xyz, t);
498     vec3_scale(t, t, q[3]);
499
500     vec3_add(r, v, t);
501     vec3_add(r, r, u);
502 }
503 LINMATH_H_FUNC void mat4x4_from_quat(mat4x4 M, quat const q)
504 {
505     float a = q[3];
506     float b = q[0];
507     float c = q[1];
508     float d = q[2];
509     float a2 = a*a;
510     float b2 = b*b;
511     float c2 = c*c;
512     float d2 = d*d;
513
514     M[0][0] = a2 + b2 - c2 - d2;
515     M[0][1] = 2.f*(b*c + a*d);
516     M[0][2] = 2.f*(b*d - a*c);
517     M[0][3] = 0.f;
518
519     M[1][0] = 2*(b*c - a*d);
520     M[1][1] = a2 - b2 + c2 - d2;
521     M[1][2] = 2.f*(c*d + a*b);
522     M[1][3] = 0.f;
523
524     M[2][0] = 2.f*(b*d + a*c);
525     M[2][1] = 2.f*(c*d - a*b);
526     M[2][2] = a2 - b2 - c2 + d2;
527     M[2][3] = 0.f;
528
529     M[3][0] = M[3][1] = M[3][2] = 0.f;
530     M[3][3] = 1.f;
531 }
532
533 LINMATH_H_FUNC void mat4x4o_mul_quat(mat4x4 R, mat4x4 const M, quat const q)
534 {
535 /*  XXX: The way this is written only works for orthogonal matrices. */
536 /* TODO: Take care of non-orthogonal case. */
537     quat_mul_vec3(R[0], q, M[0]);
538     quat_mul_vec3(R[1], q, M[1]);
539     quat_mul_vec3(R[2], q, M[2]);
540
541     R[3][0] = R[3][1] = R[3][2] = 0.f;
542     R[0][3] = M[0][3];
543     R[1][3] = M[1][3];
544     R[2][3] = M[2][3];
545     R[3][3] = M[3][3];  // typically 1.0, but here we make it general
546 }
547 LINMATH_H_FUNC void quat_from_mat4x4(quat q, mat4x4 const M)
548 {
549     float r=0.f;
550     int i;
551
552     int perm[] = { 0, 1, 2, 0, 1 };
553     int *p = perm;
554
```

```
555     for(i = 0; i<3; i++) {
556         float m = M[i][i];
557         if( m < r )
558             continue;
559         m = r;
560         p = &perm[i];
561     }
562 
563     r = sqrtf(1.f + M[p[0]][p[0]] - M[p[1]][p[1]] - M[p[2]][p[2]] );
564 
565     if(r < 1e-6) {
566         q[0] = 1.f;
567         q[1] = q[2] = q[3] = 0.f;
568         return;
569     }
570 
571     q[0] = r/2.f;
572     q[1] = (M[p[0]][p[1]] - M[p[1]][p[0]])/(2.f*r);
573     q[2] = (M[p[2]][p[0]] - M[p[0]][p[2]])/(2.f*r);
574     q[3] = (M[p[2]][p[1]] - M[p[1]][p[2]])/(2.f*r);
575 }
576 
577 LINMATH_H_FUNC void mat4x4_arcball(mat4x4 R, mat4x4 const M, vec2 const _a, vec2 const _b, float s)
578 {
579     vec2 a; memcpy(a, _a, sizeof(a));
580     vec2 b; memcpy(b, _b, sizeof(b));
581 
582     float z_a = 0.f;
583     float z_b = 0.f;
584 
585     if(vec2_len(a) < 1.f) {
586         z_a = sqrtf(1.f - vec2_mul_inner(a, a));
587     } else {
588         vec2_norm(a, a);
589     }
590 
591     if(vec2_len(b) < 1.f) {
592         z_b = sqrtf(1.f - vec2_mul_inner(b, b));
593     } else {
594         vec2_norm(b, b);
595     }
596 
597     vec3 a_ = {a[0], a[1], z_a};
598     vec3 b_ = {b[0], b[1], z_b};
599 
600     vec3 c_;
601     vec3_mul_cross(c_, a_, b_);
602 
603     float const angle = acos(vec3_mul_inner(a_, b_)) * s;
604     mat4x4_rotate(R, M, c_[0], c_[1], c_[2], angle);
605 }
606 #endif
```

27.6 _mingw_dxhelper.h

```
1 
7 #if defined(_MSC_VER) && !defined(_MSC_EXTENSIONS)
8 #define NONAMELESSUNION     1
9 #endif
10 #if defined(NONAMELESSSTRUCT) && \
11    !defined(NONAMELESSUNION)
12 #define NONAMELESSUNION     1
13 #endif
14 #if defined(NONAMELESSUNION)  && \
15    !defined(NONAMELESSSTRUCT)
16 #define NONAMELESSSTRUCT    1
17 #endif
18 #if !defined(__GNU_EXTENSION)
19 #if defined(__GNUC__) || defined(__GNUG__)
20 #define __GNU_EXTENSION     __extension__
21 #else
22 #define __GNU_EXTENSION
23 #endif
24 #endif /* __extension__ */
25 
26 #ifndef __ANONYMOUS_DEFINED
27 #define __ANONYMOUS_DEFINED
28 #if defined(__GNUC__) || defined(__GNUG__)
29 #define _ANONYMOUS_UNION    __extension__
30 #define _ANONYMOUS_STRUCT   __extension__
31 #else
32 #define _ANONYMOUS_UNION
33 #define _ANONYMOUS_STRUCT
```

```
34 #endif
35 #ifndef NONAMELESSUNION
36 #define _UNION_NAME(x)
37 #define _STRUCT_NAME(x)
38 #else /* NONAMELESSUNION */
39 #define _UNION_NAME(x)  x
40 #define _STRUCT_NAME(x) x
41 #endif
42 #endif  /* __ANONYMOUS_DEFINED */
43
44 #ifndef DUMMYUNIONNAME
45 # ifdef NONAMELESSUNION
46 #  define DUMMYUNIONNAME  u
47 #  define DUMMYUNIONNAME1 u1    /* Wine uses this variant */
48 #  define DUMMYUNIONNAME2 u2
49 #  define DUMMYUNIONNAME3 u3
50 #  define DUMMYUNIONNAME4 u4
51 #  define DUMMYUNIONNAME5 u5
52 #  define DUMMYUNIONNAME6 u6
53 #  define DUMMYUNIONNAME7 u7
54 #  define DUMMYUNIONNAME8 u8
55 #  define DUMMYUNIONNAME9 u9
56 # else /* NONAMELESSUNION */
57 #  define DUMMYUNIONNAME
58 #  define DUMMYUNIONNAME1   /* Wine uses this variant */
59 #  define DUMMYUNIONNAME2
60 #  define DUMMYUNIONNAME3
61 #  define DUMMYUNIONNAME4
62 #  define DUMMYUNIONNAME5
63 #  define DUMMYUNIONNAME6
64 #  define DUMMYUNIONNAME7
65 #  define DUMMYUNIONNAME8
66 #  define DUMMYUNIONNAME9
67 # endif
68 #endif  /* DUMMYUNIONNAME */
69
70 #if !defined(DUMMYUNIONNAME1)   /* MinGW does not define this one */
71 # ifdef NONAMELESSUNION
72 #  define DUMMYUNIONNAME1 u1    /* Wine uses this variant */
73 # else
74 #  define DUMMYUNIONNAME1   /* Wine uses this variant */
75 # endif
76 #endif  /* DUMMYUNIONNAME1 */
77
78 #ifndef DUMMYSTRUCTNAME
79 # ifdef NONAMELESSUNION
80 #  define DUMMYSTRUCTNAME  s
81 #  define DUMMYSTRUCTNAME1 s1   /* Wine uses this variant */
82 #  define DUMMYSTRUCTNAME2 s2
83 #  define DUMMYSTRUCTNAME3 s3
84 #  define DUMMYSTRUCTNAME4 s4
85 #  define DUMMYSTRUCTNAME5 s5
86 # else
87 #  define DUMMYSTRUCTNAME
88 #  define DUMMYSTRUCTNAME1  /* Wine uses this variant */
89 #  define DUMMYSTRUCTNAME2
90 #  define DUMMYSTRUCTNAME3
91 #  define DUMMYSTRUCTNAME4
92 #  define DUMMYSTRUCTNAME5
93 # endif
94 #endif /* DUMMYSTRUCTNAME */
95
96 /* These are for compatibility with the Wine source tree */
97
98 #ifndef WINELIB_NAME_AW
99 # ifdef __MINGW_NAME_AW
100 #   define WINELIB_NAME_AW  __MINGW_NAME_AW
101 # else
102 #  ifdef UNICODE
103 #   define WINELIB_NAME_AW(func) func##W
104 #  else
105 #   define WINELIB_NAME_AW(func) func##A
106 #  endif
107 # endif
108 #endif  /* WINELIB_NAME_AW */
109
110 #ifndef DECL_WINELIB_TYPE_AW
111 # ifdef __MINGW_TYPEDEF_AW
112 #  define DECL_WINELIB_TYPE_AW  __MINGW_TYPEDEF_AW
113 # else
114 #  define DECL_WINELIB_TYPE_AW(type)  typedef WINELIB_NAME_AW(type) type;
115 # endif
116 #endif  /* DECL_WINELIB_TYPE_AW */
117
```

27.7 dinput.h

```
1 /*
2  * Copyright (C) the Wine project
3  *
4  * This library is free software; you can redistribute it and/or
5  * modify it under the terms of the GNU Lesser General Public
6  * License as published by the Free Software Foundation; either
7  * version 2.1 of the License, or (at your option) any later version.
8  *
9  * This library is distributed in the hope that it will be useful,
10  * but WITHOUT ANY WARRANTY; without even the implied warranty of
11  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
12  * Lesser General Public License for more details.
13  *
14  * You should have received a copy of the GNU Lesser General Public
15  * License along with this library; if not, write to the Free Software
16  * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA
17  */
18
19 #ifndef __DINPUT_INCLUDED__
20 #define __DINPUT_INCLUDED__
21
22 #define COM_NO_WINDOWS_H
23 #include <objbase.h>
24 #include <_mingw_dxhelper.h>
25
26 #ifndef DIRECTINPUT_VERSION
27 #define DIRECTINPUT_VERSION 0x0800
28 #endif
29
30 /* Classes */
31 DEFINE_GUID(CLSID_DirectInput,      0x25E609E0,0xB259,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
32 DEFINE_GUID(CLSID_DirectInputDevice,
        0x25E609E1,0xB259,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
33
34 DEFINE_GUID(CLSID_DirectInput8,     0x25E609E4,0xB259,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
35 DEFINE_GUID(CLSID_DirectInputDevice8,
        0x25E609E5,0xB259,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
36
37 /* Interfaces */
38 DEFINE_GUID(IID_IDirectInputA,      0x89521360,0xAA8A,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
39 DEFINE_GUID(IID_IDirectInputW,      0x89521361,0xAA8A,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
40 DEFINE_GUID(IID_IDirectInput2A,     0x5944E662,0xAA8A,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
41 DEFINE_GUID(IID_IDirectInput2W,     0x5944E663,0xAA8A,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
42 DEFINE_GUID(IID_IDirectInput7A,     0x9A4CB684,0x236D,0x11D3,0x8E,0x9D,0x00,0xC0,0x4F,0x68,0x44,0xAE);
43 DEFINE_GUID(IID_IDirectInput7W,     0x9A4CB685,0x236D,0x11D3,0x8E,0x9D,0x00,0xC0,0x4F,0x68,0x44,0xAE);
44 DEFINE_GUID(IID_IDirectInput8A,     0xBF798030,0x483A,0x4DA2,0xAA,0x99,0x5D,0x64,0xED,0x36,0x97,0x00);
45 DEFINE_GUID(IID_IDirectInput8W,     0xBF798031,0x483A,0x4DA2,0xAA,0x99,0x5D,0x64,0xED,0x36,0x97,0x00);
46 DEFINE_GUID(IID_IDirectInputDeviceA,
        0x5944E680,0xC92E,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
47 DEFINE_GUID(IID_IDirectInputDeviceW,
        0x5944E681,0xC92E,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
48 DEFINE_GUID(IID_IDirectInputDevice2A,
        0x5944E682,0xC92E,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
49 DEFINE_GUID(IID_IDirectInputDevice2W,
        0x5944E683,0xC92E,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
50 DEFINE_GUID(IID_IDirectInputDevice7A,
        0x57D7C6BC,0x2356,0x11D3,0x8E,0x9D,0x00,0xC0,0x4F,0x68,0x44,0xAE);
51 DEFINE_GUID(IID_IDirectInputDevice7W,
        0x57D7C6BD,0x2356,0x11D3,0x8E,0x9D,0x00,0xC0,0x4F,0x68,0x44,0xAE);
52 DEFINE_GUID(IID_IDirectInputDevice8A,
        0x54D41080,0xDC15,0x4833,0xA4,0x1B,0x74,0x8F,0x73,0xA3,0x81,0x79);
53 DEFINE_GUID(IID_IDirectInputDevice8W,
        0x54D41081,0xDC15,0x4833,0xA4,0x1B,0x74,0x8F,0x73,0xA3,0x81,0x79);
54 DEFINE_GUID(IID_IDirectInputEffect, 0xE7E1F7C0,0x88D2,0x11D0,0x9A,0xD0,0x00,0xA0,0xC9,0xA0,0x6E,0x35);
55
56 /* Predefined object types */
57 DEFINE_GUID(GUID_XAxis, 0xA36D02E0,0xC9F3,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
58 DEFINE_GUID(GUID_YAxis, 0xA36D02E1,0xC9F3,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
59 DEFINE_GUID(GUID_ZAxis, 0xA36D02E2,0xC9F3,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
60 DEFINE_GUID(GUID_RxAxis,0xA36D02F4,0xC9F3,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
61 DEFINE_GUID(GUID_RyAxis,0xA36D02F5,0xC9F3,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
62 DEFINE_GUID(GUID_RzAxis,0xA36D02E3,0xC9F3,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
63 DEFINE_GUID(GUID_Slider,0xA36D02E4,0xC9F3,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
64 DEFINE_GUID(GUID_Button,0xA36D02F0,0xC9F3,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
65 DEFINE_GUID(GUID_Key,   0x55728220,0xD33C,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
66 DEFINE_GUID(GUID_POV,   0xA36D02F2,0xC9F3,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
67 DEFINE_GUID(GUID_Unknown,0xA36D02F3,0xC9F3,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
68
69 /* Predefined product GUIDs */
70 DEFINE_GUID(GUID_SysMouse,  0x6F1D2B60,0xD5A0,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
71 DEFINE_GUID(GUID_SysKeyboard,   0x6F1D2B61,0xD5A0,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
72 DEFINE_GUID(GUID_Joystick,  0x6F1D2B70,0xD5A0,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
73 DEFINE_GUID(GUID_SysMouseEm,    0x6F1D2B80,0xD5A0,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
74 DEFINE_GUID(GUID_SysMouseEm2,   0x6F1D2B81,0xD5A0,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
75 DEFINE_GUID(GUID_SysKeyboardEm, 0x6F1D2B82,0xD5A0,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
```

```
76 DEFINE_GUID(GUID_SysKeyboardEm2,0x6F1D2B83,0xD5A0,0x11CF,0xBF,0xC7,0x44,0x45,0x53,0x54,0x00,0x00);
77
78 /* predefined forcefeedback effects */
79 DEFINE_GUID(GUID_ConstantForce, 0x13541C20,0x8E33,0x11D0,0x9A,0xD0,0x00,0xA0,0xC9,0xA0,0x6E,0x35);
80 DEFINE_GUID(GUID_RampForce, 0x13541C21,0x8E33,0x11D0,0x9A,0xD0,0x00,0xA0,0xC9,0xA0,0x6E,0x35);
81 DEFINE_GUID(GUID_Square,    0x13541C22,0x8E33,0x11D0,0x9A,0xD0,0x00,0xA0,0xC9,0xA0,0x6E,0x35);
82 DEFINE_GUID(GUID_Sine,      0x13541C23,0x8E33,0x11D0,0x9A,0xD0,0x00,0xA0,0xC9,0xA0,0x6E,0x35);
83 DEFINE_GUID(GUID_Triangle,  0x13541C24,0x8E33,0x11D0,0x9A,0xD0,0x00,0xA0,0xC9,0xA0,0x6E,0x35);
84 DEFINE_GUID(GUID_SawtoothUp,    0x13541C25,0x8E33,0x11D0,0x9A,0xD0,0x00,0xA0,0xC9,0xA0,0x6E,0x35);
85 DEFINE_GUID(GUID_SawtoothDown,  0x13541C26,0x8E33,0x11D0,0x9A,0xD0,0x00,0xA0,0xC9,0xA0,0x6E,0x35);
86 DEFINE_GUID(GUID_Spring,    0x13541C27,0x8E33,0x11D0,0x9A,0xD0,0x00,0xA0,0xC9,0xA0,0x6E,0x35);
87 DEFINE_GUID(GUID_Damper,    0x13541C28,0x8E33,0x11D0,0x9A,0xD0,0x00,0xA0,0xC9,0xA0,0x6E,0x35);
88 DEFINE_GUID(GUID_Inertia,   0x13541C29,0x8E33,0x11D0,0x9A,0xD0,0x00,0xA0,0xC9,0xA0,0x6E,0x35);
89 DEFINE_GUID(GUID_Friction,  0x13541C2A,0x8E33,0x11D0,0x9A,0xD0,0x00,0xA0,0xC9,0xA0,0x6E,0x35);
90 DEFINE_GUID(GUID_CustomForce,   0x13541C2B,0x8E33,0x11D0,0x9A,0xD0,0x00,0xA0,0xC9,0xA0,0x6E,0x35);
91
92 typedef struct IDirectInputA *LPDIRECTINPUTA;
93 typedef struct IDirectInputW *LPDIRECTINPUTW;
94 typedef struct IDirectInput2A *LPDIRECTINPUT2A;
95 typedef struct IDirectInput2W *LPDIRECTINPUT2W;
96 typedef struct IDirectInput7A *LPDIRECTINPUT7A;
97 typedef struct IDirectInput7W *LPDIRECTINPUT7W;
98 #if DIRECTINPUT_VERSION >= 0x0800
99 typedef struct IDirectInput8A *LPDIRECTINPUT8A;
100 typedef struct IDirectInput8W *LPDIRECTINPUT8W;
101 #endif /* DI8 */
102 typedef struct IDirectInputDeviceA *LPDIRECTINPUTDEVICEA;
103 typedef struct IDirectInputDeviceW *LPDIRECTINPUTDEVICEW;
104 #if DIRECTINPUT_VERSION >= 0x0500
105 typedef struct IDirectInputDevice2A *LPDIRECTINPUTDEVICE2A;
106 typedef struct IDirectInputDevice2W *LPDIRECTINPUTDEVICE2W;
107 #endif /* DI5 */
108 #if DIRECTINPUT_VERSION >= 0x0700
109 typedef struct IDirectInputDevice7A *LPDIRECTINPUTDEVICE7A;
110 typedef struct IDirectInputDevice7W *LPDIRECTINPUTDEVICE7W;
111 #endif /* DI7 */
112 #if DIRECTINPUT_VERSION >= 0x0800
113 typedef struct IDirectInputDevice8A *LPDIRECTINPUTDEVICE8A;
114 typedef struct IDirectInputDevice8W *LPDIRECTINPUTDEVICE8W;
115 #endif /* DI8 */
116 #if DIRECTINPUT_VERSION >= 0x0500
117 typedef struct IDirectInputEffect *LPDIRECTINPUTEFFECT;
118 #endif /* DI5 */
119 typedef struct SysKeyboardA *LPSYSKEYBOARDA;
120 typedef struct SysMouseA *LPSYSMOUSEA;
121
122 #define IID_IDirectInput WINELIB_NAME_AW(IID_IDirectInput)
123 #define IDirectInput WINELIB_NAME_AW(IDirectInput)
124 DECL_WINELIB_TYPE_AW(LPDIRECTINPUT)
125 #define IID_IDirectInput2 WINELIB_NAME_AW(IID_IDirectInput2)
126 #define IDirectInput2 WINELIB_NAME_AW(IDirectInput2)
127 DECL_WINELIB_TYPE_AW(LPDIRECTINPUT2)
128 #define IID_IDirectInput7 WINELIB_NAME_AW(IID_IDirectInput7)
129 #define IDirectInput7 WINELIB_NAME_AW(IDirectInput7)
130 DECL_WINELIB_TYPE_AW(LPDIRECTINPUT7)
131 #if DIRECTINPUT_VERSION >= 0x0800
132 #define IID_IDirectInput8 WINELIB_NAME_AW(IID_IDirectInput8)
133 #define IDirectInput8 WINELIB_NAME_AW(IDirectInput8)
134 DECL_WINELIB_TYPE_AW(LPDIRECTINPUT8)
135 #endif /* DI8 */
136 #define IID_IDirectInputDevice WINELIB_NAME_AW(IID_IDirectInputDevice)
137 #define IDirectInputDevice WINELIB_NAME_AW(IDirectInputDevice)
138 DECL_WINELIB_TYPE_AW(LPDIRECTINPUTDEVICE)
139 #if DIRECTINPUT_VERSION >= 0x0500
140 #define IID_IDirectInputDevice2 WINELIB_NAME_AW(IID_IDirectInputDevice2)
141 #define IDirectInputDevice2 WINELIB_NAME_AW(IDirectInputDevice2)
142 DECL_WINELIB_TYPE_AW(LPDIRECTINPUTDEVICE2)
143 #endif /* DI5 */
144 #if DIRECTINPUT_VERSION >= 0x0700
145 #define IID_IDirectInputDevice7 WINELIB_NAME_AW(IID_IDirectInputDevice7)
146 #define IDirectInputDevice7 WINELIB_NAME_AW(IDirectInputDevice7)
147 DECL_WINELIB_TYPE_AW(LPDIRECTINPUTDEVICE7)
148 #endif /* DI7 */
149 #if DIRECTINPUT_VERSION >= 0x0800
150 #define IID_IDirectInputDevice8 WINELIB_NAME_AW(IID_IDirectInputDevice8)
151 #define IDirectInputDevice8 WINELIB_NAME_AW(IDirectInputDevice8)
152 DECL_WINELIB_TYPE_AW(LPDIRECTINPUTDEVICE8)
153 #endif /* DI8 */
154
155 #define DI_OK                           S_OK
156 #define DI_NOTATTACHED                  S_FALSE
157 #define DI_BUFFEROVERFLOW               S_FALSE
158 #define DI_PROPNOEFFECT                 S_FALSE
159 #define DI_NOEFFECT                     S_FALSE
160 #define DI_POLLEDDEVICE                 ((HRESULT)0x00000002L)
161 #define DI_DOWNLOADSKIPPED              ((HRESULT)0x00000003L)
162 #define DI_EFFECTRESTARTED              ((HRESULT)0x00000004L)
```

```
163 #define DI_TRUNCATED                    ((HRESULT)0x00000008L)
164 #define DI_SETTINGSNOTSAVED             ((HRESULT)0x0000000BL)
165 #define DI_TRUNCATEDANDRESTARTED        ((HRESULT)0x0000000CL)
166 #define DI_WRITEPROTECT                 ((HRESULT)0x00000013L)
167
168 #define DIERR_OLDDIRECTINPUTVERSION     \
169     MAKE_HRESULT(SEVERITY_ERROR, FACILITY_WIN32, ERROR_OLD_WIN_VERSION)
170 #define DIERR_BETADIRECTINPUTVERSION    \
171     MAKE_HRESULT(SEVERITY_ERROR, FACILITY_WIN32, ERROR_RMODE_APP)
172 #define DIERR_BADDRIVERVER              \
173     MAKE_HRESULT(SEVERITY_ERROR, FACILITY_WIN32, ERROR_BAD_DRIVER_LEVEL)
174 #define DIERR_DEVICENOTREG              REGDB_E_CLASSNOTREG
175 #define DIERR_NOTFOUND                  \
176     MAKE_HRESULT(SEVERITY_ERROR, FACILITY_WIN32, ERROR_FILE_NOT_FOUND)
177 #define DIERR_OBJECTNOTFOUND            \
178     MAKE_HRESULT(SEVERITY_ERROR, FACILITY_WIN32, ERROR_FILE_NOT_FOUND)
179 #define DIERR_INVALIDPARAM              E_INVALIDARG
180 #define DIERR_NOINTERFACE               E_NOINTERFACE
181 #define DIERR_GENERIC                   E_FAIL
182 #define DIERR_OUTOFMEMORY               E_OUTOFMEMORY
183 #define DIERR_UNSUPPORTED               E_NOTIMPL
184 #define DIERR_NOTINITIALIZED            \
185     MAKE_HRESULT(SEVERITY_ERROR, FACILITY_WIN32, ERROR_NOT_READY)
186 #define DIERR_ALREADYINITIALIZED        \
187     MAKE_HRESULT(SEVERITY_ERROR, FACILITY_WIN32, ERROR_ALREADY_INITIALIZED)
188 #define DIERR_NOAGGREGATION             CLASS_E_NOAGGREGATION
189 #define DIERR_OTHERAPPHASPRIO           E_ACCESSDENIED
190 #define DIERR_INPUTLOST                 \
191     MAKE_HRESULT(SEVERITY_ERROR, FACILITY_WIN32, ERROR_READ_FAULT)
192 #define DIERR_ACQUIRED                  \
193     MAKE_HRESULT(SEVERITY_ERROR, FACILITY_WIN32, ERROR_BUSY)
194 #define DIERR_NOTACQUIRED               \
195     MAKE_HRESULT(SEVERITY_ERROR, FACILITY_WIN32, ERROR_INVALID_ACCESS)
196 #define DIERR_READONLY                  E_ACCESSDENIED
197 #define DIERR_HANDLEEXISTS              E_ACCESSDENIED
198 #ifndef E_PENDING
199 #define E_PENDING                       0x8000000AL
200 #endif
201 #define DIERR_INSUFFICIENTPRIVS         0x80040200L
202 #define DIERR_DEVICEFULL                0x80040201L
203 #define DIERR_MOREDATA                  0x80040202L
204 #define DIERR_NOTDOWNLOADED             0x80040203L
205 #define DIERR_HASEFFECTS                0x80040204L
206 #define DIERR_NOTEXCLUSIVEACQUIRED      0x80040205L
207 #define DIERR_INCOMPLETEEFFECT          0x80040206L
208 #define DIERR_NOTBUFFERED               0x80040207L
209 #define DIERR_EFFECTPLAYING             0x80040208L
210 #define DIERR_UNPLUGGED                 0x80040209L
211 #define DIERR_REPORTFULL                0x8004020AL
212 #define DIERR_MAPFILEFAIL               0x8004020BL
213
214 #define DIENUM_STOP                     0
215 #define DIENUM_CONTINUE                 1
216
217 #define DIEDFL_ALLDEVICES               0x00000000
218 #define DIEDFL_ATTACHEDONLY             0x00000001
219 #define DIEDFL_FORCEFEEDBACK            0x00000100
220 #define DIEDFL_INCLUDEALIASES           0x00010000
221 #define DIEDFL_INCLUDEPHANTOMS          0x00020000
222 #define DIEDFL_INCLUDEHIDDEN        0x00040000
223
224 #define DIDEVTYPE_DEVICE                1
225 #define DIDEVTYPE_MOUSE                 2
226 #define DIDEVTYPE_KEYBOARD              3
227 #define DIDEVTYPE_JOYSTICK              4
228 #define DIDEVTYPE_HID                   0x00010000
229
230 #define DI8DEVCLASS_ALL             0
231 #define DI8DEVCLASS_DEVICE          1
232 #define DI8DEVCLASS_POINTER         2
233 #define DI8DEVCLASS_KEYBOARD        3
234 #define DI8DEVCLASS_GAMECTRL        4
235
236 #define DI8DEVTYPE_DEVICE           0x11
237 #define DI8DEVTYPE_MOUSE            0x12
238 #define DI8DEVTYPE_KEYBOARD         0x13
239 #define DI8DEVTYPE_JOYSTICK         0x14
240 #define DI8DEVTYPE_GAMEPAD          0x15
241 #define DI8DEVTYPE_DRIVING          0x16
242 #define DI8DEVTYPE_FLIGHT           0x17
243 #define DI8DEVTYPE_1STPERSON        0x18
244 #define DI8DEVTYPE_DEVICECTRL       0x19
245 #define DI8DEVTYPE_SCREENPOINTER    0x1A
246 #define DI8DEVTYPE_REMOTE           0x1B
247 #define DI8DEVTYPE_SUPPLEMENTAL     0x1C
248
249 #define DIDEVTYPEMOUSE_UNKNOWN          1
```

```
250 #define DIDEVTYPEMOUSE_TRADITIONAL      2
251 #define DIDEVTYPEMOUSE_FINGERSTICK      3
252 #define DIDEVTYPEMOUSE_TOUCHPAD         4
253 #define DIDEVTYPEMOUSE_TRACKBALL        5
254
255 #define DIDEVTYPEKEYBOARD_UNKNOWN       0
256 #define DIDEVTYPEKEYBOARD_PCXT          1
257 #define DIDEVTYPEKEYBOARD_OLIVETTI      2
258 #define DIDEVTYPEKEYBOARD_PCAT          3
259 #define DIDEVTYPEKEYBOARD_PCENH         4
260 #define DIDEVTYPEKEYBOARD_NOKIA1050     5
261 #define DIDEVTYPEKEYBOARD_NOKIA9140     6
262 #define DIDEVTYPEKEYBOARD_NEC98         7
263 #define DIDEVTYPEKEYBOARD_NEC98LAPTOP   8
264 #define DIDEVTYPEKEYBOARD_NEC98106      9
265 #define DIDEVTYPEKEYBOARD_JAPAN106     10
266 #define DIDEVTYPEKEYBOARD_JAPANAX      11
267 #define DIDEVTYPEKEYBOARD_J3100        12
268
269 #define DIDEVTYPEJOYSTICK_UNKNOWN       1
270 #define DIDEVTYPEJOYSTICK_TRADITIONAL   2
271 #define DIDEVTYPEJOYSTICK_FLIGHTSTICK   3
272 #define DIDEVTYPEJOYSTICK_GAMEPAD       4
273 #define DIDEVTYPEJOYSTICK_RUDDER        5
274 #define DIDEVTYPEJOYSTICK_WHEEL         6
275 #define DIDEVTYPEJOYSTICK_HEADTRACKER   7
276
277 #define DI8DEVTYPEMOUSE_UNKNOWN                  1
278 #define DI8DEVTYPEMOUSE_TRADITIONAL              2
279 #define DI8DEVTYPEMOUSE_FINGERSTICK              3
280 #define DI8DEVTYPEMOUSE_TOUCHPAD                 4
281 #define DI8DEVTYPEMOUSE_TRACKBALL                5
282 #define DI8DEVTYPEMOUSE_ABSOLUTE                 6
283
284 #define DI8DEVTYPEKEYBOARD_UNKNOWN               0
285 #define DI8DEVTYPEKEYBOARD_PCXT                  1
286 #define DI8DEVTYPEKEYBOARD_OLIVETTI              2
287 #define DI8DEVTYPEKEYBOARD_PCAT                  3
288 #define DI8DEVTYPEKEYBOARD_PCENH                 4
289 #define DI8DEVTYPEKEYBOARD_NOKIA1050             5
290 #define DI8DEVTYPEKEYBOARD_NOKIA9140             6
291 #define DI8DEVTYPEKEYBOARD_NEC98                 7
292 #define DI8DEVTYPEKEYBOARD_NEC98LAPTOP           8
293 #define DI8DEVTYPEKEYBOARD_NEC98106              9
294 #define DI8DEVTYPEKEYBOARD_JAPAN106             10
295 #define DI8DEVTYPEKEYBOARD_JAPANAX              11
296 #define DI8DEVTYPEKEYBOARD_J3100                12
297
298 #define DI8DEVTYPE_LIMITEDGAMESUBTYPE            1
299
300 #define DI8DEVTYPEJOYSTICK_LIMITED               DI8DEVTYPE_LIMITEDGAMESUBTYPE
301 #define DI8DEVTYPEJOYSTICK_STANDARD              2
302
303 #define DI8DEVTYPEGAMEPAD_LIMITED                DI8DEVTYPE_LIMITEDGAMESUBTYPE
304 #define DI8DEVTYPEGAMEPAD_STANDARD               2
305 #define DI8DEVTYPEGAMEPAD_TILT                   3
306
307 #define DI8DEVTYPEDRIVING_LIMITED                DI8DEVTYPE_LIMITEDGAMESUBTYPE
308 #define DI8DEVTYPEDRIVING_COMBINEDPEDALS         2
309 #define DI8DEVTYPEDRIVING_DUALPEDALS             3
310 #define DI8DEVTYPEDRIVING_THREEPEDALS            4
311 #define DI8DEVTYPEDRIVING_HANDHELD               5
312
313 #define DI8DEVTYPEFLIGHT_LIMITED                 DI8DEVTYPE_LIMITEDGAMESUBTYPE
314 #define DI8DEVTYPEFLIGHT_STICK                   2
315 #define DI8DEVTYPEFLIGHT_YOKE                    3
316 #define DI8DEVTYPEFLIGHT_RC                      4
317
318 #define DI8DEVTYPE1STPERSON_LIMITED              DI8DEVTYPE_LIMITEDGAMESUBTYPE
319 #define DI8DEVTYPE1STPERSON_UNKNOWN              2
320 #define DI8DEVTYPE1STPERSON_SIXDOF               3
321 #define DI8DEVTYPE1STPERSON_SHOOTER              4
322
323 #define DI8DEVTYPESCREENPTR_UNKNOWN              2
324 #define DI8DEVTYPESCREENPTR_LIGHTGUN             3
325 #define DI8DEVTYPESCREENPTR_LIGHTPEN             4
326 #define DI8DEVTYPESCREENPTR_TOUCH                5
327
328 #define DI8DEVTYPEREMOTE_UNKNOWN                 2
329
330 #define DI8DEVTYPEDEVICECTRL_UNKNOWN             2
331 #define DI8DEVTYPEDEVICECTRL_COMMSSELECTION      3
332 #define DI8DEVTYPEDEVICECTRL_COMMSSELECTION_HARDWIRED 4
333
334 #define DI8DEVTYPESUPPLEMENTAL_UNKNOWN           2
335 #define DI8DEVTYPESUPPLEMENTAL_2NDHANDCONTROLLER 3
336 #define DI8DEVTYPESUPPLEMENTAL_HEADTRACKER       4
```

```
337 #define DI8DEVTYPESUPPLEMENTAL_HANDTRACKER          5
338 #define DI8DEVTYPESUPPLEMENTAL_SHIFTSTICKGATE       6
339 #define DI8DEVTYPESUPPLEMENTAL_SHIFTER              7
340 #define DI8DEVTYPESUPPLEMENTAL_THROTTLE             8
341 #define DI8DEVTYPESUPPLEMENTAL_SPLITTHROTTLE        9
342 #define DI8DEVTYPESUPPLEMENTAL_COMBINEDPEDALS      10
343 #define DI8DEVTYPESUPPLEMENTAL_DUALPEDALS          11
344 #define DI8DEVTYPESUPPLEMENTAL_THREEPEDALS         12
345 #define DI8DEVTYPESUPPLEMENTAL_RUDDERPEDALS        13
346
347 #define GET_DIDEVICE_TYPE(dwDevType)     LOBYTE(dwDevType)
348 #define GET_DIDEVICE_SUBTYPE(dwDevType)  HIBYTE(dwDevType)
349
350 typedef struct DIDEVICEOBJECTINSTANCE_DX3A {
351     DWORD   dwSize;
352     GUID    guidType;
353     DWORD   dwOfs;
354     DWORD   dwType;
355     DWORD   dwFlags;
356     CHAR    tszName[MAX_PATH];
357 } DIDEVICEOBJECTINSTANCE_DX3A, *LPDIDEVICEOBJECTINSTANCE_DX3A;
358 typedef const DIDEVICEOBJECTINSTANCE_DX3A *LPCDIDEVICEOBJECTINSTANCE_DX3A;
359 typedef struct DIDEVICEOBJECTINSTANCE_DX3W {
360     DWORD   dwSize;
361     GUID    guidType;
362     DWORD   dwOfs;
363     DWORD   dwType;
364     DWORD   dwFlags;
365     WCHAR   tszName[MAX_PATH];
366 } DIDEVICEOBJECTINSTANCE_DX3W, *LPDIDEVICEOBJECTINSTANCE_DX3W;
367 typedef const DIDEVICEOBJECTINSTANCE_DX3W *LPCDIDEVICEOBJECTINSTANCE_DX3W;
368
369 DECL_WINELIB_TYPE_AW(DIDEVICEOBJECTINSTANCE_DX3)
370 DECL_WINELIB_TYPE_AW(LPDIDEVICEOBJECTINSTANCE_DX3)
371 DECL_WINELIB_TYPE_AW(LPCDIDEVICEOBJECTINSTANCE_DX3)
372
373 typedef struct DIDEVICEOBJECTINSTANCEA {
374     DWORD   dwSize;
375     GUID    guidType;
376     DWORD   dwOfs;
377     DWORD   dwType;
378     DWORD   dwFlags;
379     CHAR    tszName[MAX_PATH];
380 #if(DIRECTINPUT_VERSION >= 0x0500)
381     DWORD   dwFFMaxForce;
382     DWORD   dwFFForceResolution;
383     WORD    wCollectionNumber;
384     WORD    wDesignatorIndex;
385     WORD    wUsagePage;
386     WORD    wUsage;
387     DWORD   dwDimension;
388     WORD    wExponent;
389     WORD    wReserved;
390 #endif /* DIRECTINPUT_VERSION >= 0x0500 */
391 } DIDEVICEOBJECTINSTANCEA, *LPDIDEVICEOBJECTINSTANCEA;
392 typedef const DIDEVICEOBJECTINSTANCEA *LPCDIDEVICEOBJECTINSTANCEA;
393
394 typedef struct DIDEVICEOBJECTINSTANCEW {
395     DWORD   dwSize;
396     GUID    guidType;
397     DWORD   dwOfs;
398     DWORD   dwType;
399     DWORD   dwFlags;
400     WCHAR   tszName[MAX_PATH];
401 #if(DIRECTINPUT_VERSION >= 0x0500)
402     DWORD   dwFFMaxForce;
403     DWORD   dwFFForceResolution;
404     WORD    wCollectionNumber;
405     WORD    wDesignatorIndex;
406     WORD    wUsagePage;
407     WORD    wUsage;
408     DWORD   dwDimension;
409     WORD    wExponent;
410     WORD    wReserved;
411 #endif /* DIRECTINPUT_VERSION >= 0x0500 */
412 } DIDEVICEOBJECTINSTANCEW, *LPDIDEVICEOBJECTINSTANCEW;
413 typedef const DIDEVICEOBJECTINSTANCEW *LPCDIDEVICEOBJECTINSTANCEW;
414
415 DECL_WINELIB_TYPE_AW(DIDEVICEOBJECTINSTANCE)
416 DECL_WINELIB_TYPE_AW(LPDIDEVICEOBJECTINSTANCE)
417 DECL_WINELIB_TYPE_AW(LPCDIDEVICEOBJECTINSTANCE)
418
419 typedef struct DIDEVICEINSTANCE_DX3A {
420     DWORD   dwSize;
421     GUID    guidInstance;
422     GUID    guidProduct;
423     DWORD   dwDevType;
```

```
424     CHAR    tszInstanceName[MAX_PATH];
425     CHAR    tszProductName[MAX_PATH];
426 } DIDEVICEINSTANCE_DX3A, *LPDIDEVICEINSTANCE_DX3A;
427 typedef const DIDEVICEINSTANCE_DX3A *LPCDIDEVICEINSTANCE_DX3A;
428 typedef struct DIDEVICEINSTANCE_DX3W {
429     DWORD   dwSize;
430     GUID    guidInstance;
431     GUID    guidProduct;
432     DWORD   dwDevType;
433     WCHAR   tszInstanceName[MAX_PATH];
434     WCHAR   tszProductName[MAX_PATH];
435 } DIDEVICEINSTANCE_DX3W, *LPDIDEVICEINSTANCE_DX3W;
436 typedef const DIDEVICEINSTANCE_DX3W *LPCDIDEVICEINSTANCE_DX3W;
437
438 DECL_WINELIB_TYPE_AW(DIDEVICEINSTANCE_DX3)
439 DECL_WINELIB_TYPE_AW(LPDIDEVICEINSTANCE_DX3)
440 DECL_WINELIB_TYPE_AW(LPCDIDEVICEINSTANCE_DX3)
441
442 typedef struct DIDEVICEINSTANCEA {
443     DWORD   dwSize;
444     GUID    guidInstance;
445     GUID    guidProduct;
446     DWORD   dwDevType;
447     CHAR    tszInstanceName[MAX_PATH];
448     CHAR    tszProductName[MAX_PATH];
449 #if(DIRECTINPUT_VERSION >= 0x0500)
450     GUID    guidFFDriver;
451     WORD    wUsagePage;
452     WORD    wUsage;
453 #endif /* DIRECTINPUT_VERSION >= 0x0500 */
454 } DIDEVICEINSTANCEA, *LPDIDEVICEINSTANCEA;
455 typedef const DIDEVICEINSTANCEA *LPCDIDEVICEINSTANCEA;
456
457 typedef struct DIDEVICEINSTANCEW {
458     DWORD   dwSize;
459     GUID    guidInstance;
460     GUID    guidProduct;
461     DWORD   dwDevType;
462     WCHAR   tszInstanceName[MAX_PATH];
463     WCHAR   tszProductName[MAX_PATH];
464 #if(DIRECTINPUT_VERSION >= 0x0500)
465     GUID    guidFFDriver;
466     WORD    wUsagePage;
467     WORD    wUsage;
468 #endif /* DIRECTINPUT_VERSION >= 0x0500 */
469 } DIDEVICEINSTANCEW, *LPDIDEVICEINSTANCEW;
470 typedef const DIDEVICEINSTANCEW *LPCDIDEVICEINSTANCEW;
471
472 DECL_WINELIB_TYPE_AW(DIDEVICEINSTANCE)
473 DECL_WINELIB_TYPE_AW(LPDIDEVICEINSTANCE)
474 DECL_WINELIB_TYPE_AW(LPCDIDEVICEINSTANCE)
475
476 typedef BOOL (CALLBACK *LPDIENUMDEVICESCALLBACKA)(LPCDIDEVICEINSTANCEA,LPVOID);
477 typedef BOOL (CALLBACK *LPDIENUMDEVICESCALLBACKW)(LPCDIDEVICEINSTANCEW,LPVOID);
478 DECL_WINELIB_TYPE_AW(LPDIENUMDEVICESCALLBACK)
479
480 #define DIEDBS_MAPPEDPRI1       0x00000001
481 #define DIEDBS_MAPPEDPRI2       0x00000002
482 #define DIEDBS_RECENTDEVICE     0x00000010
483 #define DIEDBS_NEWDEVICE        0x00000020
484
485 #define DIEDBSFL_ATTACHEDONLY       0x00000000
486 #define DIEDBSFL_THISUSER       0x00000010
487 #define DIEDBSFL_FORCEFEEDBACK      DIEDFL_FORCEFEEDBACK
488 #define DIEDBSFL_AVAILABLEDEVICES   0x00001000
489 #define DIEDBSFL_MULTIMICEKEYBOARDS 0x00002000
490 #define DIEDBSFL_NONGAMINGDEVICES   0x00004000
491 #define DIEDBSFL_VALID          0x00007110
492
493 #if DIRECTINPUT_VERSION >= 0x0800
494 typedef BOOL (CALLBACK
       *LPDIENUMDEVICESBYSEMANTICSCBA)(LPCDIDEVICEINSTANCEA,LPDIRECTINPUTDEVICE8A,DWORD,DWORD,LPVOID);
495 typedef BOOL (CALLBACK
       *LPDIENUMDEVICESBYSEMANTICSCBW)(LPCDIDEVICEINSTANCEW,LPDIRECTINPUTDEVICE8W,DWORD,DWORD,LPVOID);
496 DECL_WINELIB_TYPE_AW(LPDIENUMDEVICESBYSEMANTICSCB)
497 #endif
498
499 typedef BOOL (CALLBACK *LPDICONFIGUREDEVICESCALLBACK)(LPUNKNOWN,LPVOID);
500
501 typedef BOOL (CALLBACK *LPDIENUMDEVICEOBJECTSCALLBACKA)(LPCDIDEVICEOBJECTINSTANCEA,LPVOID);
502 typedef BOOL (CALLBACK *LPDIENUMDEVICEOBJECTSCALLBACKW)(LPCDIDEVICEOBJECTINSTANCEW,LPVOID);
503 DECL_WINELIB_TYPE_AW(LPDIENUMDEVICEOBJECTSCALLBACK)
504
505 #if DIRECTINPUT_VERSION >= 0x0500
506 typedef BOOL (CALLBACK *LPDIENUMCREATEDEFFECTOBJECTSCALLBACK)(LPDIRECTINPUTEFFECT, LPVOID);
507 #endif
508
```

```
509 #define DIK_ESCAPE          0x01
510 #define DIK_1               0x02
511 #define DIK_2               0x03
512 #define DIK_3               0x04
513 #define DIK_4               0x05
514 #define DIK_5               0x06
515 #define DIK_6               0x07
516 #define DIK_7               0x08
517 #define DIK_8               0x09
518 #define DIK_9               0x0A
519 #define DIK_0               0x0B
520 #define DIK_MINUS           0x0C    /* - on main keyboard */
521 #define DIK_EQUALS          0x0D
522 #define DIK_BACK            0x0E    /* backspace */
523 #define DIK_TAB             0x0F
524 #define DIK_Q               0x10
525 #define DIK_W               0x11
526 #define DIK_E               0x12
527 #define DIK_R               0x13
528 #define DIK_T               0x14
529 #define DIK_Y               0x15
530 #define DIK_U               0x16
531 #define DIK_I               0x17
532 #define DIK_O               0x18
533 #define DIK_P               0x19
534 #define DIK_LBRACKET        0x1A
535 #define DIK_RBRACKET        0x1B
536 #define DIK_RETURN          0x1C    /* Enter on main keyboard */
537 #define DIK_LCONTROL        0x1D
538 #define DIK_A               0x1E
539 #define DIK_S               0x1F
540 #define DIK_D               0x20
541 #define DIK_F               0x21
542 #define DIK_G               0x22
543 #define DIK_H               0x23
544 #define DIK_J               0x24
545 #define DIK_K               0x25
546 #define DIK_L               0x26
547 #define DIK_SEMICOLON       0x27
548 #define DIK_APOSTROPHE      0x28
549 #define DIK_GRAVE           0x29    /* accent grave */
550 #define DIK_LSHIFT          0x2A
551 #define DIK_BACKSLASH       0x2B
552 #define DIK_Z               0x2C
553 #define DIK_X               0x2D
554 #define DIK_C               0x2E
555 #define DIK_V               0x2F
556 #define DIK_B               0x30
557 #define DIK_N               0x31
558 #define DIK_M               0x32
559 #define DIK_COMMA           0x33
560 #define DIK_PERIOD          0x34    /* . on main keyboard */
561 #define DIK_SLASH           0x35    /* / on main keyboard */
562 #define DIK_RSHIFT          0x36
563 #define DIK_MULTIPLY        0x37    /* * on numeric keypad */
564 #define DIK_LMENU           0x38    /* left Alt */
565 #define DIK_SPACE           0x39
566 #define DIK_CAPITAL         0x3A
567 #define DIK_F1              0x3B
568 #define DIK_F2              0x3C
569 #define DIK_F3              0x3D
570 #define DIK_F4              0x3E
571 #define DIK_F5              0x3F
572 #define DIK_F6              0x40
573 #define DIK_F7              0x41
574 #define DIK_F8              0x42
575 #define DIK_F9              0x43
576 #define DIK_F10             0x44
577 #define DIK_NUMLOCK         0x45
578 #define DIK_SCROLL          0x46    /* Scroll Lock */
579 #define DIK_NUMPAD7         0x47
580 #define DIK_NUMPAD8         0x48
581 #define DIK_NUMPAD9         0x49
582 #define DIK_SUBTRACT        0x4A    /* - on numeric keypad */
583 #define DIK_NUMPAD4         0x4B
584 #define DIK_NUMPAD5         0x4C
585 #define DIK_NUMPAD6         0x4D
586 #define DIK_ADD             0x4E    /* + on numeric keypad */
587 #define DIK_NUMPAD1         0x4F
588 #define DIK_NUMPAD2         0x50
589 #define DIK_NUMPAD3         0x51
590 #define DIK_NUMPAD0         0x52
591 #define DIK_DECIMAL         0x53    /* . on numeric keypad */
592 #define DIK_OEM_102         0x56    /* < > | on UK/Germany keyboards */
593 #define DIK_F11             0x57
594 #define DIK_F12             0x58
595 #define DIK_F13             0x64    /*                     (NEC PC98) */
```

```
596 #define DIK_F14             0x65    /*                     (NEC PC98) */
597 #define DIK_F15             0x66    /*                     (NEC PC98) */
598 #define DIK_KANA            0x70    /* (Japanese keyboard)            */
599 #define DIK_ABNT_C1         0x73    /* / ? on Portugese (Brazilian) keyboards */
600 #define DIK_CONVERT         0x79    /* (Japanese keyboard)            */
601 #define DIK_NOCONVERT       0x7B    /* (Japanese keyboard)            */
602 #define DIK_YEN             0x7D    /* (Japanese keyboard)            */
603 #define DIK_ABNT_C2         0x7E    /* Numpad . on Portugese (Brazilian) keyboards */
604 #define DIK_NUMPADEQUALS    0x8D    /* = on numeric keypad (NEC PC98) */
605 #define DIK_CIRCUMFLEX      0x90    /* (Japanese keyboard)            */
606 #define DIK_AT              0x91    /*                     (NEC PC98) */
607 #define DIK_COLON           0x92    /*                     (NEC PC98) */
608 #define DIK_UNDERLINE       0x93    /*                     (NEC PC98) */
609 #define DIK_KANJI           0x94    /* (Japanese keyboard)            */
610 #define DIK_STOP            0x95    /*                     (NEC PC98) */
611 #define DIK_AX              0x96    /*                     (Japan AX) */
612 #define DIK_UNLABELED       0x97    /*                        (J3100) */
613 #define DIK_NEXTTRACK       0x99    /* Next Track */
614 #define DIK_NUMPADENTER     0x9C    /* Enter on numeric keypad */
615 #define DIK_RCONTROL        0x9D
616 #define DIK_MUTE        0xA0    /* Mute */
617 #define DIK_CALCULATOR      0xA1    /* Calculator */
618 #define DIK_PLAYPAUSE       0xA2    /* Play / Pause */
619 #define DIK_MEDIASTOP       0xA4    /* Media Stop */
620 #define DIK_VOLUMEDOWN      0xAE    /* Volume - */
621 #define DIK_VOLUMEUP        0xB0    /* Volume + */
622 #define DIK_WEBHOME         0xB2    /* Web home */
623 #define DIK_NUMPADCOMMA     0xB3    /* , on numeric keypad (NEC PC98) */
624 #define DIK_DIVIDE          0xB5    /* / on numeric keypad */
625 #define DIK_SYSRQ           0xB7
626 #define DIK_RMENU           0xB8    /* right Alt */
627 #define DIK_PAUSE           0xC5    /* Pause */
628 #define DIK_HOME            0xC7    /* Home on arrow keypad */
629 #define DIK_UP              0xC8    /* UpArrow on arrow keypad */
630 #define DIK_PRIOR           0xC9    /* PgUp on arrow keypad */
631 #define DIK_LEFT            0xCB    /* LeftArrow on arrow keypad */
632 #define DIK_RIGHT           0xCD    /* RightArrow on arrow keypad */
633 #define DIK_END             0xCF    /* End on arrow keypad */
634 #define DIK_DOWN            0xD0    /* DownArrow on arrow keypad */
635 #define DIK_NEXT            0xD1    /* PgDn on arrow keypad */
636 #define DIK_INSERT          0xD2    /* Insert on arrow keypad */
637 #define DIK_DELETE          0xD3    /* Delete on arrow keypad */
638 #define DIK_LWIN            0xDB    /* Left Windows key */
639 #define DIK_RWIN            0xDC    /* Right Windows key */
640 #define DIK_APPS            0xDD    /* AppMenu key */
641 #define DIK_POWER           0xDE
642 #define DIK_SLEEP           0xDF
643 #define DIK_WAKE            0xE3    /* System Wake */
644 #define DIK_WEBSEARCH       0xE5    /* Web Search */
645 #define DIK_WEBFAVORITES    0xE6    /* Web Favorites */
646 #define DIK_WEBREFRESH      0xE7    /* Web Refresh */
647 #define DIK_WEBSTOP         0xE8    /* Web Stop */
648 #define DIK_WEBFORWARD      0xE9    /* Web Forward */
649 #define DIK_WEBBACK         0xEA    /* Web Back */
650 #define DIK_MYCOMPUTER      0xEB    /* My Computer */
651 #define DIK_MAIL            0xEC    /* Mail */
652 #define DIK_MEDIASELECT     0xED    /* Media Select */
653
654 #define DIK_BACKSPACE       DIK_BACK            /* backspace */
655 #define DIK_NUMPADSTAR      DIK_MULTIPLY        /* * on numeric keypad */
656 #define DIK_LALT            DIK_LMENU           /* left Alt */
657 #define DIK_CAPSLOCK        DIK_CAPITAL         /* CapsLock */
658 #define DIK_NUMPADMINUS     DIK_SUBTRACT        /* - on numeric keypad */
659 #define DIK_NUMPADPLUS      DIK_ADD             /* + on numeric keypad */
660 #define DIK_NUMPADPERIOD    DIK_DECIMAL         /* . on numeric keypad */
661 #define DIK_NUMPADSLASH     DIK_DIVIDE          /* / on numeric keypad */
662 #define DIK_RALT            DIK_RMENU           /* right Alt */
663 #define DIK_UPARROW         DIK_UP              /* UpArrow on arrow keypad */
664 #define DIK_PGUP            DIK_PRIOR           /* PgUp on arrow keypad */
665 #define DIK_LEFTARROW       DIK_LEFT            /* LeftArrow on arrow keypad */
666 #define DIK_RIGHTARROW      DIK_RIGHT           /* RightArrow on arrow keypad */
667 #define DIK_DOWNARROW       DIK_DOWN            /* DownArrow on arrow keypad */
668 #define DIK_PGDN            DIK_NEXT            /* PgDn on arrow keypad */
669
670 #define DIDFT_ALL       0x00000000
671 #define DIDFT_RELAXIS       0x00000001
672 #define DIDFT_ABSAXIS       0x00000002
673 #define DIDFT_AXIS      0x00000003
674 #define DIDFT_PSHBUTTON     0x00000004
675 #define DIDFT_TGLBUTTON     0x00000008
676 #define DIDFT_BUTTON        0x0000000C
677 #define DIDFT_POV       0x00000010
678 #define DIDFT_COLLECTION    0x00000040
679 #define DIDFT_NODATA        0x00000080
680 #define DIDFT_ANYINSTANCE   0x00FFFF00
681 #define DIDFT_INSTANCEMASK  DIDFT_ANYINSTANCE
682 #define DIDFT_MAKEINSTANCE(n)    ((WORD)(n) « 8)
```

```
683 #define DIDFT_GETTYPE(n)    LOBYTE(n)
684 #define DIDFT_GETINSTANCE(n)    LOWORD((n) » 8)
685 #define DIDFT_FFACTUATOR    0x01000000
686 #define DIDFT_FFEFFECTTRIGGER   0x02000000
687 #if DIRECTINPUT_VERSION >= 0x050a
688 #define DIDFT_OUTPUT        0x10000000
689 #define DIDFT_VENDORDEFINED 0x04000000
690 #define DIDFT_ALIAS     0x08000000
691 #endif /* DI5a */
692 #ifndef DIDFT_OPTIONAL
693 #define DIDFT_OPTIONAL      0x80000000
694 #endif
695 #define DIDFT_ENUMCOLLECTION(n) ((WORD)(n) « 8)
696 #define DIDFT_NOCOLLECTION  0x00FFFF00
697
698 #define DIDF_ABSAXIS        0x00000001
699 #define DIDF_RELAXIS        0x00000002
700
701 #define DIGDD_PEEK      0x00000001
702
703 #define DISEQUENCE_COMPARE(dwSq1,cmp,dwSq2) ((int)((dwSq1) - (dwSq2)) cmp 0)
704
705 typedef struct DIDEVICEOBJECTDATA_DX3 {
706     DWORD   dwOfs;
707     DWORD   dwData;
708     DWORD   dwTimeStamp;
709     DWORD   dwSequence;
710 } DIDEVICEOBJECTDATA_DX3,*LPDIDEVICEOBJECTDATA_DX3;
711 typedef const DIDEVICEOBJECTDATA_DX3 *LPCDIDEVICEOBJECTDATA_DX3;
712
713 typedef struct DIDEVICEOBJECTDATA {
714     DWORD   dwOfs;
715     DWORD   dwData;
716     DWORD   dwTimeStamp;
717     DWORD   dwSequence;
718 #if(DIRECTINPUT_VERSION >= 0x0800)
719     UINT_PTR    uAppData;
720 #endif /* DIRECTINPUT_VERSION >= 0x0800 */
721 } DIDEVICEOBJECTDATA, *LPDIDEVICEOBJECTDATA;
722 typedef const DIDEVICEOBJECTDATA *LPCDIDEVICEOBJECTDATA;
723
724 typedef struct _DIOBJECTDATAFORMAT {
725     const GUID *pguid;
726     DWORD   dwOfs;
727     DWORD   dwType;
728     DWORD   dwFlags;
729 } DIOBJECTDATAFORMAT, *LPDIOBJECTDATAFORMAT;
730 typedef const DIOBJECTDATAFORMAT *LPCDIOBJECTDATAFORMAT;
731
732 typedef struct _DIDATAFORMAT {
733     DWORD           dwSize;
734     DWORD           dwObjSize;
735     DWORD           dwFlags;
736     DWORD           dwDataSize;
737     DWORD           dwNumObjs;
738     LPDIOBJECTDATAFORMAT    rgodf;
739 } DIDATAFORMAT, *LPDIDATAFORMAT;
740 typedef const DIDATAFORMAT *LPCDIDATAFORMAT;
741
742 #if DIRECTINPUT_VERSION >= 0x0500
743 #define DIDOI_FFACTUATOR    0x00000001
744 #define DIDOI_FFEFFECTTRIGGER   0x00000002
745 #define DIDOI_POLLED        0x00008000
746 #define DIDOI_ASPECTPOSITION    0x00000100
747 #define DIDOI_ASPECTVELOCITY    0x00000200
748 #define DIDOI_ASPECTACCEL   0x00000300
749 #define DIDOI_ASPECTFORCE   0x00000400
750 #define DIDOI_ASPECTMASK    0x00000F00
751 #endif /* DI5 */
752 #if DIRECTINPUT_VERSION >= 0x050a
753 #define DIDOI_GUIDISUSAGE   0x00010000
754 #endif /* DI5a */
755
756 typedef struct DIPROPHEADER {
757     DWORD   dwSize;
758     DWORD   dwHeaderSize;
759     DWORD   dwObj;
760     DWORD   dwHow;
761 } DIPROPHEADER,*LPDIPROPHEADER;
762 typedef const DIPROPHEADER *LPCDIPROPHEADER;
763
764 #define DIPH_DEVICE 0
765 #define DIPH_BYOFFSET   1
766 #define DIPH_BYID   2
767 #if DIRECTINPUT_VERSION >= 0x050a
768 #define DIPH_BYUSAGE    3
769
```

```
770 #define DIMAKEUSAGEDWORD(UsagePage, Usage) (DWORD)MAKELONG(Usage, UsagePage)
771 #endif /* DI5a */
772
773 typedef struct DIPROPDWORD {
774     DIPROPHEADER    diph;
775     DWORD       dwData;
776 } DIPROPDWORD, *LPDIPROPDWORD;
777 typedef const DIPROPDWORD *LPCDIPROPDWORD;
778
779 typedef struct DIPROPRANGE {
780     DIPROPHEADER    diph;
781     LONG        lMin;
782     LONG        lMax;
783 } DIPROPRANGE, *LPDIPROPRANGE;
784 typedef const DIPROPRANGE *LPCDIPROPRANGE;
785
786 #define DIPROPRANGE_NOMIN   ((LONG)0x80000000)
787 #define DIPROPRANGE_NOMAX   ((LONG)0x7FFFFFFF)
788
789 #if DIRECTINPUT_VERSION >= 0x050a
790 typedef struct DIPROPCAL {
791     DIPROPHEADER diph;
792     LONG    lMin;
793     LONG    lCenter;
794     LONG    lMax;
795 } DIPROPCAL, *LPDIPROPCAL;
796 typedef const DIPROPCAL *LPCDIPROPCAL;
797
798 typedef struct DIPROPCALPOV {
799     DIPROPHEADER    diph;
800     LONG        lMin[5];
801     LONG        lMax[5];
802 } DIPROPCALPOV, *LPDIPROPCALPOV;
803 typedef const DIPROPCALPOV *LPCDIPROPCALPOV;
804
805 typedef struct DIPROPGUIDANDPATH {
806     DIPROPHEADER diph;
807     GUID    guidClass;
808     WCHAR   wszPath[MAX_PATH];
809 } DIPROPGUIDANDPATH, *LPDIPROPGUIDANDPATH;
810 typedef const DIPROPGUIDANDPATH *LPCDIPROPGUIDANDPATH;
811
812 typedef struct DIPROPSTRING {
813         DIPROPHEADER diph;
814         WCHAR       wsz[MAX_PATH];
815 } DIPROPSTRING, *LPDIPROPSTRING;
816 typedef const DIPROPSTRING *LPCDIPROPSTRING;
817 #endif /* DI5a */
818
819 #if DIRECTINPUT_VERSION >= 0x0800
820 typedef struct DIPROPPOINTER {
821     DIPROPHEADER diph;
822     UINT_PTR    uData;
823 } DIPROPPOINTER, *LPDIPROPPOINTER;
824 typedef const DIPROPPOINTER *LPCDIPROPPOINTER;
825 #endif /* DI8 */
826
827 /* special property GUIDs */
828 #ifdef __cplusplus
829 #define MAKEDIPROP(prop)    (*(const GUID *)(prop))
830 #else
831 #define MAKEDIPROP(prop)    ((REFGUID)(prop))
832 #endif
833 #define DIPROP_BUFFERSIZE   MAKEDIPROP(1)
834 #define DIPROP_AXISMODE     MAKEDIPROP(2)
835
836 #define DIPROPAXISMODE_ABS  0
837 #define DIPROPAXISMODE_REL  1
838
839 #define DIPROP_GRANULARITY  MAKEDIPROP(3)
840 #define DIPROP_RANGE        MAKEDIPROP(4)
841 #define DIPROP_DEADZONE     MAKEDIPROP(5)
842 #define DIPROP_SATURATION   MAKEDIPROP(6)
843 #define DIPROP_FFGAIN       MAKEDIPROP(7)
844 #define DIPROP_FFLOAD       MAKEDIPROP(8)
845 #define DIPROP_AUTOCENTER   MAKEDIPROP(9)
846
847 #define DIPROPAUTOCENTER_OFF    0
848 #define DIPROPAUTOCENTER_ON 1
849
850 #define DIPROP_CALIBRATIONMODE  MAKEDIPROP(10)
851
852 #define DIPROPCALIBRATIONMODE_COOKED    0
853 #define DIPROPCALIBRATIONMODE_RAW   1
854
855 #if DIRECTINPUT_VERSION >= 0x050a
856 #define DIPROP_CALIBRATION  MAKEDIPROP(11)
```

```
857 #define DIPROP_GUIDANDPATH  MAKEDIPROP(12)
858 #define DIPROP_INSTANCENAME MAKEDIPROP(13)
859 #define DIPROP_PRODUCTNAME  MAKEDIPROP(14)
860 #endif
861
862 #if DIRECTINPUT_VERSION >= 0x5B2
863 #define DIPROP_JOYSTICKID   MAKEDIPROP(15)
864 #define DIPROP_GETPORTDISPLAYNAME   MAKEDIPROP(16)
865 #endif
866
867 #if DIRECTINPUT_VERSION >= 0x0700
868 #define DIPROP_PHYSICALRANGE    MAKEDIPROP(18)
869 #define DIPROP_LOGICALRANGE MAKEDIPROP(19)
870 #endif
871
872 #if(DIRECTINPUT_VERSION >= 0x0800)
873 #define DIPROP_KEYNAME      MAKEDIPROP(20)
874 #define DIPROP_CPOINTS      MAKEDIPROP(21)
875 #define DIPROP_APPDATA      MAKEDIPROP(22)
876 #define DIPROP_SCANCODE     MAKEDIPROP(23)
877 #define DIPROP_VIDPID       MAKEDIPROP(24)
878 #define DIPROP_USERNAME     MAKEDIPROP(25)
879 #define DIPROP_TYPENAME     MAKEDIPROP(26)
880
881 #define MAXCPOINTSNUM       8
882
883 typedef struct _CPOINT {
884     LONG    lP;
885     DWORD   dwLog;
886 } CPOINT, *PCPOINT;
887
888 typedef struct DIPROPCPOINTS {
889     DIPROPHEADER diph;
890     DWORD   dwCPointsNum;
891     CPOINT    cp[MAXCPOINTSNUM];
892 } DIPROPCPOINTS, *LPDIPROPCPOINTS;
893 typedef const DIPROPCPOINTS *LPCDIPROPCPOINTS;
894 #endif /* DI8 */
895
896
897 typedef struct DIDEVCAPS_DX3 {
898     DWORD   dwSize;
899     DWORD   dwFlags;
900     DWORD   dwDevType;
901     DWORD   dwAxes;
902     DWORD   dwButtons;
903     DWORD   dwPOVs;
904 } DIDEVCAPS_DX3, *LPDIDEVCAPS_DX3;
905
906 typedef struct DIDEVCAPS {
907     DWORD   dwSize;
908     DWORD   dwFlags;
909     DWORD   dwDevType;
910     DWORD   dwAxes;
911     DWORD   dwButtons;
912     DWORD   dwPOVs;
913 #if(DIRECTINPUT_VERSION >= 0x0500)
914     DWORD   dwFFSamplePeriod;
915     DWORD   dwFFMinTimeResolution;
916     DWORD   dwFirmwareRevision;
917     DWORD   dwHardwareRevision;
918     DWORD   dwFFDriverVersion;
919 #endif /* DIRECTINPUT_VERSION >= 0x0500 */
920 } DIDEVCAPS,*LPDIDEVCAPS;
921
922 #define DIDC_ATTACHED       0x00000001
923 #define DIDC_POLLEDDEVICE   0x00000002
924 #define DIDC_EMULATED       0x00000004
925 #define DIDC_POLLEDDATAFORMAT   0x00000008
926 #define DIDC_FORCEFEEDBACK  0x00000100
927 #define DIDC_FFATTACK       0x00000200
928 #define DIDC_FFFADE     0x00000400
929 #define DIDC_SATURATION     0x00000800
930 #define DIDC_POSNEGCOEFFICIENTS 0x00001000
931 #define DIDC_POSNEGSATURATION   0x00002000
932 #define DIDC_DEADBAND       0x00004000
933 #define DIDC_STARTDELAY     0x00008000
934 #define DIDC_ALIAS      0x00010000
935 #define DIDC_PHANTOM        0x00020000
936 #define DIDC_HIDDEN     0x00040000
937
938
939 /* SetCooperativeLevel dwFlags */
940 #define DISCL_EXCLUSIVE     0x00000001
941 #define DISCL_NONEXCLUSIVE  0x00000002
942 #define DISCL_FOREGROUND    0x00000004
943 #define DISCL_BACKGROUND    0x00000008
```

```
944 #define DISCL_NOWINKEY          0x00000010
945
946 #if (DIRECTINPUT_VERSION >= 0x0500)
947 /* Device FF flags */
948 #define DISFFC_RESET            0x00000001
949 #define DISFFC_STOPALL          0x00000002
950 #define DISFFC_PAUSE            0x00000004
951 #define DISFFC_CONTINUE         0x00000008
952 #define DISFFC_SETACTUATORSON   0x00000010
953 #define DISFFC_SETACTUATORSOFF  0x00000020
954
955 #define DIGFFS_EMPTY            0x00000001
956 #define DIGFFS_STOPPED          0x00000002
957 #define DIGFFS_PAUSED           0x00000004
958 #define DIGFFS_ACTUATORSON      0x00000010
959 #define DIGFFS_ACTUATORSOFF     0x00000020
960 #define DIGFFS_POWERON          0x00000040
961 #define DIGFFS_POWEROFF         0x00000080
962 #define DIGFFS_SAFETYSWITCHON   0x00000100
963 #define DIGFFS_SAFETYSWITCHOFF  0x00000200
964 #define DIGFFS_USERFFSWITCHON   0x00000400
965 #define DIGFFS_USERFFSWITCHOFF  0x00000800
966 #define DIGFFS_DEVICELOST       0x80000000
967
968 /* Effect flags */
969 #define DIEFT_ALL       0x00000000
970
971 #define DIEFT_CONSTANTFORCE 0x00000001
972 #define DIEFT_RAMPFORCE     0x00000002
973 #define DIEFT_PERIODIC      0x00000003
974 #define DIEFT_CONDITION     0x00000004
975 #define DIEFT_CUSTOMFORCE   0x00000005
976 #define DIEFT_HARDWARE      0x000000FF
977 #define DIEFT_FFATTACK      0x00000200
978 #define DIEFT_FFFADE        0x00000400
979 #define DIEFT_SATURATION    0x00000800
980 #define DIEFT_POSNEGCOEFFICIENTS 0x00001000
981 #define DIEFT_POSNEGSATURATION  0x00002000
982 #define DIEFT_DEADBAND      0x00004000
983 #define DIEFT_STARTDELAY    0x00008000
984 #define DIEFT_GETTYPE(n)    LOBYTE(n)
985
986 #define DIEFF_OBJECTIDS         0x00000001
987 #define DIEFF_OBJECTOFFSETS     0x00000002
988 #define DIEFF_CARTESIAN         0x00000010
989 #define DIEFF_POLAR             0x00000020
990 #define DIEFF_SPHERICAL         0x00000040
991
992 #define DIEP_DURATION           0x00000001
993 #define DIEP_SAMPLEPERIOD       0x00000002
994 #define DIEP_GAIN               0x00000004
995 #define DIEP_TRIGGERBUTTON      0x00000008
996 #define DIEP_TRIGGERREPEATINTERVAL 0x00000010
997 #define DIEP_AXES               0x00000020
998 #define DIEP_DIRECTION          0x00000040
999 #define DIEP_ENVELOPE           0x00000080
1000 #define DIEP_TYPESPECIFICPARAMS 0x00000100
1001 #if(DIRECTINPUT_VERSION >= 0x0600)
1002 #define DIEP_STARTDELAY         0x00000200
1003 #define DIEP_ALLPARAMS_DX5      0x000001FF
1004 #define DIEP_ALLPARAMS          0x000003FF
1005 #else
1006 #define DIEP_ALLPARAMS          0x000001FF
1007 #endif /* DIRECTINPUT_VERSION >= 0x0600 */
1008 #define DIEP_START              0x20000000
1009 #define DIEP_NORESTART          0x40000000
1010 #define DIEP_NODOWNLOAD         0x80000000
1011 #define DIEB_NOTRIGGER          0xFFFFFFFF
1012
1013 #define DIES_SOLO               0x00000001
1014 #define DIES_NODOWNLOAD         0x80000000
1015
1016 #define DIEGES_PLAYING          0x00000001
1017 #define DIEGES_EMULATED         0x00000002
1018
1019 #define DI_DEGREES      100
1020 #define DI_FFNOMINALMAX     10000
1021 #define DI_SECONDS      1000000
1022
1023 typedef struct DICONSTANTFORCE {
1024     LONG            lMagnitude;
1025 } DICONSTANTFORCE, *LPDICONSTANTFORCE;
1026 typedef const DICONSTANTFORCE *LPCDICONSTANTFORCE;
1027
1028 typedef struct DIRAMPFORCE {
1029     LONG            lStart;
1030     LONG            lEnd;
```

```
1031 } DIRAMPFORCE, *LPDIRAMPFORCE;
1032 typedef const DIRAMPFORCE *LPCDIRAMPFORCE;
1033
1034 typedef struct DIPERIODIC {
1035     DWORD           dwMagnitude;
1036     LONG            lOffset;
1037     DWORD           dwPhase;
1038     DWORD           dwPeriod;
1039 } DIPERIODIC, *LPDIPERIODIC;
1040 typedef const DIPERIODIC *LPCDIPERIODIC;
1041
1042 typedef struct DICONDITION {
1043     LONG            lOffset;
1044     LONG            lPositiveCoefficient;
1045     LONG            lNegativeCoefficient;
1046     DWORD           dwPositiveSaturation;
1047     DWORD           dwNegativeSaturation;
1048     LONG            lDeadBand;
1049 } DICONDITION, *LPDICONDITION;
1050 typedef const DICONDITION *LPCDICONDITION;
1051
1052 typedef struct DICUSTOMFORCE {
1053     DWORD           cChannels;
1054     DWORD           dwSamplePeriod;
1055     DWORD           cSamples;
1056     LPLONG          rglForceData;
1057 } DICUSTOMFORCE, *LPDICUSTOMFORCE;
1058 typedef const DICUSTOMFORCE *LPCDICUSTOMFORCE;
1059
1060 typedef struct DIENVELOPE {
1061     DWORD           dwSize;
1062     DWORD           dwAttackLevel;
1063     DWORD           dwAttackTime;
1064     DWORD           dwFadeLevel;
1065     DWORD           dwFadeTime;
1066 } DIENVELOPE, *LPDIENVELOPE;
1067 typedef const DIENVELOPE *LPCDIENVELOPE;
1068
1069 typedef struct DIEFFECT_DX5 {
1070     DWORD           dwSize;
1071     DWORD           dwFlags;
1072     DWORD           dwDuration;
1073     DWORD           dwSamplePeriod;
1074     DWORD           dwGain;
1075     DWORD           dwTriggerButton;
1076     DWORD           dwTriggerRepeatInterval;
1077     DWORD           cAxes;
1078     LPDWORD         rgdwAxes;
1079     LPLONG          rglDirection;
1080     LPDIENVELOPE        lpEnvelope;
1081     DWORD           cbTypeSpecificParams;
1082     LPVOID          lpvTypeSpecificParams;
1083 } DIEFFECT_DX5, *LPDIEFFECT_DX5;
1084 typedef const DIEFFECT_DX5 *LPCDIEFFECT_DX5;
1085
1086 typedef struct DIEFFECT {
1087     DWORD           dwSize;
1088     DWORD           dwFlags;
1089     DWORD           dwDuration;
1090     DWORD           dwSamplePeriod;
1091     DWORD           dwGain;
1092     DWORD           dwTriggerButton;
1093     DWORD           dwTriggerRepeatInterval;
1094     DWORD           cAxes;
1095     LPDWORD         rgdwAxes;
1096     LPLONG          rglDirection;
1097     LPDIENVELOPE        lpEnvelope;
1098     DWORD           cbTypeSpecificParams;
1099     LPVOID          lpvTypeSpecificParams;
1100 #if(DIRECTINPUT_VERSION >= 0x0600)
1101     DWORD           dwStartDelay;
1102 #endif /* DIRECTINPUT_VERSION >= 0x0600 */
1103 } DIEFFECT, *LPDIEFFECT;
1104 typedef const DIEFFECT *LPCDIEFFECT;
1105 typedef DIEFFECT DIEFFECT_DX6;
1106 typedef LPDIEFFECT LPDIEFFECT_DX6;
1107
1108 typedef struct DIEFFECTINFOA {
1109     DWORD           dwSize;
1110     GUID            guid;
1111     DWORD           dwEffType;
1112     DWORD           dwStaticParams;
1113     DWORD           dwDynamicParams;
1114     CHAR            tszName[MAX_PATH];
1115 } DIEFFECTINFOA, *LPDIEFFECTINFOA;
1116 typedef const DIEFFECTINFOA *LPCDIEFFECTINFOA;
1117
```

```
1118 typedef struct DIEFFECTINFOW {
1119     DWORD           dwSize;
1120     GUID            guid;
1121     DWORD           dwEffType;
1122     DWORD           dwStaticParams;
1123     DWORD           dwDynamicParams;
1124     WCHAR           tszName[MAX_PATH];
1125 } DIEFFECTINFOW, *LPDIEFFECTINFOW;
1126 typedef const DIEFFECTINFOW *LPCDIEFFECTINFOW;
1127
1128 DECL_WINELIB_TYPE_AW(DIEFFECTINFO)
1129 DECL_WINELIB_TYPE_AW(LPDIEFFECTINFO)
1130 DECL_WINELIB_TYPE_AW(LPCDIEFFECTINFO)
1131
1132 typedef BOOL (CALLBACK *LPDIENUMEFFECTSCALLBACKA)(LPCDIEFFECTINFOA, LPVOID);
1133 typedef BOOL (CALLBACK *LPDIENUMEFFECTSCALLBACKW)(LPCDIEFFECTINFOW, LPVOID);
1134
1135 typedef struct DIEFFESCAPE {
1136     DWORD   dwSize;
1137     DWORD   dwCommand;
1138     LPVOID  lpvInBuffer;
1139     DWORD   cbInBuffer;
1140     LPVOID  lpvOutBuffer;
1141     DWORD   cbOutBuffer;
1142 } DIEFFESCAPE, *LPDIEFFESCAPE;
1143
1144 typedef struct DIJOYSTATE {
1145     LONG    lX;
1146     LONG    lY;
1147     LONG    lZ;
1148     LONG    lRx;
1149     LONG    lRy;
1150     LONG    lRz;
1151     LONG    rglSlider[2];
1152     DWORD   rgdwPOV[4];
1153     BYTE    rgbButtons[32];
1154 } DIJOYSTATE, *LPDIJOYSTATE;
1155
1156 typedef struct DIJOYSTATE2 {
1157     LONG    lX;
1158     LONG    lY;
1159     LONG    lZ;
1160     LONG    lRx;
1161     LONG    lRy;
1162     LONG    lRz;
1163     LONG    rglSlider[2];
1164     DWORD   rgdwPOV[4];
1165     BYTE    rgbButtons[128];
1166     LONG    lVX;        /* 'v' as in velocity */
1167     LONG    lVY;
1168     LONG    lVZ;
1169     LONG    lVRx;
1170     LONG    lVRy;
1171     LONG    lVRz;
1172     LONG    rglVSlider[2];
1173     LONG    lAX;        /* 'a' as in acceleration */
1174     LONG    lAY;
1175     LONG    lAZ;
1176     LONG    lARx;
1177     LONG    lARy;
1178     LONG    lARz;
1179     LONG    rglASlider[2];
1180     LONG    lFX;        /* 'f' as in force */
1181     LONG    lFY;
1182     LONG    lFZ;
1183     LONG    lFRx;       /* 'fr' as in rotational force aka torque */
1184     LONG    lFRy;
1185     LONG    lFRz;
1186     LONG    rglFSlider[2];
1187 } DIJOYSTATE2, *LPDIJOYSTATE2;
1188
1189 #define DIJOFS_X        FIELD_OFFSET(DIJOYSTATE, lX)
1190 #define DIJOFS_Y        FIELD_OFFSET(DIJOYSTATE, lY)
1191 #define DIJOFS_Z        FIELD_OFFSET(DIJOYSTATE, lZ)
1192 #define DIJOFS_RX       FIELD_OFFSET(DIJOYSTATE, lRx)
1193 #define DIJOFS_RY       FIELD_OFFSET(DIJOYSTATE, lRy)
1194 #define DIJOFS_RZ       FIELD_OFFSET(DIJOYSTATE, lRz)
1195 #define DIJOFS_SLIDER(n)    (FIELD_OFFSET(DIJOYSTATE, rglSlider) + \
1196                                                   (n) * sizeof(LONG))
1197 #define DIJOFS_POV(n)       (FIELD_OFFSET(DIJOYSTATE, rgdwPOV) + \
1198                                                   (n) * sizeof(DWORD))
1199 #define DIJOFS_BUTTON(n)    (FIELD_OFFSET(DIJOYSTATE, rgbButtons) + (n))
1200 #define DIJOFS_BUTTON0      DIJOFS_BUTTON(0)
1201 #define DIJOFS_BUTTON1      DIJOFS_BUTTON(1)
1202 #define DIJOFS_BUTTON2      DIJOFS_BUTTON(2)
1203 #define DIJOFS_BUTTON3      DIJOFS_BUTTON(3)
1204 #define DIJOFS_BUTTON4      DIJOFS_BUTTON(4)
```

```
1205 #define DIJOFS_BUTTON5      DIJOFS_BUTTON(5)
1206 #define DIJOFS_BUTTON6      DIJOFS_BUTTON(6)
1207 #define DIJOFS_BUTTON7      DIJOFS_BUTTON(7)
1208 #define DIJOFS_BUTTON8      DIJOFS_BUTTON(8)
1209 #define DIJOFS_BUTTON9      DIJOFS_BUTTON(9)
1210 #define DIJOFS_BUTTON10     DIJOFS_BUTTON(10)
1211 #define DIJOFS_BUTTON11     DIJOFS_BUTTON(11)
1212 #define DIJOFS_BUTTON12     DIJOFS_BUTTON(12)
1213 #define DIJOFS_BUTTON13     DIJOFS_BUTTON(13)
1214 #define DIJOFS_BUTTON14     DIJOFS_BUTTON(14)
1215 #define DIJOFS_BUTTON15     DIJOFS_BUTTON(15)
1216 #define DIJOFS_BUTTON16     DIJOFS_BUTTON(16)
1217 #define DIJOFS_BUTTON17     DIJOFS_BUTTON(17)
1218 #define DIJOFS_BUTTON18     DIJOFS_BUTTON(18)
1219 #define DIJOFS_BUTTON19     DIJOFS_BUTTON(19)
1220 #define DIJOFS_BUTTON20     DIJOFS_BUTTON(20)
1221 #define DIJOFS_BUTTON21     DIJOFS_BUTTON(21)
1222 #define DIJOFS_BUTTON22     DIJOFS_BUTTON(22)
1223 #define DIJOFS_BUTTON23     DIJOFS_BUTTON(23)
1224 #define DIJOFS_BUTTON24     DIJOFS_BUTTON(24)
1225 #define DIJOFS_BUTTON25     DIJOFS_BUTTON(25)
1226 #define DIJOFS_BUTTON26     DIJOFS_BUTTON(26)
1227 #define DIJOFS_BUTTON27     DIJOFS_BUTTON(27)
1228 #define DIJOFS_BUTTON28     DIJOFS_BUTTON(28)
1229 #define DIJOFS_BUTTON29     DIJOFS_BUTTON(29)
1230 #define DIJOFS_BUTTON30     DIJOFS_BUTTON(30)
1231 #define DIJOFS_BUTTON31     DIJOFS_BUTTON(31)
1232 #endif /* DIRECTINPUT_VERSION >= 0x0500 */
1233
1234 /* DInput 7 structures, types */
1235 #if(DIRECTINPUT_VERSION >= 0x0700)
1236 typedef struct DIFILEEFFECT {
1237   DWORD       dwSize;
1238   GUID        GuidEffect;
1239   LPCDIEFFECT lpDiEffect;
1240   CHAR        szFriendlyName[MAX_PATH];
1241 } DIFILEEFFECT, *LPDIFILEEFFECT;
1242
1243 typedef const DIFILEEFFECT *LPCDIFILEEFFECT;
1244 typedef BOOL (CALLBACK *LPDIENUMEFFECTSINFILECALLBACK)(LPCDIFILEEFFECT , LPVOID);
1245 #endif /* DIRECTINPUT_VERSION >= 0x0700 */
1246
1247 /* DInput 8 structures and types */
1248 #if DIRECTINPUT_VERSION >= 0x0800
1249 typedef struct _DIACTIONA {
1250     UINT_PTR    uAppData;
1251     DWORD       dwSemantic;
1252     DWORD       dwFlags;
1253     __GNU_EXTENSION union {
1254         LPCSTR  lptszActionName;
1255         UINT    uResIdString;
1256     } DUMMYUNIONNAME;
1257     GUID        guidInstance;
1258     DWORD       dwObjID;
1259     DWORD       dwHow;
1260 } DIACTIONA, *LPDIACTIONA;
1261 typedef const DIACTIONA *LPCDIACTIONA;
1262
1263 typedef struct _DIACTIONW {
1264     UINT_PTR    uAppData;
1265     DWORD       dwSemantic;
1266     DWORD       dwFlags;
1267     __GNU_EXTENSION union {
1268         LPCWSTR lptszActionName;
1269         UINT    uResIdString;
1270     } DUMMYUNIONNAME;
1271     GUID        guidInstance;
1272     DWORD       dwObjID;
1273     DWORD       dwHow;
1274 } DIACTIONW, *LPDIACTIONW;
1275 typedef const DIACTIONW *LPCDIACTIONW;
1276
1277 DECL_WINELIB_TYPE_AW(DIACTION)
1278 DECL_WINELIB_TYPE_AW(LPDIACTION)
1279 DECL_WINELIB_TYPE_AW(LPCDIACTION)
1280
1281 #define DIA_FORCEFEEDBACK   0x00000001
1282 #define DIA_APPMAPPED       0x00000002
1283 #define DIA_APPNOMAP        0x00000004
1284 #define DIA_NORANGE     0x00000008
1285 #define DIA_APPFIXED        0x00000010
1286
1287 #define DIAH_UNMAPPED       0x00000000
1288 #define DIAH_USERCONFIG     0x00000001
1289 #define DIAH_APPREQUESTED   0x00000002
1290 #define DIAH_HWAPP      0x00000004
1291 #define DIAH_HWDEFAULT      0x00000008
```

```
1292 #define DIAH_DEFAULT        0x00000020
1293 #define DIAH_ERROR      0x80000000
1294
1295 typedef struct _DIACTIONFORMATA {
1296     DWORD       dwSize;
1297     DWORD       dwActionSize;
1298     DWORD       dwDataSize;
1299     DWORD       dwNumActions;
1300     LPDIACTIONA  rgoAction;
1301     GUID        guidActionMap;
1302     DWORD       dwGenre;
1303     DWORD       dwBufferSize;
1304     LONG        lAxisMin;
1305     LONG        lAxisMax;
1306     HINSTANCE   hInstString;
1307     FILETIME    ftTimeStamp;
1308     DWORD       dwCRC;
1309     CHAR        tszActionMap[MAX_PATH];
1310 } DIACTIONFORMATA, *LPDIACTIONFORMATA;
1311 typedef const DIACTIONFORMATA *LPCDIACTIONFORMATA;
1312
1313 typedef struct _DIACTIONFORMATW {
1314     DWORD       dwSize;
1315     DWORD       dwActionSize;
1316     DWORD       dwDataSize;
1317     DWORD       dwNumActions;
1318     LPDIACTIONW  rgoAction;
1319     GUID        guidActionMap;
1320     DWORD       dwGenre;
1321     DWORD       dwBufferSize;
1322     LONG        lAxisMin;
1323     LONG        lAxisMax;
1324     HINSTANCE   hInstString;
1325     FILETIME    ftTimeStamp;
1326     DWORD       dwCRC;
1327     WCHAR       tszActionMap[MAX_PATH];
1328 } DIACTIONFORMATW, *LPDIACTIONFORMATW;
1329 typedef const DIACTIONFORMATW *LPCDIACTIONFORMATW;
1330
1331 DECL_WINELIB_TYPE_AW(DIACTIONFORMAT)
1332 DECL_WINELIB_TYPE_AW(LPDIACTIONFORMAT)
1333 DECL_WINELIB_TYPE_AW(LPCDIACTIONFORMAT)
1334
1335 #define DIAFTS_NEWDEVICELOW 0xFFFFFFFF
1336 #define DIAFTS_NEWDEVICEHIGH    0xFFFFFFFF
1337 #define DIAFTS_UNUSEDDEVICELOW  0x00000000
1338 #define DIAFTS_UNUSEDDEVICEHIGH 0x00000000
1339
1340 #define DIDBAM_DEFAULT      0x00000000
1341 #define DIDBAM_PRESERVE     0x00000001
1342 #define DIDBAM_INITIALIZE   0x00000002
1343 #define DIDBAM_HWDEFAULTS   0x00000004
1344
1345 #define DIDSAM_DEFAULT      0x00000000
1346 #define DIDSAM_NOUSER       0x00000001
1347 #define DIDSAM_FORCESAVE    0x00000002
1348
1349 #define DICD_DEFAULT        0x00000000
1350 #define DICD_EDIT       0x00000001
1351
1352 #ifndef D3DCOLOR_DEFINED
1353 typedef DWORD D3DCOLOR;
1354 #define D3DCOLOR_DEFINED
1355 #endif
1356
1357 typedef struct _DICOLORSET {
1358     DWORD       dwSize;
1359     D3DCOLOR    cTextFore;
1360     D3DCOLOR    cTextHighlight;
1361     D3DCOLOR    cCalloutLine;
1362     D3DCOLOR    cCalloutHighlight;
1363     D3DCOLOR    cBorder;
1364     D3DCOLOR    cControlFill;
1365     D3DCOLOR    cHighlightFill;
1366     D3DCOLOR    cAreaFill;
1367 } DICOLORSET, *LPDICOLORSET;
1368 typedef const DICOLORSET *LPCDICOLORSET;
1369
1370 typedef struct _DICONFIGUREDEVICESPARAMSA {
1371     DWORD           dwSize;
1372     DWORD           dwcUsers;
1373     LPSTR           lptszUserNames;
1374     DWORD           dwcFormats;
1375     LPDIACTIONFORMATA  lprgFormats;
1376     HWND            hwnd;
1377     DICOLORSET        dics;
1378     LPUNKNOWN       lpUnkDDSTarget;
```

```
1379 } DICONFIGUREDEVICESPARAMSA, *LPDICONFIGUREDEVICESPARAMSA;
1380 typedef const DICONFIGUREDEVICESPARAMSA *LPCDICONFIGUREDEVICESPARAMSA;
1381
1382 typedef struct _DICONFIGUREDEVICESPARAMSW {
1383     DWORD           dwSize;
1384     DWORD           dwcUsers;
1385     LPWSTR          lptszUserNames;
1386     DWORD           dwcFormats;
1387     LPDIACTIONFORMATW  lprgFormats;
1388     HWND            hwnd;
1389     DICOLORSET        dics;
1390     LPUNKNOWN       lpUnkDDSTarget;
1391 } DICONFIGUREDEVICESPARAMSW, *LPDICONFIGUREDEVICESPARAMSW;
1392 typedef const DICONFIGUREDEVICESPARAMSW *LPCDICONFIGUREDEVICESPARAMSW;
1393
1394 DECL_WINELIB_TYPE_AW(DICONFIGUREDEVICESPARAMS)
1395 DECL_WINELIB_TYPE_AW(LPDICONFIGUREDEVICESPARAMS)
1396 DECL_WINELIB_TYPE_AW(LPCDICONFIGUREDEVICESPARAMS)
1397
1398 #define DIDIFT_CONFIGURATION    0x00000001
1399 #define DIDIFT_OVERLAY      0x00000002
1400
1401 #define DIDAL_CENTERED      0x00000000
1402 #define DIDAL_LEFTALIGNED   0x00000001
1403 #define DIDAL_RIGHTALIGNED  0x00000002
1404 #define DIDAL_MIDDLE        0x00000000
1405 #define DIDAL_TOPALIGNED    0x00000004
1406 #define DIDAL_BOTTOMALIGNED 0x00000008
1407
1408 typedef struct _DIDEVICEIMAGEINFOA {
1409     CHAR    tszImagePath[MAX_PATH];
1410     DWORD   dwFlags;
1411     DWORD   dwViewID;
1412     RECT    rcOverlay;
1413     DWORD   dwObjID;
1414     DWORD   dwcValidPts;
1415     POINT   rgptCalloutLine[5];
1416     RECT    rcCalloutRect;
1417     DWORD   dwTextAlign;
1418 } DIDEVICEIMAGEINFOA, *LPDIDEVICEIMAGEINFOA;
1419 typedef const DIDEVICEIMAGEINFOA *LPCDIDEVICEIMAGEINFOA;
1420
1421 typedef struct _DIDEVICEIMAGEINFOW {
1422     WCHAR   tszImagePath[MAX_PATH];
1423     DWORD   dwFlags;
1424     DWORD   dwViewID;
1425     RECT    rcOverlay;
1426     DWORD   dwObjID;
1427     DWORD   dwcValidPts;
1428     POINT   rgptCalloutLine[5];
1429     RECT    rcCalloutRect;
1430     DWORD   dwTextAlign;
1431 } DIDEVICEIMAGEINFOW, *LPDIDEVICEIMAGEINFOW;
1432 typedef const DIDEVICEIMAGEINFOW *LPCDIDEVICEIMAGEINFOW;
1433
1434 DECL_WINELIB_TYPE_AW(DIDEVICEIMAGEINFO)
1435 DECL_WINELIB_TYPE_AW(LPDIDEVICEIMAGEINFO)
1436 DECL_WINELIB_TYPE_AW(LPCDIDEVICEIMAGEINFO)
1437
1438 typedef struct _DIDEVICEIMAGEINFOHEADERA {
1439     DWORD   dwSize;
1440     DWORD   dwSizeImageInfo;
1441     DWORD   dwcViews;
1442     DWORD   dwcButtons;
1443     DWORD   dwcAxes;
1444     DWORD   dwcPOVs;
1445     DWORD   dwBufferSize;
1446     DWORD   dwBufferUsed;
1447     LPDIDEVICEIMAGEINFOA    lprgImageInfoArray;
1448 } DIDEVICEIMAGEINFOHEADERA, *LPDIDEVICEIMAGEINFOHEADERA;
1449 typedef const DIDEVICEIMAGEINFOHEADERA *LPCDIDEVICEIMAGEINFOHEADERA;
1450
1451 typedef struct _DIDEVICEIMAGEINFOHEADERW {
1452     DWORD   dwSize;
1453     DWORD   dwSizeImageInfo;
1454     DWORD   dwcViews;
1455     DWORD   dwcButtons;
1456     DWORD   dwcAxes;
1457     DWORD   dwcPOVs;
1458     DWORD   dwBufferSize;
1459     DWORD   dwBufferUsed;
1460     LPDIDEVICEIMAGEINFOW    lprgImageInfoArray;
1461 } DIDEVICEIMAGEINFOHEADERW, *LPDIDEVICEIMAGEINFOHEADERW;
1462 typedef const DIDEVICEIMAGEINFOHEADERW *LPCDIDEVICEIMAGEINFOHEADERW;
1463
1464 DECL_WINELIB_TYPE_AW(DIDEVICEIMAGEINFOHEADER)
1465 DECL_WINELIB_TYPE_AW(LPDIDEVICEIMAGEINFOHEADER)
```

```
1466 DECL_WINELIB_TYPE_AW(LPCDIDEVICEIMAGEINFOHEADER)
1467
1468 #endif /* DI8 */
1469
1470
1471 /*****************************************************************************
1472  * IDirectInputEffect interface
1473  */
1474 #if (DIRECTINPUT_VERSION >= 0x0500)
1475 #undef INTERFACE
1476 #define INTERFACE IDirectInputEffect
1477 DECLARE_INTERFACE_(IDirectInputEffect,IUnknown)
1478 {
1479     /*** IUnknown methods ***/
1480     STDMETHOD_(HRESULT,QueryInterface)(THIS_ REFIID riid, void** ppvObject) PURE;
1481     STDMETHOD_(ULONG,AddRef)(THIS) PURE;
1482     STDMETHOD_(ULONG,Release)(THIS) PURE;
1483     /*** IDirectInputEffect methods ***/
1484     STDMETHOD(Initialize)(THIS_ HINSTANCE, DWORD, REFGUID) PURE;
1485     STDMETHOD(GetEffectGuid)(THIS_ LPGUID) PURE;
1486     STDMETHOD(GetParameters)(THIS_ LPDIEFFECT, DWORD) PURE;
1487     STDMETHOD(SetParameters)(THIS_ LPCDIEFFECT, DWORD) PURE;
1488     STDMETHOD(Start)(THIS_ DWORD, DWORD) PURE;
1489     STDMETHOD(Stop)(THIS) PURE;
1490     STDMETHOD(GetEffectStatus)(THIS_ LPDWORD) PURE;
1491     STDMETHOD(Download)(THIS) PURE;
1492     STDMETHOD(Unload)(THIS) PURE;
1493     STDMETHOD(Escape)(THIS_ LPDIEFFESCAPE) PURE;
1494 };
1495
1496 #if !defined(__cplusplus) || defined(CINTERFACE)
1497 /*** IUnknown methods ***/
1498 #define IDirectInputEffect_QueryInterface(p,a,b) (p)->lpVtbl->QueryInterface(p,a,b)
1499 #define IDirectInputEffect_AddRef(p)             (p)->lpVtbl->AddRef(p)
1500 #define IDirectInputEffect_Release(p)            (p)->lpVtbl->Release(p)
1501 /*** IDirectInputEffect methods ***/
1502 #define IDirectInputEffect_Initialize(p,a,b,c)    (p)->lpVtbl->Initialize(p,a,b,c)
1503 #define IDirectInputEffect_GetEffectGuid(p,a)     (p)->lpVtbl->GetEffectGuid(p,a)
1504 #define IDirectInputEffect_GetParameters(p,a,b)   (p)->lpVtbl->GetParameters(p,a,b)
1505 #define IDirectInputEffect_SetParameters(p,a,b)   (p)->lpVtbl->SetParameters(p,a,b)
1506 #define IDirectInputEffect_Start(p,a,b)           (p)->lpVtbl->Start(p,a,b)
1507 #define IDirectInputEffect_Stop(p)                (p)->lpVtbl->Stop(p)
1508 #define IDirectInputEffect_GetEffectStatus(p,a)   (p)->lpVtbl->GetEffectStatus(p,a)
1509 #define IDirectInputEffect_Download(p)            (p)->lpVtbl->Download(p)
1510 #define IDirectInputEffect_Unload(p)              (p)->lpVtbl->Unload(p)
1511 #define IDirectInputEffect_Escape(p,a)            (p)->lpVtbl->Escape(p,a)
1512 #else
1513 /*** IUnknown methods ***/
1514 #define IDirectInputEffect_QueryInterface(p,a,b) (p)->QueryInterface(a,b)
1515 #define IDirectInputEffect_AddRef(p)             (p)->AddRef()
1516 #define IDirectInputEffect_Release(p)            (p)->Release()
1517 /*** IDirectInputEffect methods ***/
1518 #define IDirectInputEffect_Initialize(p,a,b,c)    (p)->Initialize(a,b,c)
1519 #define IDirectInputEffect_GetEffectGuid(p,a)     (p)->GetEffectGuid(a)
1520 #define IDirectInputEffect_GetParameters(p,a,b)   (p)->GetParameters(a,b)
1521 #define IDirectInputEffect_SetParameters(p,a,b)   (p)->SetParameters(a,b)
1522 #define IDirectInputEffect_Start(p,a,b)           (p)->Start(a,b)
1523 #define IDirectInputEffect_Stop(p)                (p)->Stop()
1524 #define IDirectInputEffect_GetEffectStatus(p,a)   (p)->GetEffectStatus(a)
1525 #define IDirectInputEffect_Download(p)            (p)->Download()
1526 #define IDirectInputEffect_Unload(p)              (p)->Unload()
1527 #define IDirectInputEffect_Escape(p,a)            (p)->Escape(a)
1528 #endif
1529
1530 #endif /* DI5 */
1531
1532
1533 /*****************************************************************************
1534  * IDirectInputDeviceA interface
1535  */
1536 #undef INTERFACE
1537 #define INTERFACE IDirectInputDeviceA
1538 DECLARE_INTERFACE_(IDirectInputDeviceA,IUnknown)
1539 {
1540     /*** IUnknown methods ***/
1541     STDMETHOD_(HRESULT,QueryInterface)(THIS_ REFIID riid, void** ppvObject) PURE;
1542     STDMETHOD_(ULONG,AddRef)(THIS) PURE;
1543     STDMETHOD_(ULONG,Release)(THIS) PURE;
1544     /*** IDirectInputDeviceA methods ***/
1545     STDMETHOD(GetCapabilities)(THIS_ LPDIDEVCAPS lpDIDevCaps) PURE;
1546     STDMETHOD(EnumObjects)(THIS_ LPDIENUMDEVICEOBJECTSCALLBACKA lpCallback, LPVOID pvRef, DWORD
      dwFlags) PURE;
1547     STDMETHOD(GetProperty)(THIS_ REFGUID rguidProp, LPDIPROPHEADER pdiph) PURE;
1548     STDMETHOD(SetProperty)(THIS_ REFGUID rguidProp, LPCDIPROPHEADER pdiph) PURE;
1549     STDMETHOD(Acquire)(THIS) PURE;
1550     STDMETHOD(Unacquire)(THIS) PURE;
1551     STDMETHOD(GetDeviceState)(THIS_ DWORD cbData, LPVOID lpvData) PURE;
```

```
1552     STDMETHOD(GetDeviceData)(THIS_ DWORD cbObjectData, LPDIDEVICEOBJECTDATA rgdod, LPDWORD pdwInOut,
       DWORD dwFlags) PURE;
1553     STDMETHOD(SetDataFormat)(THIS_ LPCDIDATAFORMAT lpdf) PURE;
1554     STDMETHOD(SetEventNotification)(THIS_ HANDLE hEvent) PURE;
1555     STDMETHOD(SetCooperativeLevel)(THIS_ HWND hwnd, DWORD dwFlags) PURE;
1556     STDMETHOD(GetObjectInfo)(THIS_ LPDIDEVICEOBJECTINSTANCEA pdidoi, DWORD dwObj, DWORD dwHow) PURE;
1557     STDMETHOD(GetDeviceInfo)(THIS_ LPDIDEVICEINSTANCEA pdidi) PURE;
1558     STDMETHOD(RunControlPanel)(THIS_ HWND hwndOwner, DWORD dwFlags) PURE;
1559     STDMETHOD(Initialize)(THIS_ HINSTANCE hinst, DWORD dwVersion, REFGUID rguid) PURE;
1560 };
1561
1562 /*****************************************************************************
1563  * IDirectInputDeviceW interface
1564  */
1565 #undef INTERFACE
1566 #define INTERFACE IDirectInputDeviceW
1567 DECLARE_INTERFACE_(IDirectInputDeviceW,IUnknown)
1568 {
1569     /*** IUnknown methods ***/
1570     STDMETHOD_(HRESULT,QueryInterface)(THIS_ REFIID riid, void** ppvObject) PURE;
1571     STDMETHOD_(ULONG,AddRef)(THIS) PURE;
1572     STDMETHOD_(ULONG,Release)(THIS) PURE;
1573     /*** IDirectInputDeviceW methods ***/
1574     STDMETHOD(GetCapabilities)(THIS_ LPDIDEVCAPS lpDIDevCaps) PURE;
1575     STDMETHOD(EnumObjects)(THIS_ LPDIENUMDEVICEOBJECTSCALLBACKW lpCallback, LPVOID pvRef, DWORD
       dwFlags) PURE;
1576     STDMETHOD(GetProperty)(THIS_ REFGUID rguidProp, LPDIPROPHEADER pdiph) PURE;
1577     STDMETHOD(SetProperty)(THIS_ REFGUID rguidProp, LPCDIPROPHEADER pdiph) PURE;
1578     STDMETHOD(Acquire)(THIS) PURE;
1579     STDMETHOD(Unacquire)(THIS) PURE;
1580     STDMETHOD(GetDeviceState)(THIS_ DWORD cbData, LPVOID lpvData) PURE;
1581     STDMETHOD(GetDeviceData)(THIS_ DWORD cbObjectData, LPDIDEVICEOBJECTDATA rgdod, LPDWORD pdwInOut,
       DWORD dwFlags) PURE;
1582     STDMETHOD(SetDataFormat)(THIS_ LPCDIDATAFORMAT lpdf) PURE;
1583     STDMETHOD(SetEventNotification)(THIS_ HANDLE hEvent) PURE;
1584     STDMETHOD(SetCooperativeLevel)(THIS_ HWND hwnd, DWORD dwFlags) PURE;
1585     STDMETHOD(GetObjectInfo)(THIS_ LPDIDEVICEOBJECTINSTANCEW pdidoi, DWORD dwObj, DWORD dwHow) PURE;
1586     STDMETHOD(GetDeviceInfo)(THIS_ LPDIDEVICEINSTANCEW pdidi) PURE;
1587     STDMETHOD(RunControlPanel)(THIS_ HWND hwndOwner, DWORD dwFlags) PURE;
1588     STDMETHOD(Initialize)(THIS_ HINSTANCE hinst, DWORD dwVersion, REFGUID rguid) PURE;
1589 };
1590
1591 #if !defined(__cplusplus) || defined(CINTERFACE)
1592 /*** IUnknown methods ***/
1593 #define IDirectInputDevice_QueryInterface(p,a,b) (p)->lpVtbl->QueryInterface(p,a,b)
1594 #define IDirectInputDevice_AddRef(p)             (p)->lpVtbl->AddRef(p)
1595 #define IDirectInputDevice_Release(p)            (p)->lpVtbl->Release(p)
1596 /*** IDirectInputDevice methods ***/
1597 #define IDirectInputDevice_GetCapabilities(p,a)        (p)->lpVtbl->GetCapabilities(p,a)
1598 #define IDirectInputDevice_EnumObjects(p,a,b,c)        (p)->lpVtbl->EnumObjects(p,a,b,c)
1599 #define IDirectInputDevice_GetProperty(p,a,b)          (p)->lpVtbl->GetProperty(p,a,b)
1600 #define IDirectInputDevice_SetProperty(p,a,b)          (p)->lpVtbl->SetProperty(p,a,b)
1601 #define IDirectInputDevice_Acquire(p)                  (p)->lpVtbl->Acquire(p)
1602 #define IDirectInputDevice_Unacquire(p)                (p)->lpVtbl->Unacquire(p)
1603 #define IDirectInputDevice_GetDeviceState(p,a,b)       (p)->lpVtbl->GetDeviceState(p,a,b)
1604 #define IDirectInputDevice_GetDeviceData(p,a,b,c,d)    (p)->lpVtbl->GetDeviceData(p,a,b,c,d)
1605 #define IDirectInputDevice_SetDataFormat(p,a)          (p)->lpVtbl->SetDataFormat(p,a)
1606 #define IDirectInputDevice_SetEventNotification(p,a)   (p)->lpVtbl->SetEventNotification(p,a)
1607 #define IDirectInputDevice_SetCooperativeLevel(p,a,b) (p)->lpVtbl->SetCooperativeLevel(p,a,b)
1608 #define IDirectInputDevice_GetObjectInfo(p,a,b,c)      (p)->lpVtbl->GetObjectInfo(p,a,b,c)
1609 #define IDirectInputDevice_GetDeviceInfo(p,a)          (p)->lpVtbl->GetDeviceInfo(p,a)
1610 #define IDirectInputDevice_RunControlPanel(p,a,b)      (p)->lpVtbl->RunControlPanel(p,a,b)
1611 #define IDirectInputDevice_Initialize(p,a,b,c)         (p)->lpVtbl->Initialize(p,a,b,c)
1612 #else
1613 /*** IUnknown methods ***/
1614 #define IDirectInputDevice_QueryInterface(p,a,b) (p)->QueryInterface(a,b)
1615 #define IDirectInputDevice_AddRef(p)             (p)->AddRef()
1616 #define IDirectInputDevice_Release(p)            (p)->Release()
1617 /*** IDirectInputDevice methods ***/
1618 #define IDirectInputDevice_GetCapabilities(p,a)        (p)->GetCapabilities(a)
1619 #define IDirectInputDevice_EnumObjects(p,a,b,c)        (p)->EnumObjects(a,b,c)
1620 #define IDirectInputDevice_GetProperty(p,a,b)          (p)->GetProperty(a,b)
1621 #define IDirectInputDevice_SetProperty(p,a,b)          (p)->SetProperty(a,b)
1622 #define IDirectInputDevice_Acquire(p)                  (p)->Acquire()
1623 #define IDirectInputDevice_Unacquire(p)                (p)->Unacquire()
1624 #define IDirectInputDevice_GetDeviceState(p,a,b)       (p)->GetDeviceState(a,b)
1625 #define IDirectInputDevice_GetDeviceData(p,a,b,c,d)    (p)->GetDeviceData(a,b,c,d)
1626 #define IDirectInputDevice_SetDataFormat(p,a)          (p)->SetDataFormat(a)
1627 #define IDirectInputDevice_SetEventNotification(p,a)   (p)->SetEventNotification(a)
1628 #define IDirectInputDevice_SetCooperativeLevel(p,a,b) (p)->SetCooperativeLevel(a,b)
1629 #define IDirectInputDevice_GetObjectInfo(p,a,b,c)      (p)->GetObjectInfo(a,b,c)
1630 #define IDirectInputDevice_GetDeviceInfo(p,a)          (p)->GetDeviceInfo(a)
1631 #define IDirectInputDevice_RunControlPanel(p,a,b)      (p)->RunControlPanel(a,b)
1632 #define IDirectInputDevice_Initialize(p,a,b,c)         (p)->Initialize(a,b,c)
1633 #endif
1634
1635
```

```
1636 #if (DIRECTINPUT_VERSION >= 0x0500)
1637 /****************************************************************************
1638  * IDirectInputDevice2A interface
1639  */
1640 #undef INTERFACE
1641 #define INTERFACE IDirectInputDevice2A
1642 DECLARE_INTERFACE_(IDirectInputDevice2A,IDirectInputDeviceA)
1643 {
1644     /*** IUnknown methods ***/
1645     STDMETHOD_(HRESULT,QueryInterface)(THIS_ REFIID riid, void** ppvObject) PURE;
1646     STDMETHOD_(ULONG,AddRef)(THIS) PURE;
1647     STDMETHOD_(ULONG,Release)(THIS) PURE;
1648     /*** IDirectInputDeviceA methods ***/
1649     STDMETHOD(GetCapabilities)(THIS_ LPDIDEVCAPS lpDIDevCaps) PURE;
1650     STDMETHOD(EnumObjects)(THIS_ LPDIENUMDEVICEOBJECTSCALLBACKA lpCallback, LPVOID pvRef, DWORD
      dwFlags) PURE;
1651     STDMETHOD(GetProperty)(THIS_ REFGUID rguidProp, LPDIPROPHEADER pdiph) PURE;
1652     STDMETHOD(SetProperty)(THIS_ REFGUID rguidProp, LPCDIPROPHEADER pdiph) PURE;
1653     STDMETHOD(Acquire)(THIS) PURE;
1654     STDMETHOD(Unacquire)(THIS) PURE;
1655     STDMETHOD(GetDeviceState)(THIS_ DWORD cbData, LPVOID lpvData) PURE;
1656     STDMETHOD(GetDeviceData)(THIS_ DWORD cbObjectData, LPDIDEVICEOBJECTDATA rgdod, LPDWORD pdwInOut,
      DWORD dwFlags) PURE;
1657     STDMETHOD(SetDataFormat)(THIS_ LPCDIDATAFORMAT lpdf) PURE;
1658     STDMETHOD(SetEventNotification)(THIS_ HANDLE hEvent) PURE;
1659     STDMETHOD(SetCooperativeLevel)(THIS_ HWND hwnd, DWORD dwFlags) PURE;
1660     STDMETHOD(GetObjectInfo)(THIS_ LPDIDEVICEOBJECTINSTANCEA pdidoi, DWORD dwObj, DWORD dwHow) PURE;
1661     STDMETHOD(GetDeviceInfo)(THIS_ LPDIDEVICEINSTANCEA pdidi) PURE;
1662     STDMETHOD(RunControlPanel)(THIS_ HWND hwndOwner, DWORD dwFlags) PURE;
1663     STDMETHOD(Initialize)(THIS_ HINSTANCE hinst, DWORD dwVersion, REFGUID rguid) PURE;
1664     /*** IDirectInputDevice2A methods ***/
1665     STDMETHOD(CreateEffect)(THIS_ REFGUID rguid, LPCDIEFFECT lpeff, LPDIRECTINPUTEFFECT *ppdeff,
      LPUNKNOWN punkOuter) PURE;
1666     STDMETHOD(EnumEffects)(THIS_ LPDIENUMEFFECTSCALLBACKA lpCallback, LPVOID pvRef, DWORD dwEffType)
      PURE;
1667     STDMETHOD(GetEffectInfo)(THIS_ LPDIEFFECTINFOA pdei, REFGUID rguid) PURE;
1668     STDMETHOD(GetForceFeedbackState)(THIS_ LPDWORD pdwOut) PURE;
1669     STDMETHOD(SendForceFeedbackCommand)(THIS_ DWORD dwFlags) PURE;
1670     STDMETHOD(EnumCreatedEffectObjects)(THIS_ LPDIENUMCREATEDEFFECTOBJECTSCALLBACK lpCallback, LPVOID
      pvRef, DWORD fl) PURE;
1671     STDMETHOD(Escape)(THIS_ LPDIEFFESCAPE pesc) PURE;
1672     STDMETHOD(Poll)(THIS) PURE;
1673     STDMETHOD(SendDeviceData)(THIS_ DWORD cbObjectData, LPCDIDEVICEOBJECTDATA rgdod, LPDWORD pdwInOut,
      DWORD fl) PURE;
1674 };
1675
1676 /****************************************************************************
1677  * IDirectInputDevice2W interface
1678  */
1679 #undef INTERFACE
1680 #define INTERFACE IDirectInputDevice2W
1681 DECLARE_INTERFACE_(IDirectInputDevice2W,IDirectInputDeviceW)
1682 {
1683     /*** IUnknown methods ***/
1684     STDMETHOD_(HRESULT,QueryInterface)(THIS_ REFIID riid, void** ppvObject) PURE;
1685     STDMETHOD_(ULONG,AddRef)(THIS) PURE;
1686     STDMETHOD_(ULONG,Release)(THIS) PURE;
1687     /*** IDirectInputDeviceW methods ***/
1688     STDMETHOD(GetCapabilities)(THIS_ LPDIDEVCAPS lpDIDevCaps) PURE;
1689     STDMETHOD(EnumObjects)(THIS_ LPDIENUMDEVICEOBJECTSCALLBACKW lpCallback, LPVOID pvRef, DWORD
      dwFlags) PURE;
1690     STDMETHOD(GetProperty)(THIS_ REFGUID rguidProp, LPDIPROPHEADER pdiph) PURE;
1691     STDMETHOD(SetProperty)(THIS_ REFGUID rguidProp, LPCDIPROPHEADER pdiph) PURE;
1692     STDMETHOD(Acquire)(THIS) PURE;
1693     STDMETHOD(Unacquire)(THIS) PURE;
1694     STDMETHOD(GetDeviceState)(THIS_ DWORD cbData, LPVOID lpvData) PURE;
1695     STDMETHOD(GetDeviceData)(THIS_ DWORD cbObjectData, LPDIDEVICEOBJECTDATA rgdod, LPDWORD pdwInOut,
      DWORD dwFlags) PURE;
1696     STDMETHOD(SetDataFormat)(THIS_ LPCDIDATAFORMAT lpdf) PURE;
1697     STDMETHOD(SetEventNotification)(THIS_ HANDLE hEvent) PURE;
1698     STDMETHOD(SetCooperativeLevel)(THIS_ HWND hwnd, DWORD dwFlags) PURE;
1699     STDMETHOD(GetObjectInfo)(THIS_ LPDIDEVICEOBJECTINSTANCEW pdidoi, DWORD dwObj, DWORD dwHow) PURE;
1700     STDMETHOD(GetDeviceInfo)(THIS_ LPDIDEVICEINSTANCEW pdidi) PURE;
1701     STDMETHOD(RunControlPanel)(THIS_ HWND hwndOwner, DWORD dwFlags) PURE;
1702     STDMETHOD(Initialize)(THIS_ HINSTANCE hinst, DWORD dwVersion, REFGUID rguid) PURE;
1703     /*** IDirectInputDevice2W methods ***/
1704     STDMETHOD(CreateEffect)(THIS_ REFGUID rguid, LPCDIEFFECT lpeff, LPDIRECTINPUTEFFECT *ppdeff,
      LPUNKNOWN punkOuter) PURE;
1705     STDMETHOD(EnumEffects)(THIS_ LPDIENUMEFFECTSCALLBACKW lpCallback, LPVOID pvRef, DWORD dwEffType)
      PURE;
1706     STDMETHOD(GetEffectInfo)(THIS_ LPDIEFFECTINFOW pdei, REFGUID rguid) PURE;
1707     STDMETHOD(GetForceFeedbackState)(THIS_ LPDWORD pdwOut) PURE;
1708     STDMETHOD(SendForceFeedbackCommand)(THIS_ DWORD dwFlags) PURE;
1709     STDMETHOD(EnumCreatedEffectObjects)(THIS_ LPDIENUMCREATEDEFFECTOBJECTSCALLBACK lpCallback, LPVOID
      pvRef, DWORD fl) PURE;
1710     STDMETHOD(Escape)(THIS_ LPDIEFFESCAPE pesc) PURE;
1711     STDMETHOD(Poll)(THIS) PURE;
```

```
1712     STDMETHOD(SendDeviceData)(THIS_ DWORD cbObjectData, LPCDIDEVICEOBJECTDATA rgdod, LPDWORD pdwInOut,
       DWORD fl) PURE;
1713 };
1714
1715 #if !defined(__cplusplus) || defined(CINTERFACE)
1716 /*** IUnknown methods ***/
1717 #define IDirectInputDevice2_QueryInterface(p,a,b) (p)->lpVtbl->QueryInterface(p,a,b)
1718 #define IDirectInputDevice2_AddRef(p)             (p)->lpVtbl->AddRef(p)
1719 #define IDirectInputDevice2_Release(p)            (p)->lpVtbl->Release(p)
1720 /*** IDirectInputDevice methods ***/
1721 #define IDirectInputDevice2_GetCapabilities(p,a)        (p)->lpVtbl->GetCapabilities(p,a)
1722 #define IDirectInputDevice2_EnumObjects(p,a,b,c)        (p)->lpVtbl->EnumObjects(p,a,b,c)
1723 #define IDirectInputDevice2_GetProperty(p,a,b)          (p)->lpVtbl->GetProperty(p,a,b)
1724 #define IDirectInputDevice2_SetProperty(p,a,b)          (p)->lpVtbl->SetProperty(p,a,b)
1725 #define IDirectInputDevice2_Acquire(p)                  (p)->lpVtbl->Acquire(p)
1726 #define IDirectInputDevice2_Unacquire(p)                (p)->lpVtbl->Unacquire(p)
1727 #define IDirectInputDevice2_GetDeviceState(p,a,b)       (p)->lpVtbl->GetDeviceState(p,a,b)
1728 #define IDirectInputDevice2_GetDeviceData(p,a,b,c,d)    (p)->lpVtbl->GetDeviceData(p,a,b,c,d)
1729 #define IDirectInputDevice2_SetDataFormat(p,a)          (p)->lpVtbl->SetDataFormat(p,a)
1730 #define IDirectInputDevice2_SetEventNotification(p,a)   (p)->lpVtbl->SetEventNotification(p,a)
1731 #define IDirectInputDevice2_SetCooperativeLevel(p,a,b)  (p)->lpVtbl->SetCooperativeLevel(p,a,b)
1732 #define IDirectInputDevice2_GetObjectInfo(p,a,b,c)      (p)->lpVtbl->GetObjectInfo(p,a,b,c)
1733 #define IDirectInputDevice2_GetDeviceInfo(p,a)          (p)->lpVtbl->GetDeviceInfo(p,a)
1734 #define IDirectInputDevice2_RunControlPanel(p,a,b)      (p)->lpVtbl->RunControlPanel(p,a,b)
1735 #define IDirectInputDevice2_Initialize(p,a,b,c)         (p)->lpVtbl->Initialize(p,a,b,c)
1736 /*** IDirectInputDevice2 methods ***/
1737 #define IDirectInputDevice2_CreateEffect(p,a,b,c,d)           (p)->lpVtbl->CreateEffect(p,a,b,c,d)
1738 #define IDirectInputDevice2_EnumEffects(p,a,b,c)              (p)->lpVtbl->EnumEffects(p,a,b,c)
1739 #define IDirectInputDevice2_GetEffectInfo(p,a,b)              (p)->lpVtbl->GetEffectInfo(p,a,b)
1740 #define IDirectInputDevice2_GetForceFeedbackState(p,a)        (p)->lpVtbl->GetForceFeedbackState(p,a)
1741 #define IDirectInputDevice2_SendForceFeedbackCommand(p,a)
       (p)->lpVtbl->SendForceFeedbackCommand(p,a)
1742 #define IDirectInputDevice2_EnumCreatedEffectObjects(p,a,b,c)
       (p)->lpVtbl->EnumCreatedEffectObjects(p,a,b,c)
1743 #define IDirectInputDevice2_Escape(p,a)                       (p)->lpVtbl->Escape(p,a)
1744 #define IDirectInputDevice2_Poll(p)                           (p)->lpVtbl->Poll(p)
1745 #define IDirectInputDevice2_SendDeviceData(p,a,b,c,d)         (p)->lpVtbl->SendDeviceData(p,a,b,c,d)
1746 #else
1747 /*** IUnknown methods ***/
1748 #define IDirectInputDevice2_QueryInterface(p,a,b) (p)->QueryInterface(a,b)
1749 #define IDirectInputDevice2_AddRef(p)             (p)->AddRef()
1750 #define IDirectInputDevice2_Release(p)            (p)->Release()
1751 /*** IDirectInputDevice methods ***/
1752 #define IDirectInputDevice2_GetCapabilities(p,a)        (p)->GetCapabilities(a)
1753 #define IDirectInputDevice2_EnumObjects(p,a,b,c)        (p)->EnumObjects(a,b,c)
1754 #define IDirectInputDevice2_GetProperty(p,a,b)          (p)->GetProperty(a,b)
1755 #define IDirectInputDevice2_SetProperty(p,a,b)          (p)->SetProperty(a,b)
1756 #define IDirectInputDevice2_Acquire(p)                  (p)->Acquire()
1757 #define IDirectInputDevice2_Unacquire(p)                (p)->Unacquire()
1758 #define IDirectInputDevice2_GetDeviceState(p,a,b)       (p)->GetDeviceState(a,b)
1759 #define IDirectInputDevice2_GetDeviceData(p,a,b,c,d)    (p)->GetDeviceData(a,b,c,d)
1760 #define IDirectInputDevice2_SetDataFormat(p,a)          (p)->SetDataFormat(a)
1761 #define IDirectInputDevice2_SetEventNotification(p,a)   (p)->SetEventNotification(a)
1762 #define IDirectInputDevice2_SetCooperativeLevel(p,a,b)  (p)->SetCooperativeLevel(a,b)
1763 #define IDirectInputDevice2_GetObjectInfo(p,a,b,c)      (p)->GetObjectInfo(a,b,c)
1764 #define IDirectInputDevice2_GetDeviceInfo(p,a)          (p)->GetDeviceInfo(a)
1765 #define IDirectInputDevice2_RunControlPanel(p,a,b)      (p)->RunControlPanel(a,b)
1766 #define IDirectInputDevice2_Initialize(p,a,b,c)         (p)->Initialize(a,b,c)
1767 /*** IDirectInputDevice2 methods ***/
1768 #define IDirectInputDevice2_CreateEffect(p,a,b,c,d)           (p)->CreateEffect(a,b,c,d)
1769 #define IDirectInputDevice2_EnumEffects(p,a,b,c)              (p)->EnumEffects(a,b,c)
1770 #define IDirectInputDevice2_GetEffectInfo(p,a,b)              (p)->GetEffectInfo(a,b)
1771 #define IDirectInputDevice2_GetForceFeedbackState(p,a)        (p)->GetForceFeedbackState(a)
1772 #define IDirectInputDevice2_SendForceFeedbackCommand(p,a)     (p)->SendForceFeedbackCommand(a)
1773 #define IDirectInputDevice2_EnumCreatedEffectObjects(p,a,b,c) (p)->EnumCreatedEffectObjects(a,b,c)
1774 #define IDirectInputDevice2_Escape(p,a)                       (p)->Escape(a)
1775 #define IDirectInputDevice2_Poll(p)                           (p)->Poll()
1776 #define IDirectInputDevice2_SendDeviceData(p,a,b,c,d)         (p)->SendDeviceData(a,b,c,d)
1777 #endif
1778 #endif /* DI5 */
1779
1780 #if DIRECTINPUT_VERSION >= 0x0700
1781 /*****************************************************************************
1782  * IDirectInputDevice7A interface
1783  */
1784 #undef INTERFACE
1785 #define INTERFACE IDirectInputDevice7A
1786 DECLARE_INTERFACE_(IDirectInputDevice7A,IDirectInputDevice2A)
1787 {
1788     /*** IUnknown methods ***/
1789     STDMETHOD_(HRESULT,QueryInterface)(THIS_ REFIID riid, void** ppvObject) PURE;
1790     STDMETHOD_(ULONG,AddRef)(THIS) PURE;
1791     STDMETHOD_(ULONG,Release)(THIS) PURE;
1792     /*** IDirectInputDeviceA methods ***/
1793     STDMETHOD(GetCapabilities)(THIS_ LPDIDEVCAPS lpDIDevCaps) PURE;
1794     STDMETHOD(EnumObjects)(THIS_ LPDIENUMDEVICEOBJECTSCALLBACKA lpCallback, LPVOID pvRef, DWORD
       dwFlags) PURE;
```

```
1795     STDMETHOD(GetProperty)(THIS_ REFGUID rguidProp, LPDIPROPHEADER pdiph) PURE;
1796     STDMETHOD(SetProperty)(THIS_ REFGUID rguidProp, LPCDIPROPHEADER pdiph) PURE;
1797     STDMETHOD(Acquire)(THIS) PURE;
1798     STDMETHOD(Unacquire)(THIS) PURE;
1799     STDMETHOD(GetDeviceState)(THIS_ DWORD cbData, LPVOID lpvData) PURE;
1800     STDMETHOD(GetDeviceData)(THIS_ DWORD cbObjectData, LPDIDEVICEOBJECTDATA rgdod, LPDWORD pdwInOut,
       DWORD dwFlags) PURE;
1801     STDMETHOD(SetDataFormat)(THIS_ LPCDIDATAFORMAT lpdf) PURE;
1802     STDMETHOD(SetEventNotification)(THIS_ HANDLE hEvent) PURE;
1803     STDMETHOD(SetCooperativeLevel)(THIS_ HWND hwnd, DWORD dwFlags) PURE;
1804     STDMETHOD(GetObjectInfo)(THIS_ LPDIDEVICEOBJECTINSTANCEA pdidoi, DWORD dwObj, DWORD dwHow) PURE;
1805     STDMETHOD(GetDeviceInfo)(THIS_ LPDIDEVICEINSTANCEA pdidi) PURE;
1806     STDMETHOD(RunControlPanel)(THIS_ HWND hwndOwner, DWORD dwFlags) PURE;
1807     STDMETHOD(Initialize)(THIS_ HINSTANCE hinst, DWORD dwVersion, REFGUID rguid) PURE;
1808     /*** IDirectInputDevice2A methods ***/
1809     STDMETHOD(CreateEffect)(THIS_ REFGUID rguid, LPCDIEFFECT lpeff, LPDIRECTINPUTEFFECT *ppdeff,
       LPUNKNOWN punkOuter) PURE;
1810     STDMETHOD(EnumEffects)(THIS_ LPDIENUMEFFECTSCALLBACKA lpCallback, LPVOID pvRef, DWORD dwEffType)
       PURE;
1811     STDMETHOD(GetEffectInfo)(THIS_ LPDIEFFECTINFOA pdei, REFGUID rguid) PURE;
1812     STDMETHOD(GetForceFeedbackState)(THIS_ LPDWORD pdwOut) PURE;
1813     STDMETHOD(SendForceFeedbackCommand)(THIS_ DWORD dwFlags) PURE;
1814     STDMETHOD(EnumCreatedEffectObjects)(THIS_ LPDIENUMCREATEDEFFECTOBJECTSCALLBACK lpCallback, LPVOID
       pvRef, DWORD fl) PURE;
1815     STDMETHOD(Escape)(THIS_ LPDIEFFESCAPE pesc) PURE;
1816     STDMETHOD(Poll)(THIS) PURE;
1817     STDMETHOD(SendDeviceData)(THIS_ DWORD cbObjectData, LPCDIDEVICEOBJECTDATA rgdod, LPDWORD pdwInOut,
       DWORD fl) PURE;
1818     /*** IDirectInputDevice7A methods ***/
1819     STDMETHOD(EnumEffectsInFile)(THIS_ LPCSTR lpszFileName,LPDIENUMEFFECTSINFILECALLBACK pec,LPVOID
       pvRef,DWORD dwFlags) PURE;
1820     STDMETHOD(WriteEffectToFile)(THIS_ LPCSTR lpszFileName,DWORD dwEntries,LPDIFILEEFFECT
       rgDiFileEft,DWORD dwFlags) PURE;
1821 };
1822
1823 /*****************************************************************************
1824  * IDirectInputDevice7W interface
1825  */
1826 #undef INTERFACE
1827 #define INTERFACE IDirectInputDevice7W
1828 DECLARE_INTERFACE_(IDirectInputDevice7W,IDirectInputDevice2W)
1829 {
1830     /*** IUnknown methods ***/
1831     STDMETHOD_(HRESULT,QueryInterface)(THIS_ REFIID riid, void** ppvObject) PURE;
1832     STDMETHOD_(ULONG,AddRef)(THIS) PURE;
1833     STDMETHOD_(ULONG,Release)(THIS) PURE;
1834     /*** IDirectInputDeviceW methods ***/
1835     STDMETHOD(GetCapabilities)(THIS_ LPDIDEVCAPS lpDIDevCaps) PURE;
1836     STDMETHOD(EnumObjects)(THIS_ LPDIENUMDEVICEOBJECTSCALLBACKW lpCallback, LPVOID pvRef, DWORD
       dwFlags) PURE;
1837     STDMETHOD(GetProperty)(THIS_ REFGUID rguidProp, LPDIPROPHEADER pdiph) PURE;
1838     STDMETHOD(SetProperty)(THIS_ REFGUID rguidProp, LPCDIPROPHEADER pdiph) PURE;
1839     STDMETHOD(Acquire)(THIS) PURE;
1840     STDMETHOD(Unacquire)(THIS) PURE;
1841     STDMETHOD(GetDeviceState)(THIS_ DWORD cbData, LPVOID lpvData) PURE;
1842     STDMETHOD(GetDeviceData)(THIS_ DWORD cbObjectData, LPDIDEVICEOBJECTDATA rgdod, LPDWORD pdwInOut,
       DWORD dwFlags) PURE;
1843     STDMETHOD(SetDataFormat)(THIS_ LPCDIDATAFORMAT lpdf) PURE;
1844     STDMETHOD(SetEventNotification)(THIS_ HANDLE hEvent) PURE;
1845     STDMETHOD(SetCooperativeLevel)(THIS_ HWND hwnd, DWORD dwFlags) PURE;
1846     STDMETHOD(GetObjectInfo)(THIS_ LPDIDEVICEOBJECTINSTANCEW pdidoi, DWORD dwObj, DWORD dwHow) PURE;
1847     STDMETHOD(GetDeviceInfo)(THIS_ LPDIDEVICEINSTANCEW pdidi) PURE;
1848     STDMETHOD(RunControlPanel)(THIS_ HWND hwndOwner, DWORD dwFlags) PURE;
1849     STDMETHOD(Initialize)(THIS_ HINSTANCE hinst, DWORD dwVersion, REFGUID rguid) PURE;
1850     /*** IDirectInputDevice2W methods ***/
1851     STDMETHOD(CreateEffect)(THIS_ REFGUID rguid, LPCDIEFFECT lpeff, LPDIRECTINPUTEFFECT *ppdeff,
       LPUNKNOWN punkOuter) PURE;
1852     STDMETHOD(EnumEffects)(THIS_ LPDIENUMEFFECTSCALLBACKW lpCallback, LPVOID pvRef, DWORD dwEffType)
       PURE;
1853     STDMETHOD(GetEffectInfo)(THIS_ LPDIEFFECTINFOW pdei, REFGUID rguid) PURE;
1854     STDMETHOD(GetForceFeedbackState)(THIS_ LPDWORD pdwOut) PURE;
1855     STDMETHOD(SendForceFeedbackCommand)(THIS_ DWORD dwFlags) PURE;
1856     STDMETHOD(EnumCreatedEffectObjects)(THIS_ LPDIENUMCREATEDEFFECTOBJECTSCALLBACK lpCallback, LPVOID
       pvRef, DWORD fl) PURE;
1857     STDMETHOD(Escape)(THIS_ LPDIEFFESCAPE pesc) PURE;
1858     STDMETHOD(Poll)(THIS) PURE;
1859     STDMETHOD(SendDeviceData)(THIS_ DWORD cbObjectData, LPCDIDEVICEOBJECTDATA rgdod, LPDWORD pdwInOut,
       DWORD fl) PURE;
1860     /*** IDirectInputDevice7W methods ***/
1861     STDMETHOD(EnumEffectsInFile)(THIS_ LPCWSTR lpszFileName,LPDIENUMEFFECTSINFILECALLBACK pec,LPVOID
       pvRef,DWORD dwFlags) PURE;
1862     STDMETHOD(WriteEffectToFile)(THIS_ LPCWSTR lpszFileName,DWORD dwEntries,LPDIFILEEFFECT
       rgDiFileEft,DWORD dwFlags) PURE;
1863 };
1864
1865 #if !defined(__cplusplus) || defined(CINTERFACE)
1866 /*** IUnknown methods ***/
```

```
1867 #define IDirectInputDevice7_QueryInterface(p,a,b) (p)->lpVtbl->QueryInterface(p,a,b)
1868 #define IDirectInputDevice7_AddRef(p)             (p)->lpVtbl->AddRef(p)
1869 #define IDirectInputDevice7_Release(p)            (p)->lpVtbl->Release(p)
1870 /*** IDirectInputDevice methods ***/
1871 #define IDirectInputDevice7_GetCapabilities(p,a)        (p)->lpVtbl->GetCapabilities(p,a)
1872 #define IDirectInputDevice7_EnumObjects(p,a,b,c)        (p)->lpVtbl->EnumObjects(p,a,b,c)
1873 #define IDirectInputDevice7_GetProperty(p,a,b)          (p)->lpVtbl->GetProperty(p,a,b)
1874 #define IDirectInputDevice7_SetProperty(p,a,b)          (p)->lpVtbl->SetProperty(p,a,b)
1875 #define IDirectInputDevice7_Acquire(p)                  (p)->lpVtbl->Acquire(p)
1876 #define IDirectInputDevice7_Unacquire(p)                (p)->lpVtbl->Unacquire(p)
1877 #define IDirectInputDevice7_GetDeviceState(p,a,b)       (p)->lpVtbl->GetDeviceState(p,a,b)
1878 #define IDirectInputDevice7_GetDeviceData(p,a,b,c,d)    (p)->lpVtbl->GetDeviceData(p,a,b,c,d)
1879 #define IDirectInputDevice7_SetDataFormat(p,a)          (p)->lpVtbl->SetDataFormat(p,a)
1880 #define IDirectInputDevice7_SetEventNotification(p,a)   (p)->lpVtbl->SetEventNotification(p,a)
1881 #define IDirectInputDevice7_SetCooperativeLevel(p,a,b)  (p)->lpVtbl->SetCooperativeLevel(p,a,b)
1882 #define IDirectInputDevice7_GetObjectInfo(p,a,b,c)      (p)->lpVtbl->GetObjectInfo(p,a,b,c)
1883 #define IDirectInputDevice7_GetDeviceInfo(p,a)          (p)->lpVtbl->GetDeviceInfo(p,a)
1884 #define IDirectInputDevice7_RunControlPanel(p,a,b)      (p)->lpVtbl->RunControlPanel(p,a,b)
1885 #define IDirectInputDevice7_Initialize(p,a,b,c)         (p)->lpVtbl->Initialize(p,a,b,c)
1886 /*** IDirectInputDevice2 methods ***/
1887 #define IDirectInputDevice7_CreateEffect(p,a,b,c,d)           (p)->lpVtbl->CreateEffect(p,a,b,c,d)
1888 #define IDirectInputDevice7_EnumEffects(p,a,b,c)              (p)->lpVtbl->EnumEffects(p,a,b,c)
1889 #define IDirectInputDevice7_GetEffectInfo(p,a,b)              (p)->lpVtbl->GetEffectInfo(p,a,b)
1890 #define IDirectInputDevice7_GetForceFeedbackState(p,a)        (p)->lpVtbl->GetForceFeedbackState(p,a)
1891 #define IDirectInputDevice7_SendForceFeedbackCommand(p,a)
       (p)->lpVtbl->SendForceFeedbackCommand(p,a)
1892 #define IDirectInputDevice7_EnumCreatedEffectObjects(p,a,b,c)
       (p)->lpVtbl->EnumCreatedEffectObjects(p,a,b,c)
1893 #define IDirectInputDevice7_Escape(p,a)                       (p)->lpVtbl->Escape(p,a)
1894 #define IDirectInputDevice7_Poll(p)                           (p)->lpVtbl->Poll(p)
1895 #define IDirectInputDevice7_SendDeviceData(p,a,b,c,d)         (p)->lpVtbl->SendDeviceData(p,a,b,c,d)
1896 /*** IDirectInputDevice7 methods ***/
1897 #define IDirectInputDevice7_EnumEffectsInFile(p,a,b,c,d)  (p)->lpVtbl->EnumEffectsInFile(p,a,b,c,d)
1898 #define IDirectInputDevice7_WriteEffectToFile(p,a,b,c,d)  (p)->lpVtbl->WriteEffectToFile(p,a,b,c,d)
1899 #else
1900 /*** IUnknown methods ***/
1901 #define IDirectInputDevice7_QueryInterface(p,a,b) (p)->QueryInterface(a,b)
1902 #define IDirectInputDevice7_AddRef(p)             (p)->AddRef()
1903 #define IDirectInputDevice7_Release(p)            (p)->Release()
1904 /*** IDirectInputDevice methods ***/
1905 #define IDirectInputDevice7_GetCapabilities(p,a)        (p)->GetCapabilities(a)
1906 #define IDirectInputDevice7_EnumObjects(p,a,b,c)        (p)->EnumObjects(a,b,c)
1907 #define IDirectInputDevice7_GetProperty(p,a,b)          (p)->GetProperty(a,b)
1908 #define IDirectInputDevice7_SetProperty(p,a,b)          (p)->SetProperty(a,b)
1909 #define IDirectInputDevice7_Acquire(p)                  (p)->Acquire()
1910 #define IDirectInputDevice7_Unacquire(p)                (p)->Unacquire()
1911 #define IDirectInputDevice7_GetDeviceState(p,a,b)       (p)->GetDeviceState(a,b)
1912 #define IDirectInputDevice7_GetDeviceData(p,a,b,c,d)    (p)->GetDeviceData(a,b,c,d)
1913 #define IDirectInputDevice7_SetDataFormat(p,a)          (p)->SetDataFormat(a)
1914 #define IDirectInputDevice7_SetEventNotification(p,a)   (p)->SetEventNotification(a)
1915 #define IDirectInputDevice7_SetCooperativeLevel(p,a,b)  (p)->SetCooperativeLevel(a,b)
1916 #define IDirectInputDevice7_GetObjectInfo(p,a,b,c)      (p)->GetObjectInfo(a,b,c)
1917 #define IDirectInputDevice7_GetDeviceInfo(p,a)          (p)->GetDeviceInfo(a)
1918 #define IDirectInputDevice7_RunControlPanel(p,a,b)      (p)->RunControlPanel(a,b)
1919 #define IDirectInputDevice7_Initialize(p,a,b,c)         (p)->Initialize(a,b,c)
1920 /*** IDirectInputDevice2 methods ***/
1921 #define IDirectInputDevice7_CreateEffect(p,a,b,c,d)           (p)->CreateEffect(a,b,c,d)
1922 #define IDirectInputDevice7_EnumEffects(p,a,b,c)              (p)->EnumEffects(a,b,c)
1923 #define IDirectInputDevice7_GetEffectInfo(p,a,b)              (p)->GetEffectInfo(a,b)
1924 #define IDirectInputDevice7_GetForceFeedbackState(p,a)        (p)->GetForceFeedbackState(a)
1925 #define IDirectInputDevice7_SendForceFeedbackCommand(p,a)     (p)->SendForceFeedbackCommand(a)
1926 #define IDirectInputDevice7_EnumCreatedEffectObjects(p,a,b,c) (p)->EnumCreatedEffectObjects(a,b,c)
1927 #define IDirectInputDevice7_Escape(p,a)                       (p)->Escape(a)
1928 #define IDirectInputDevice7_Poll(p)                           (p)->Poll()
1929 #define IDirectInputDevice7_SendDeviceData(p,a,b,c,d)         (p)->SendDeviceData(a,b,c,d)
1930 /*** IDirectInputDevice7 methods ***/
1931 #define IDirectInputDevice7_EnumEffectsInFile(p,a,b,c,d)  (p)->EnumEffectsInFile(a,b,c,d)
1932 #define IDirectInputDevice7_WriteEffectToFile(p,a,b,c,d)  (p)->WriteEffectToFile(a,b,c,d)
1933 #endif
1934
1935 #endif /* DI7 */
1936
1937 #if DIRECTINPUT_VERSION >= 0x0800
1938 /*****************************************************************************
1939  * IDirectInputDevice8A interface
1940  */
1941 #undef INTERFACE
1942 #define INTERFACE IDirectInputDevice8A
1943 DECLARE_INTERFACE_(IDirectInputDevice8A,IDirectInputDevice7A)
1944 {
1945     /*** IUnknown methods ***/
1946     STDMETHOD_(HRESULT,QueryInterface)(THIS_ REFIID riid, void** ppvObject) PURE;
1947     STDMETHOD_(ULONG,AddRef)(THIS) PURE;
1948     STDMETHOD_(ULONG,Release)(THIS) PURE;
1949     /*** IDirectInputDeviceA methods ***/
1950     STDMETHOD(GetCapabilities)(THIS_ LPDIDEVCAPS lpDIDevCaps) PURE;
1951     STDMETHOD(EnumObjects)(THIS_ LPDIENUMDEVICEOBJECTSCALLBACKA lpCallback, LPVOID pvRef, DWORD
```

```
    dwFlags) PURE;
1952    STDMETHOD(GetProperty)(THIS_ REFGUID rguidProp, LPDIPROPHEADER pdiph) PURE;
1953    STDMETHOD(SetProperty)(THIS_ REFGUID rguidProp, LPCDIPROPHEADER pdiph) PURE;
1954    STDMETHOD(Acquire)(THIS) PURE;
1955    STDMETHOD(Unacquire)(THIS) PURE;
1956    STDMETHOD(GetDeviceState)(THIS_ DWORD cbData, LPVOID lpvData) PURE;
1957    STDMETHOD(GetDeviceData)(THIS_ DWORD cbObjectData, LPDIDEVICEOBJECTDATA rgdod, LPDWORD pdwInOut,
    DWORD dwFlags) PURE;
1958    STDMETHOD(SetDataFormat)(THIS_ LPCDIDATAFORMAT lpdf) PURE;
1959    STDMETHOD(SetEventNotification)(THIS_ HANDLE hEvent) PURE;
1960    STDMETHOD(SetCooperativeLevel)(THIS_ HWND hwnd, DWORD dwFlags) PURE;
1961    STDMETHOD(GetObjectInfo)(THIS_ LPDIDEVICEOBJECTINSTANCEA pdidoi, DWORD dwObj, DWORD dwHow) PURE;
1962    STDMETHOD(GetDeviceInfo)(THIS_ LPDIDEVICEINSTANCEA pdidi) PURE;
1963    STDMETHOD(RunControlPanel)(THIS_ HWND hwndOwner, DWORD dwFlags) PURE;
1964    STDMETHOD(Initialize)(THIS_ HINSTANCE hinst, DWORD dwVersion, REFGUID rguid) PURE;
1965    /*** IDirectInputDevice2A methods ***/
1966    STDMETHOD(CreateEffect)(THIS_ REFGUID rguid, LPCDIEFFECT lpeff, LPDIRECTINPUTEFFECT *ppdeff,
    LPUNKNOWN punkOuter) PURE;
1967    STDMETHOD(EnumEffects)(THIS_ LPDIENUMEFFECTSCALLBACKA lpCallback, LPVOID pvRef, DWORD dwEffType)
    PURE;
1968    STDMETHOD(GetEffectInfo)(THIS_ LPDIEFFECTINFOA pdei, REFGUID rguid) PURE;
1969    STDMETHOD(GetForceFeedbackState)(THIS_ LPDWORD pdwOut) PURE;
1970    STDMETHOD(SendForceFeedbackCommand)(THIS_ DWORD dwFlags) PURE;
1971    STDMETHOD(EnumCreatedEffectObjects)(THIS_ LPDIENUMCREATEDEFFECTOBJECTSCALLBACK lpCallback, LPVOID
    pvRef, DWORD fl) PURE;
1972    STDMETHOD(Escape)(THIS_ LPDIEFFESCAPE pesc) PURE;
1973    STDMETHOD(Poll)(THIS) PURE;
1974    STDMETHOD(SendDeviceData)(THIS_ DWORD cbObjectData, LPCDIDEVICEOBJECTDATA rgdod, LPDWORD pdwInOut,
    DWORD fl) PURE;
1975    /*** IDirectInputDevice7A methods ***/
1976    STDMETHOD(EnumEffectsInFile)(THIS_ LPCSTR lpszFileName,LPDIENUMEFFECTSINFILECALLBACK pec,LPVOID
    pvRef,DWORD dwFlags) PURE;
1977    STDMETHOD(WriteEffectToFile)(THIS_ LPCSTR lpszFileName,DWORD dwEntries,LPDIFILEEFFECT
    rgDiFileEft,DWORD dwFlags) PURE;
1978    /*** IDirectInputDevice8A methods ***/
1979    STDMETHOD(BuildActionMap)(THIS_ LPDIACTIONFORMATA lpdiaf, LPCSTR lpszUserName, DWORD dwFlags) PURE;
1980    STDMETHOD(SetActionMap)(THIS_ LPDIACTIONFORMATA lpdiaf, LPCSTR lpszUserName, DWORD dwFlags) PURE;
1981    STDMETHOD(GetImageInfo)(THIS_ LPDIDEVICEIMAGEINFOHEADERA lpdiDevImageInfoHeader) PURE;
1982 };
1983
1984 /*****************************************************************************
1985  * IDirectInputDevice8W interface
1986  */
1987 #undef INTERFACE
1988 #define INTERFACE IDirectInputDevice8W
1989 DECLARE_INTERFACE_(IDirectInputDevice8W,IDirectInputDevice7W)
1990 {
1991    /*** IUnknown methods ***/
1992    STDMETHOD_(HRESULT,QueryInterface)(THIS_ REFIID riid, void** ppvObject) PURE;
1993    STDMETHOD_(ULONG,AddRef)(THIS) PURE;
1994    STDMETHOD_(ULONG,Release)(THIS) PURE;
1995    /*** IDirectInputDeviceW methods ***/
1996    STDMETHOD(GetCapabilities)(THIS_ LPDIDEVCAPS lpDIDevCaps) PURE;
1997    STDMETHOD(EnumObjects)(THIS_ LPDIENUMDEVICEOBJECTSCALLBACKW lpCallback, LPVOID pvRef, DWORD
    dwFlags) PURE;
1998    STDMETHOD(GetProperty)(THIS_ REFGUID rguidProp, LPDIPROPHEADER pdiph) PURE;
1999    STDMETHOD(SetProperty)(THIS_ REFGUID rguidProp, LPCDIPROPHEADER pdiph) PURE;
2000    STDMETHOD(Acquire)(THIS) PURE;
2001    STDMETHOD(Unacquire)(THIS) PURE;
2002    STDMETHOD(GetDeviceState)(THIS_ DWORD cbData, LPVOID lpvData) PURE;
2003    STDMETHOD(GetDeviceData)(THIS_ DWORD cbObjectData, LPDIDEVICEOBJECTDATA rgdod, LPDWORD pdwInOut,
    DWORD dwFlags) PURE;
2004    STDMETHOD(SetDataFormat)(THIS_ LPCDIDATAFORMAT lpdf) PURE;
2005    STDMETHOD(SetEventNotification)(THIS_ HANDLE hEvent) PURE;
2006    STDMETHOD(SetCooperativeLevel)(THIS_ HWND hwnd, DWORD dwFlags) PURE;
2007    STDMETHOD(GetObjectInfo)(THIS_ LPDIDEVICEOBJECTINSTANCEW pdidoi, DWORD dwObj, DWORD dwHow) PURE;
2008    STDMETHOD(GetDeviceInfo)(THIS_ LPDIDEVICEINSTANCEW pdidi) PURE;
2009    STDMETHOD(RunControlPanel)(THIS_ HWND hwndOwner, DWORD dwFlags) PURE;
2010    STDMETHOD(Initialize)(THIS_ HINSTANCE hinst, DWORD dwVersion, REFGUID rguid) PURE;
2011    /*** IDirectInputDevice2W methods ***/
2012    STDMETHOD(CreateEffect)(THIS_ REFGUID rguid, LPCDIEFFECT lpeff, LPDIRECTINPUTEFFECT *ppdeff,
    LPUNKNOWN punkOuter) PURE;
2013    STDMETHOD(EnumEffects)(THIS_ LPDIENUMEFFECTSCALLBACKW lpCallback, LPVOID pvRef, DWORD dwEffType)
    PURE;
2014    STDMETHOD(GetEffectInfo)(THIS_ LPDIEFFECTINFOW pdei, REFGUID rguid) PURE;
2015    STDMETHOD(GetForceFeedbackState)(THIS_ LPDWORD pdwOut) PURE;
2016    STDMETHOD(SendForceFeedbackCommand)(THIS_ DWORD dwFlags) PURE;
2017    STDMETHOD(EnumCreatedEffectObjects)(THIS_ LPDIENUMCREATEDEFFECTOBJECTSCALLBACK lpCallback, LPVOID
    pvRef, DWORD fl) PURE;
2018    STDMETHOD(Escape)(THIS_ LPDIEFFESCAPE pesc) PURE;
2019    STDMETHOD(Poll)(THIS) PURE;
2020    STDMETHOD(SendDeviceData)(THIS_ DWORD cbObjectData, LPCDIDEVICEOBJECTDATA rgdod, LPDWORD pdwInOut,
    DWORD fl) PURE;
2021    /*** IDirectInputDevice7W methods ***/
2022    STDMETHOD(EnumEffectsInFile)(THIS_ LPCWSTR lpszFileName,LPDIENUMEFFECTSINFILECALLBACK pec,LPVOID
    pvRef,DWORD dwFlags) PURE;
2023    STDMETHOD(WriteEffectToFile)(THIS_ LPCWSTR lpszFileName,DWORD dwEntries,LPDIFILEEFFECT
```

```
    rgDiFileEft,DWORD dwFlags) PURE;
2024    /*** IDirectInputDevice8W methods ***/
2025    STDMETHOD(BuildActionMap)(THIS_ LPDIACTIONFORMATW lpdiaf, LPCWSTR lpszUserName, DWORD dwFlags)
    PURE;
2026    STDMETHOD(SetActionMap)(THIS_ LPDIACTIONFORMATW lpdiaf, LPCWSTR lpszUserName, DWORD dwFlags) PURE;
2027    STDMETHOD(GetImageInfo)(THIS_ LPDIDEVICEIMAGEINFOHEADERW lpdiDevImageInfoHeader) PURE;
2028 };
2029
2030 #if !defined(__cplusplus) || defined(CINTERFACE)
2031 /*** IUnknown methods ***/
2032 #define IDirectInputDevice8_QueryInterface(p,a,b) (p)->lpVtbl->QueryInterface(p,a,b)
2033 #define IDirectInputDevice8_AddRef(p)             (p)->lpVtbl->AddRef(p)
2034 #define IDirectInputDevice8_Release(p)            (p)->lpVtbl->Release(p)
2035 /*** IDirectInputDevice methods ***/
2036 #define IDirectInputDevice8_GetCapabilities(p,a)        (p)->lpVtbl->GetCapabilities(p,a)
2037 #define IDirectInputDevice8_EnumObjects(p,a,b,c)        (p)->lpVtbl->EnumObjects(p,a,b,c)
2038 #define IDirectInputDevice8_GetProperty(p,a,b)          (p)->lpVtbl->GetProperty(p,a,b)
2039 #define IDirectInputDevice8_SetProperty(p,a,b)          (p)->lpVtbl->SetProperty(p,a,b)
2040 #define IDirectInputDevice8_Acquire(p)                  (p)->lpVtbl->Acquire(p)
2041 #define IDirectInputDevice8_Unacquire(p)                (p)->lpVtbl->Unacquire(p)
2042 #define IDirectInputDevice8_GetDeviceState(p,a,b)       (p)->lpVtbl->GetDeviceState(p,a,b)
2043 #define IDirectInputDevice8_GetDeviceData(p,a,b,c,d)    (p)->lpVtbl->GetDeviceData(p,a,b,c,d)
2044 #define IDirectInputDevice8_SetDataFormat(p,a)          (p)->lpVtbl->SetDataFormat(p,a)
2045 #define IDirectInputDevice8_SetEventNotification(p,a)   (p)->lpVtbl->SetEventNotification(p,a)
2046 #define IDirectInputDevice8_SetCooperativeLevel(p,a,b)  (p)->lpVtbl->SetCooperativeLevel(p,a,b)
2047 #define IDirectInputDevice8_GetObjectInfo(p,a,b,c)      (p)->lpVtbl->GetObjectInfo(p,a,b,c)
2048 #define IDirectInputDevice8_GetDeviceInfo(p,a)          (p)->lpVtbl->GetDeviceInfo(p,a)
2049 #define IDirectInputDevice8_RunControlPanel(p,a,b)      (p)->lpVtbl->RunControlPanel(p,a,b)
2050 #define IDirectInputDevice8_Initialize(p,a,b,c)         (p)->lpVtbl->Initialize(p,a,b,c)
2051 /*** IDirectInputDevice2 methods ***/
2052 #define IDirectInputDevice8_CreateEffect(p,a,b,c,d)           (p)->lpVtbl->CreateEffect(p,a,b,c,d)
2053 #define IDirectInputDevice8_EnumEffects(p,a,b,c)              (p)->lpVtbl->EnumEffects(p,a,b,c)
2054 #define IDirectInputDevice8_GetEffectInfo(p,a,b)              (p)->lpVtbl->GetEffectInfo(p,a,b)
2055 #define IDirectInputDevice8_GetForceFeedbackState(p,a)        (p)->lpVtbl->GetForceFeedbackState(p,a)
2056 #define IDirectInputDevice8_SendForceFeedbackCommand(p,a)
      (p)->lpVtbl->SendForceFeedbackCommand(p,a)
2057 #define IDirectInputDevice8_EnumCreatedEffectObjects(p,a,b,c)
      (p)->lpVtbl->EnumCreatedEffectObjects(p,a,b,c)
2058 #define IDirectInputDevice8_Escape(p,a)                       (p)->lpVtbl->Escape(p,a)
2059 #define IDirectInputDevice8_Poll(p)                           (p)->lpVtbl->Poll(p)
2060 #define IDirectInputDevice8_SendDeviceData(p,a,b,c,d)         (p)->lpVtbl->SendDeviceData(p,a,b,c,d)
2061 /*** IDirectInputDevice7 methods ***/
2062 #define IDirectInputDevice8_EnumEffectsInFile(p,a,b,c,d)  (p)->lpVtbl->EnumEffectsInFile(p,a,b,c,d)
2063 #define IDirectInputDevice8_WriteEffectToFile(p,a,b,c,d)  (p)->lpVtbl->WriteEffectToFile(p,a,b,c,d)
2064 /*** IDirectInputDevice8 methods ***/
2065 #define IDirectInputDevice8_BuildActionMap(p,a,b,c)  (p)->lpVtbl->BuildActionMap(p,a,b,c)
2066 #define IDirectInputDevice8_SetActionMap(p,a,b,c)    (p)->lpVtbl->SetActionMap(p,a,b,c)
2067 #define IDirectInputDevice8_GetImageInfo(p,a)        (p)->lpVtbl->GetImageInfo(p,a)
2068 #else
2069 /*** IUnknown methods ***/
2070 #define IDirectInputDevice8_QueryInterface(p,a,b) (p)->QueryInterface(a,b)
2071 #define IDirectInputDevice8_AddRef(p)             (p)->AddRef()
2072 #define IDirectInputDevice8_Release(p)            (p)->Release()
2073 /*** IDirectInputDevice methods ***/
2074 #define IDirectInputDevice8_GetCapabilities(p,a)        (p)->GetCapabilities(a)
2075 #define IDirectInputDevice8_EnumObjects(p,a,b,c)        (p)->EnumObjects(a,b,c)
2076 #define IDirectInputDevice8_GetProperty(p,a,b)          (p)->GetProperty(a,b)
2077 #define IDirectInputDevice8_SetProperty(p,a,b)          (p)->SetProperty(a,b)
2078 #define IDirectInputDevice8_Acquire(p)                  (p)->Acquire()
2079 #define IDirectInputDevice8_Unacquire(p)                (p)->Unacquire()
2080 #define IDirectInputDevice8_GetDeviceState(p,a,b)       (p)->GetDeviceState(a,b)
2081 #define IDirectInputDevice8_GetDeviceData(p,a,b,c,d)    (p)->GetDeviceData(a,b,c,d)
2082 #define IDirectInputDevice8_SetDataFormat(p,a)          (p)->SetDataFormat(a)
2083 #define IDirectInputDevice8_SetEventNotification(p,a)   (p)->SetEventNotification(a)
2084 #define IDirectInputDevice8_SetCooperativeLevel(p,a,b)  (p)->SetCooperativeLevel(a,b)
2085 #define IDirectInputDevice8_GetObjectInfo(p,a,b,c)      (p)->GetObjectInfo(a,b,c)
2086 #define IDirectInputDevice8_GetDeviceInfo(p,a)          (p)->GetDeviceInfo(a)
2087 #define IDirectInputDevice8_RunControlPanel(p,a,b)      (p)->RunControlPanel(a,b)
2088 #define IDirectInputDevice8_Initialize(p,a,b,c)         (p)->Initialize(a,b,c)
2089 /*** IDirectInputDevice2 methods ***/
2090 #define IDirectInputDevice8_CreateEffect(p,a,b,c,d)           (p)->CreateEffect(a,b,c,d)
2091 #define IDirectInputDevice8_EnumEffects(p,a,b,c)              (p)->EnumEffects(a,b,c)
2092 #define IDirectInputDevice8_GetEffectInfo(p,a,b)              (p)->GetEffectInfo(a,b)
2093 #define IDirectInputDevice8_GetForceFeedbackState(p,a)        (p)->GetForceFeedbackState(a)
2094 #define IDirectInputDevice8_SendForceFeedbackCommand(p,a)     (p)->SendForceFeedbackCommand(a)
2095 #define IDirectInputDevice8_EnumCreatedEffectObjects(p,a,b,c) (p)->EnumCreatedEffectObjects(a,b,c)
2096 #define IDirectInputDevice8_Escape(p,a)                       (p)->Escape(a)
2097 #define IDirectInputDevice8_Poll(p)                           (p)->Poll()
2098 #define IDirectInputDevice8_SendDeviceData(p,a,b,c,d)         (p)->SendDeviceData(a,b,c,d)
2099 /*** IDirectInputDevice7 methods ***/
2100 #define IDirectInputDevice8_EnumEffectsInFile(p,a,b,c,d)  (p)->EnumEffectsInFile(a,b,c,d)
2101 #define IDirectInputDevice8_WriteEffectToFile(p,a,b,c,d)  (p)->WriteEffectToFile(a,b,c,d)
2102 /*** IDirectInputDevice8 methods ***/
2103 #define IDirectInputDevice8_BuildActionMap(p,a,b,c)  (p)->BuildActionMap(a,b,c)
2104 #define IDirectInputDevice8_SetActionMap(p,a,b,c)    (p)->SetActionMap(a,b,c)
2105 #define IDirectInputDevice8_GetImageInfo(p,a)        (p)->GetImageInfo(a)
2106 #endif
```

```
2107
2108 #endif /* DI8 */
2109
2110 /* "Standard" Mouse report... */
2111 typedef struct DIMOUSESTATE {
2112   LONG lX;
2113   LONG lY;
2114   LONG lZ;
2115   BYTE rgbButtons[4];
2116 } DIMOUSESTATE;
2117
2118 #if DIRECTINPUT_VERSION >= 0x0700
2119 /* "Standard" Mouse report for DInput 7... */
2120 typedef struct DIMOUSESTATE2 {
2121   LONG lX;
2122   LONG lY;
2123   LONG lZ;
2124   BYTE rgbButtons[8];
2125 } DIMOUSESTATE2;
2126 #endif /* DI7 */
2127
2128 #define DIMOFS_X        FIELD_OFFSET(DIMOUSESTATE, lX)
2129 #define DIMOFS_Y        FIELD_OFFSET(DIMOUSESTATE, lY)
2130 #define DIMOFS_Z        FIELD_OFFSET(DIMOUSESTATE, lZ)
2131 #define DIMOFS_BUTTON0 (FIELD_OFFSET(DIMOUSESTATE, rgbButtons) + 0)
2132 #define DIMOFS_BUTTON1 (FIELD_OFFSET(DIMOUSESTATE, rgbButtons) + 1)
2133 #define DIMOFS_BUTTON2 (FIELD_OFFSET(DIMOUSESTATE, rgbButtons) + 2)
2134 #define DIMOFS_BUTTON3 (FIELD_OFFSET(DIMOUSESTATE, rgbButtons) + 3)
2135 #if DIRECTINPUT_VERSION >= 0x0700
2136 #define DIMOFS_BUTTON4 (FIELD_OFFSET(DIMOUSESTATE2, rgbButtons) + 4)
2137 #define DIMOFS_BUTTON5 (FIELD_OFFSET(DIMOUSESTATE2, rgbButtons) + 5)
2138 #define DIMOFS_BUTTON6 (FIELD_OFFSET(DIMOUSESTATE2, rgbButtons) + 6)
2139 #define DIMOFS_BUTTON7 (FIELD_OFFSET(DIMOUSESTATE2, rgbButtons) + 7)
2140 #endif /* DI7 */
2141
2142 #ifdef __cplusplus
2143 extern "C" {
2144 #endif
2145 extern const DIDATAFORMAT c_dfDIMouse;
2146 #if DIRECTINPUT_VERSION >= 0x0700
2147 extern const DIDATAFORMAT c_dfDIMouse2; /* DX 7 */
2148 #endif /* DI7 */
2149 extern const DIDATAFORMAT c_dfDIKeyboard;
2150 #if DIRECTINPUT_VERSION >= 0x0500
2151 extern const DIDATAFORMAT c_dfDIJoystick;
2152 extern const DIDATAFORMAT c_dfDIJoystick2;
2153 #endif /* DI5 */
2154 #ifdef __cplusplus
2155 };
2156 #endif
2157
2158 /*****************************************************************************
2159  * IDirectInputA interface
2160  */
2161 #undef INTERFACE
2162 #define INTERFACE IDirectInputA
2163 DECLARE_INTERFACE_(IDirectInputA,IUnknown)
2164 {
2165     /*** IUnknown methods ***/
2166     STDMETHOD_(HRESULT,QueryInterface)(THIS_ REFIID riid, void** ppvObject) PURE;
2167     STDMETHOD_(ULONG,AddRef)(THIS) PURE;
2168     STDMETHOD_(ULONG,Release)(THIS) PURE;
2169     /*** IDirectInputA methods ***/
2170     STDMETHOD(CreateDevice)(THIS_ REFGUID rguid, LPDIRECTINPUTDEVICEA *lplpDirectInputDevice, LPUNKNOWN
      pUnkOuter) PURE;
2171     STDMETHOD(EnumDevices)(THIS_ DWORD dwDevType, LPDIENUMDEVICESCALLBACKA lpCallback, LPVOID pvRef,
      DWORD dwFlags) PURE;
2172     STDMETHOD(GetDeviceStatus)(THIS_ REFGUID rguidInstance) PURE;
2173     STDMETHOD(RunControlPanel)(THIS_ HWND hwndOwner, DWORD dwFlags) PURE;
2174     STDMETHOD(Initialize)(THIS_ HINSTANCE hinst, DWORD dwVersion) PURE;
2175 };
2176
2177 /*****************************************************************************
2178  * IDirectInputW interface
2179  */
2180 #undef INTERFACE
2181 #define INTERFACE IDirectInputW
2182 DECLARE_INTERFACE_(IDirectInputW,IUnknown)
2183 {
2184     /*** IUnknown methods ***/
2185     STDMETHOD_(HRESULT,QueryInterface)(THIS_ REFIID riid, void** ppvObject) PURE;
2186     STDMETHOD_(ULONG,AddRef)(THIS) PURE;
2187     STDMETHOD_(ULONG,Release)(THIS) PURE;
2188     /*** IDirectInputW methods ***/
2189     STDMETHOD(CreateDevice)(THIS_ REFGUID rguid, LPDIRECTINPUTDEVICEW *lplpDirectInputDevice, LPUNKNOWN
      pUnkOuter) PURE;
2190     STDMETHOD(EnumDevices)(THIS_ DWORD dwDevType, LPDIENUMDEVICESCALLBACKW lpCallback, LPVOID pvRef,
```

```
    DWORD dwFlags) PURE;
2191    STDMETHOD(GetDeviceStatus)(THIS_ REFGUID rguidInstance) PURE;
2192    STDMETHOD(RunControlPanel)(THIS_ HWND hwndOwner, DWORD dwFlags) PURE;
2193    STDMETHOD(Initialize)(THIS_ HINSTANCE hinst, DWORD dwVersion) PURE;
2194 };
2195
2196 #if !defined(__cplusplus) || defined(CINTERFACE)
2197 /*** IUnknown methods ***/
2198 #define IDirectInput_QueryInterface(p,a,b) (p)->lpVtbl->QueryInterface(p,a,b)
2199 #define IDirectInput_AddRef(p)             (p)->lpVtbl->AddRef(p)
2200 #define IDirectInput_Release(p)            (p)->lpVtbl->Release(p)
2201 /*** IDirectInput methods ***/
2202 #define IDirectInput_CreateDevice(p,a,b,c)   (p)->lpVtbl->CreateDevice(p,a,b,c)
2203 #define IDirectInput_EnumDevices(p,a,b,c,d) (p)->lpVtbl->EnumDevices(p,a,b,c,d)
2204 #define IDirectInput_GetDeviceStatus(p,a)    (p)->lpVtbl->GetDeviceStatus(p,a)
2205 #define IDirectInput_RunControlPanel(p,a,b) (p)->lpVtbl->RunControlPanel(p,a,b)
2206 #define IDirectInput_Initialize(p,a,b)       (p)->lpVtbl->Initialize(p,a,b)
2207 #else
2208 /*** IUnknown methods ***/
2209 #define IDirectInput_QueryInterface(p,a,b) (p)->QueryInterface(a,b)
2210 #define IDirectInput_AddRef(p)             (p)->AddRef()
2211 #define IDirectInput_Release(p)            (p)->Release()
2212 /*** IDirectInput methods ***/
2213 #define IDirectInput_CreateDevice(p,a,b,c)   (p)->CreateDevice(a,b,c)
2214 #define IDirectInput_EnumDevices(p,a,b,c,d) (p)->EnumDevices(a,b,c,d)
2215 #define IDirectInput_GetDeviceStatus(p,a)    (p)->GetDeviceStatus(a)
2216 #define IDirectInput_RunControlPanel(p,a,b) (p)->RunControlPanel(a,b)
2217 #define IDirectInput_Initialize(p,a,b)       (p)->Initialize(a,b)
2218 #endif
2219
2220 /*****************************************************************************
2221  * IDirectInput2A interface
2222  */
2223 #undef INTERFACE
2224 #define INTERFACE IDirectInput2A
2225 DECLARE_INTERFACE_(IDirectInput2A,IDirectInputA)
2226 {
2227     /*** IUnknown methods ***/
2228     STDMETHOD_(HRESULT,QueryInterface)(THIS_ REFIID riid, void** ppvObject) PURE;
2229     STDMETHOD_(ULONG,AddRef)(THIS) PURE;
2230     STDMETHOD_(ULONG,Release)(THIS) PURE;
2231     /*** IDirectInputA methods ***/
2232     STDMETHOD(CreateDevice)(THIS_ REFGUID rguid, LPDIRECTINPUTDEVICEA *lplpDirectInputDevice, LPUNKNOWN
    pUnkOuter) PURE;
2233     STDMETHOD(EnumDevices)(THIS_ DWORD dwDevType, LPDIENUMDEVICESCALLBACKA lpCallback, LPVOID pvRef,
    DWORD dwFlags) PURE;
2234     STDMETHOD(GetDeviceStatus)(THIS_ REFGUID rguidInstance) PURE;
2235     STDMETHOD(RunControlPanel)(THIS_ HWND hwndOwner, DWORD dwFlags) PURE;
2236     STDMETHOD(Initialize)(THIS_ HINSTANCE hinst, DWORD dwVersion) PURE;
2237     /*** IDirectInput2A methods ***/
2238     STDMETHOD(FindDevice)(THIS_ REFGUID rguid, LPCSTR pszName, LPGUID pguidInstance) PURE;
2239 };
2240
2241 /*****************************************************************************
2242  * IDirectInput2W interface
2243  */
2244 #undef INTERFACE
2245 #define INTERFACE IDirectInput2W
2246 DECLARE_INTERFACE_(IDirectInput2W,IDirectInputW)
2247 {
2248     /*** IUnknown methods ***/
2249     STDMETHOD_(HRESULT,QueryInterface)(THIS_ REFIID riid, void** ppvObject) PURE;
2250     STDMETHOD_(ULONG,AddRef)(THIS) PURE;
2251     STDMETHOD_(ULONG,Release)(THIS) PURE;
2252     /*** IDirectInputW methods ***/
2253     STDMETHOD(CreateDevice)(THIS_ REFGUID rguid, LPDIRECTINPUTDEVICEW *lplpDirectInputDevice, LPUNKNOWN
    pUnkOuter) PURE;
2254     STDMETHOD(EnumDevices)(THIS_ DWORD dwDevType, LPDIENUMDEVICESCALLBACKW lpCallback, LPVOID pvRef,
    DWORD dwFlags) PURE;
2255     STDMETHOD(GetDeviceStatus)(THIS_ REFGUID rguidInstance) PURE;
2256     STDMETHOD(RunControlPanel)(THIS_ HWND hwndOwner, DWORD dwFlags) PURE;
2257     STDMETHOD(Initialize)(THIS_ HINSTANCE hinst, DWORD dwVersion) PURE;
2258     /*** IDirectInput2W methods ***/
2259     STDMETHOD(FindDevice)(THIS_ REFGUID rguid, LPCWSTR pszName, LPGUID pguidInstance) PURE;
2260 };
2261
2262 #if !defined(__cplusplus) || defined(CINTERFACE)
2263 /*** IUnknown methods ***/
2264 #define IDirectInput2_QueryInterface(p,a,b) (p)->lpVtbl->QueryInterface(p,a,b)
2265 #define IDirectInput2_AddRef(p)             (p)->lpVtbl->AddRef(p)
2266 #define IDirectInput2_Release(p)            (p)->lpVtbl->Release(p)
2267 /*** IDirectInput methods ***/
2268 #define IDirectInput2_CreateDevice(p,a,b,c)   (p)->lpVtbl->CreateDevice(p,a,b,c)
2269 #define IDirectInput2_EnumDevices(p,a,b,c,d) (p)->lpVtbl->EnumDevices(p,a,b,c,d)
2270 #define IDirectInput2_GetDeviceStatus(p,a)    (p)->lpVtbl->GetDeviceStatus(p,a)
2271 #define IDirectInput2_RunControlPanel(p,a,b) (p)->lpVtbl->RunControlPanel(p,a,b)
2272 #define IDirectInput2_Initialize(p,a,b)       (p)->lpVtbl->Initialize(p,a,b)
```

```
2273 /*** IDirectInput2 methods ***/
2274 #define IDirectInput2_FindDevice(p,a,b,c)     (p)->lpVtbl->FindDevice(p,a,b,c)
2275 #else
2276 /*** IUnknown methods ***/
2277 #define IDirectInput2_QueryInterface(p,a,b) (p)->QueryInterface(a,b)
2278 #define IDirectInput2_AddRef(p)             (p)->AddRef()
2279 #define IDirectInput2_Release(p)            (p)->Release()
2280 /*** IDirectInput methods ***/
2281 #define IDirectInput2_CreateDevice(p,a,b,c)   (p)->CreateDevice(a,b,c)
2282 #define IDirectInput2_EnumDevices(p,a,b,c,d) (p)->EnumDevices(a,b,c,d)
2283 #define IDirectInput2_GetDeviceStatus(p,a)    (p)->GetDeviceStatus(a)
2284 #define IDirectInput2_RunControlPanel(p,a,b) (p)->RunControlPanel(a,b)
2285 #define IDirectInput2_Initialize(p,a,b)       (p)->Initialize(a,b)
2286 /*** IDirectInput2 methods ***/
2287 #define IDirectInput2_FindDevice(p,a,b,c)     (p)->FindDevice(a,b,c)
2288 #endif
2289
2290 /*****************************************************************************
2291  * IDirectInput7A interface
2292  */
2293 #undef INTERFACE
2294 #define INTERFACE IDirectInput7A
2295 DECLARE_INTERFACE_(IDirectInput7A,IDirectInput2A)
2296 {
2297     /*** IUnknown methods ***/
2298     STDMETHOD_(HRESULT,QueryInterface)(THIS_ REFIID riid, void** ppvObject) PURE;
2299     STDMETHOD_(ULONG,AddRef)(THIS) PURE;
2300     STDMETHOD_(ULONG,Release)(THIS) PURE;
2301     /*** IDirectInputA methods ***/
2302     STDMETHOD(CreateDevice)(THIS_ REFGUID rguid, LPDIRECTINPUTDEVICEA *lplpDirectInputDevice, LPUNKNOWN
      pUnkOuter) PURE;
2303     STDMETHOD(EnumDevices)(THIS_ DWORD dwDevType, LPDIENUMDEVICESCALLBACKA lpCallback, LPVOID pvRef,
      DWORD dwFlags) PURE;
2304     STDMETHOD(GetDeviceStatus)(THIS_ REFGUID rguidInstance) PURE;
2305     STDMETHOD(RunControlPanel)(THIS_ HWND hwndOwner, DWORD dwFlags) PURE;
2306     STDMETHOD(Initialize)(THIS_ HINSTANCE hinst, DWORD dwVersion) PURE;
2307     /*** IDirectInput2A methods ***/
2308     STDMETHOD(FindDevice)(THIS_ REFGUID rguid, LPCSTR pszName, LPGUID pguidInstance) PURE;
2309     /*** IDirectInput7A methods ***/
2310     STDMETHOD(CreateDeviceEx)(THIS_ REFGUID rguid, REFIID riid, LPVOID *pvOut, LPUNKNOWN
      lpUnknownOuter) PURE;
2311 };
2312
2313 /*****************************************************************************
2314  * IDirectInput7W interface
2315  */
2316 #undef INTERFACE
2317 #define INTERFACE IDirectInput7W
2318 DECLARE_INTERFACE_(IDirectInput7W,IDirectInput2W)
2319 {
2320     /*** IUnknown methods ***/
2321     STDMETHOD_(HRESULT,QueryInterface)(THIS_ REFIID riid, void** ppvObject) PURE;
2322     STDMETHOD_(ULONG,AddRef)(THIS) PURE;
2323     STDMETHOD_(ULONG,Release)(THIS) PURE;
2324     /*** IDirectInputW methods ***/
2325     STDMETHOD(CreateDevice)(THIS_ REFGUID rguid, LPDIRECTINPUTDEVICEW *lplpDirectInputDevice, LPUNKNOWN
      pUnkOuter) PURE;
2326     STDMETHOD(EnumDevices)(THIS_ DWORD dwDevType, LPDIENUMDEVICESCALLBACKW lpCallback, LPVOID pvRef,
      DWORD dwFlags) PURE;
2327     STDMETHOD(GetDeviceStatus)(THIS_ REFGUID rguidInstance) PURE;
2328     STDMETHOD(RunControlPanel)(THIS_ HWND hwndOwner, DWORD dwFlags) PURE;
2329     STDMETHOD(Initialize)(THIS_ HINSTANCE hinst, DWORD dwVersion) PURE;
2330     /*** IDirectInput2W methods ***/
2331     STDMETHOD(FindDevice)(THIS_ REFGUID rguid, LPCWSTR pszName, LPGUID pguidInstance) PURE;
2332     /*** IDirectInput7W methods ***/
2333     STDMETHOD(CreateDeviceEx)(THIS_ REFGUID rguid, REFIID riid, LPVOID *pvOut, LPUNKNOWN
      lpUnknownOuter) PURE;
2334 };
2335
2336 #if !defined(__cplusplus) || defined(CINTERFACE)
2337 /*** IUnknown methods ***/
2338 #define IDirectInput7_QueryInterface(p,a,b) (p)->lpVtbl->QueryInterface(p,a,b)
2339 #define IDirectInput7_AddRef(p)             (p)->lpVtbl->AddRef(p)
2340 #define IDirectInput7_Release(p)            (p)->lpVtbl->Release(p)
2341 /*** IDirectInput methods ***/
2342 #define IDirectInput7_CreateDevice(p,a,b,c)   (p)->lpVtbl->CreateDevice(p,a,b,c)
2343 #define IDirectInput7_EnumDevices(p,a,b,c,d) (p)->lpVtbl->EnumDevices(p,a,b,c,d)
2344 #define IDirectInput7_GetDeviceStatus(p,a)    (p)->lpVtbl->GetDeviceStatus(p,a)
2345 #define IDirectInput7_RunControlPanel(p,a,b) (p)->lpVtbl->RunControlPanel(p,a,b)
2346 #define IDirectInput7_Initialize(p,a,b)       (p)->lpVtbl->Initialize(p,a,b)
2347 /*** IDirectInput2 methods ***/
2348 #define IDirectInput7_FindDevice(p,a,b,c)     (p)->lpVtbl->FindDevice(p,a,b,c)
2349 /*** IDirectInput7 methods ***/
2350 #define IDirectInput7_CreateDeviceEx(p,a,b,c,d) (p)->lpVtbl->CreateDeviceEx(p,a,b,c,d)
2351 #else
2352 /*** IUnknown methods ***/
2353 #define IDirectInput7_QueryInterface(p,a,b) (p)->QueryInterface(a,b)
```

```
2354 #define IDirectInput7_AddRef(p)              (p)->AddRef()
2355 #define IDirectInput7_Release(p)             (p)->Release()
2356 /*** IDirectInput methods ***/
2357 #define IDirectInput7_CreateDevice(p,a,b,c)   (p)->CreateDevice(a,b,c)
2358 #define IDirectInput7_EnumDevices(p,a,b,c,d)  (p)->EnumDevices(a,b,c,d)
2359 #define IDirectInput7_GetDeviceStatus(p,a)    (p)->GetDeviceStatus(a)
2360 #define IDirectInput7_RunControlPanel(p,a,b)  (p)->RunControlPanel(a,b)
2361 #define IDirectInput7_Initialize(p,a,b)       (p)->Initialize(a,b)
2362 /*** IDirectInput2 methods ***/
2363 #define IDirectInput7_FindDevice(p,a,b,c)     (p)->FindDevice(a,b,c)
2364 /*** IDirectInput7 methods ***/
2365 #define IDirectInput7_CreateDeviceEx(p,a,b,c,d)  (p)->CreateDeviceEx(a,b,c,d)
2366 #endif
2367
2368
2369 #if DIRECTINPUT_VERSION >= 0x0800
2370 /*****************************************************************************
2371  * IDirectInput8A interface
2372  */
2373 #undef INTERFACE
2374 #define INTERFACE IDirectInput8A
2375 DECLARE_INTERFACE_(IDirectInput8A,IUnknown)
2376 {
2377     /*** IUnknown methods ***/
2378     STDMETHOD_(HRESULT,QueryInterface)(THIS_ REFIID riid, void** ppvObject) PURE;
2379     STDMETHOD_(ULONG,AddRef)(THIS) PURE;
2380     STDMETHOD_(ULONG,Release)(THIS) PURE;
2381     /*** IDirectInput8A methods ***/
2382     STDMETHOD(CreateDevice)(THIS_ REFGUID rguid, LPDIRECTINPUTDEVICE8A *lplpDirectInputDevice,
    LPUNKNOWN pUnkOuter) PURE;
2383     STDMETHOD(EnumDevices)(THIS_ DWORD dwDevType, LPDIENUMDEVICESCALLBACKA lpCallback, LPVOID pvRef,
    DWORD dwFlags) PURE;
2384     STDMETHOD(GetDeviceStatus)(THIS_ REFGUID rguidInstance) PURE;
2385     STDMETHOD(RunControlPanel)(THIS_ HWND hwndOwner, DWORD dwFlags) PURE;
2386     STDMETHOD(Initialize)(THIS_ HINSTANCE hinst, DWORD dwVersion) PURE;
2387     STDMETHOD(FindDevice)(THIS_ REFGUID rguid, LPCSTR pszName, LPGUID pguidInstance) PURE;
2388     STDMETHOD(EnumDevicesBySemantics)(THIS_ LPCSTR ptszUserName, LPDIACTIONFORMATA lpdiActionFormat,
    LPDIENUMDEVICESBYSEMANTICSCBA lpCallback, LPVOID pvRef, DWORD dwFlags) PURE;
2389     STDMETHOD(ConfigureDevices)(THIS_ LPDICONFIGUREDEVICESCALLBACK lpdiCallback,
    LPDICONFIGUREDEVICESPARAMSA lpdiCDParams, DWORD dwFlags, LPVOID pvRefData) PURE;
2390 };
2391
2392 /*****************************************************************************
2393  * IDirectInput8W interface
2394  */
2395 #undef INTERFACE
2396 #define INTERFACE IDirectInput8W
2397 DECLARE_INTERFACE_(IDirectInput8W,IUnknown)
2398 {
2399     /*** IUnknown methods ***/
2400     STDMETHOD_(HRESULT,QueryInterface)(THIS_ REFIID riid, void** ppvObject) PURE;
2401     STDMETHOD_(ULONG,AddRef)(THIS) PURE;
2402     STDMETHOD_(ULONG,Release)(THIS) PURE;
2403     /*** IDirectInput8W methods ***/
2404     STDMETHOD(CreateDevice)(THIS_ REFGUID rguid, LPDIRECTINPUTDEVICE8W *lplpDirectInputDevice,
    LPUNKNOWN pUnkOuter) PURE;
2405     STDMETHOD(EnumDevices)(THIS_ DWORD dwDevType, LPDIENUMDEVICESCALLBACKW lpCallback, LPVOID pvRef,
    DWORD dwFlags) PURE;
2406     STDMETHOD(GetDeviceStatus)(THIS_ REFGUID rguidInstance) PURE;
2407     STDMETHOD(RunControlPanel)(THIS_ HWND hwndOwner, DWORD dwFlags) PURE;
2408     STDMETHOD(Initialize)(THIS_ HINSTANCE hinst, DWORD dwVersion) PURE;
2409     STDMETHOD(FindDevice)(THIS_ REFGUID rguid, LPCWSTR pszName, LPGUID pguidInstance) PURE;
2410     STDMETHOD(EnumDevicesBySemantics)(THIS_ LPCWSTR ptszUserName, LPDIACTIONFORMATW lpdiActionFormat,
    LPDIENUMDEVICESBYSEMANTICSCBW lpCallback, LPVOID pvRef, DWORD dwFlags) PURE;
2411     STDMETHOD(ConfigureDevices)(THIS_ LPDICONFIGUREDEVICESCALLBACK lpdiCallback,
    LPDICONFIGUREDEVICESPARAMSW lpdiCDParams, DWORD dwFlags, LPVOID pvRefData) PURE;
2412 };
2413 #undef INTERFACE
2414
2415 #if !defined(__cplusplus) || defined(CINTERFACE)
2416 /*** IUnknown methods ***/
2417 #define IDirectInput8_QueryInterface(p,a,b) (p)->lpVtbl->QueryInterface(p,a,b)
2418 #define IDirectInput8_AddRef(p)             (p)->lpVtbl->AddRef(p)
2419 #define IDirectInput8_Release(p)            (p)->lpVtbl->Release(p)
2420 /*** IDirectInput8 methods ***/
2421 #define IDirectInput8_CreateDevice(p,a,b,c)       (p)->lpVtbl->CreateDevice(p,a,b,c)
2422 #define IDirectInput8_EnumDevices(p,a,b,c,d)      (p)->lpVtbl->EnumDevices(p,a,b,c,d)
2423 #define IDirectInput8_GetDeviceStatus(p,a)        (p)->lpVtbl->GetDeviceStatus(p,a)
2424 #define IDirectInput8_RunControlPanel(p,a,b)      (p)->lpVtbl->RunControlPanel(p,a,b)
2425 #define IDirectInput8_Initialize(p,a,b)           (p)->lpVtbl->Initialize(p,a,b)
2426 #define IDirectInput8_FindDevice(p,a,b,c)         (p)->lpVtbl->FindDevice(p,a,b,c)
2427 #define IDirectInput8_EnumDevicesBySemantics(p,a,b,c,d,e)
    (p)->lpVtbl->EnumDevicesBySemantics(p,a,b,c,d,e)
2428 #define IDirectInput8_ConfigureDevices(p,a,b,c,d) (p)->lpVtbl->ConfigureDevices(p,a,b,c,d)
2429 #else
2430 /*** IUnknown methods ***/
2431 #define IDirectInput8_QueryInterface(p,a,b) (p)->QueryInterface(a,b)
```

```
2432 #define IDirectInput8_AddRef(p)             (p)->AddRef()
2433 #define IDirectInput8_Release(p)            (p)->Release()
2434 /*** IDirectInput8 methods ***/
2435 #define IDirectInput8_CreateDevice(p,a,b,c)        (p)->CreateDevice(a,b,c)
2436 #define IDirectInput8_EnumDevices(p,a,b,c,d)       (p)->EnumDevices(a,b,c,d)
2437 #define IDirectInput8_GetDeviceStatus(p,a)         (p)->GetDeviceStatus(a)
2438 #define IDirectInput8_RunControlPanel(p,a,b)       (p)->RunControlPanel(a,b)
2439 #define IDirectInput8_Initialize(p,a,b)            (p)->Initialize(a,b)
2440 #define IDirectInput8_FindDevice(p,a,b,c)          (p)->FindDevice(a,b,c)
2441 #define IDirectInput8_EnumDevicesBySemantics(p,a,b,c,d,e) (p)->EnumDevicesBySemantics(a,b,c,d,e)
2442 #define IDirectInput8_ConfigureDevices(p,a,b,c,d) (p)->ConfigureDevices(a,b,c,d)
2443 #endif
2444
2445 #endif /* DI8 */
2446
2447 /* Export functions */
2448
2449 #ifdef __cplusplus
2450 extern "C" {
2451 #endif
2452
2453 #if DIRECTINPUT_VERSION >= 0x0800
2454 HRESULT WINAPI DirectInput8Create(HINSTANCE,DWORD,REFIID,LPVOID *,LPUNKNOWN);
2455 #else /* DI < 8 */
2456 HRESULT WINAPI DirectInputCreateA(HINSTANCE,DWORD,LPDIRECTINPUTA *,LPUNKNOWN);
2457 HRESULT WINAPI DirectInputCreateW(HINSTANCE,DWORD,LPDIRECTINPUTW *,LPUNKNOWN);
2458 #define DirectInputCreate WINELIB_NAME_AW(DirectInputCreate)
2459
2460 HRESULT WINAPI DirectInputCreateEx(HINSTANCE,DWORD,REFIID,LPVOID *,LPUNKNOWN);
2461 #endif /* DI8 */
2462
2463 #ifdef __cplusplus
2464 };
2465 #endif
2466
2467 #endif /* __DINPUT_INCLUDED__ */
```

27.8 xinput.h

```
1 /*
2  * The Wine project - Xinput Joystick Library
3  * Copyright 2008 Andrew Fenn
4  *
5  * This library is free software; you can redistribute it and/or
6  * modify it under the terms of the GNU Lesser General Public
7  * License as published by the Free Software Foundation; either
8  * version 2.1 of the License, or (at your option) any later version.
9  *
10  * This library is distributed in the hope that it will be useful,
11  * but WITHOUT ANY WARRANTY; without even the implied warranty of
12  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
13  * Lesser General Public License for more details.
14  *
15  * You should have received a copy of the GNU Lesser General Public
16  * License along with this library; if not, write to the Free Software
17  * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA
18  */
19
20 #ifndef __WINE_XINPUT_H
21 #define __WINE_XINPUT_H
22
23 #include <windef.h>
24
25 /*
26  * Bitmasks for the joysticks buttons, determines what has
27  * been pressed on the joystick, these need to be mapped
28  * to whatever device you're using instead of an xbox 360
29  * joystick
30  */
31
32 #define XINPUT_GAMEPAD_DPAD_UP          0x0001
33 #define XINPUT_GAMEPAD_DPAD_DOWN        0x0002
34 #define XINPUT_GAMEPAD_DPAD_LEFT        0x0004
35 #define XINPUT_GAMEPAD_DPAD_RIGHT       0x0008
36 #define XINPUT_GAMEPAD_START            0x0010
37 #define XINPUT_GAMEPAD_BACK             0x0020
38 #define XINPUT_GAMEPAD_LEFT_THUMB       0x0040
39 #define XINPUT_GAMEPAD_RIGHT_THUMB      0x0080
40 #define XINPUT_GAMEPAD_LEFT_SHOULDER    0x0100
41 #define XINPUT_GAMEPAD_RIGHT_SHOULDER   0x0200
42 #define XINPUT_GAMEPAD_A                0x1000
43 #define XINPUT_GAMEPAD_B                0x2000
44 #define XINPUT_GAMEPAD_X                0x4000
```

```
45 #define XINPUT_GAMEPAD_Y                0x8000
46
47 /*
48  * Defines the flags used to determine if the user is pushing
49  * down on a button, not holding a button, etc
50  */
51
52 #define XINPUT_KEYSTROKE_KEYDOWN        0x0001
53 #define XINPUT_KEYSTROKE_KEYUP          0x0002
54 #define XINPUT_KEYSTROKE_REPEAT         0x0004
55
56 /*
57  * Defines the codes which are returned by XInputGetKeystroke
58  */
59
60 #define VK_PAD_A                        0x5800
61 #define VK_PAD_B                        0x5801
62 #define VK_PAD_X                        0x5802
63 #define VK_PAD_Y                        0x5803
64 #define VK_PAD_RSHOULDER                0x5804
65 #define VK_PAD_LSHOULDER                0x5805
66 #define VK_PAD_LTRIGGER                 0x5806
67 #define VK_PAD_RTRIGGER                 0x5807
68 #define VK_PAD_DPAD_UP                  0x5810
69 #define VK_PAD_DPAD_DOWN                0x5811
70 #define VK_PAD_DPAD_LEFT                0x5812
71 #define VK_PAD_DPAD_RIGHT               0x5813
72 #define VK_PAD_START                    0x5814
73 #define VK_PAD_BACK                     0x5815
74 #define VK_PAD_LTHUMB_PRESS             0x5816
75 #define VK_PAD_RTHUMB_PRESS             0x5817
76 #define VK_PAD_LTHUMB_UP                0x5820
77 #define VK_PAD_LTHUMB_DOWN              0x5821
78 #define VK_PAD_LTHUMB_RIGHT             0x5822
79 #define VK_PAD_LTHUMB_LEFT              0x5823
80 #define VK_PAD_LTHUMB_UPLEFT            0x5824
81 #define VK_PAD_LTHUMB_UPRIGHT           0x5825
82 #define VK_PAD_LTHUMB_DOWNRIGHT         0x5826
83 #define VK_PAD_LTHUMB_DOWNLEFT          0x5827
84 #define VK_PAD_RTHUMB_UP                0x5830
85 #define VK_PAD_RTHUMB_DOWN              0x5831
86 #define VK_PAD_RTHUMB_RIGHT             0x5832
87 #define VK_PAD_RTHUMB_LEFT              0x5833
88 #define VK_PAD_RTHUMB_UPLEFT            0x5834
89 #define VK_PAD_RTHUMB_UPRIGHT           0x5835
90 #define VK_PAD_RTHUMB_DOWNRIGHT         0x5836
91 #define VK_PAD_RTHUMB_DOWNLEFT          0x5837
92
93 /*
94  * Deadzones are for analogue joystick controls on the joypad
95  * which determine when input should be assumed to be in the
96  * middle of the pad. This is a threshold to stop a joypad
97  * controlling the game when the player isn't touching the
98  * controls.
99  */
100
101 #define XINPUT_GAMEPAD_LEFT_THUMB_DEADZONE  7849
102 #define XINPUT_GAMEPAD_RIGHT_THUMB_DEADZONE 8689
103 #define XINPUT_GAMEPAD_TRIGGER_THRESHOLD    30
104
105
106 /*
107  * Defines what type of abilities the type of joystick has
108  * DEVTYPE_GAMEPAD is available for all joysticks, however
109  * there may be more specific identifiers for other joysticks
110  * which are being used.
111  */
112
113 #define XINPUT_DEVTYPE_GAMEPAD          0x01
114 #define XINPUT_DEVSUBTYPE_GAMEPAD       0x01
115 #define XINPUT_DEVSUBTYPE_WHEEL         0x02
116 #define XINPUT_DEVSUBTYPE_ARCADE_STICK  0x03
117 #define XINPUT_DEVSUBTYPE_FLIGHT_SICK   0x04
118 #define XINPUT_DEVSUBTYPE_DANCE_PAD     0x05
119 #define XINPUT_DEVSUBTYPE_GUITAR        0x06
120 #define XINPUT_DEVSUBTYPE_DRUM_KIT      0x08
121
122 /*
123  * These are used with the XInputGetCapabilities function to
124  * determine the abilities to the joystick which has been
125  * plugged in.
126  */
127
128 #define XINPUT_CAPS_VOICE_SUPPORTED     0x0004
129 #define XINPUT_FLAG_GAMEPAD             0x00000001
130
131 /*
```

```
132  * Defines the status of the battery if one is used in the
133  * attached joystick. The first two define if the joystick
134  * supports a battery. Disconnected means that the joystick
135  * isn't connected. Wired shows that the joystick is a wired
136  * joystick.
137  */
138
139 #define BATTERY_DEVTYPE_GAMEPAD         0x00
140 #define BATTERY_DEVTYPE_HEADSET         0x01
141 #define BATTERY_TYPE_DISCONNECTED       0x00
142 #define BATTERY_TYPE_WIRED              0x01
143 #define BATTERY_TYPE_ALKALINE           0x02
144 #define BATTERY_TYPE_NIMH               0x03
145 #define BATTERY_TYPE_UNKNOWN            0xFF
146 #define BATTERY_LEVEL_EMPTY             0x00
147 #define BATTERY_LEVEL_LOW               0x01
148 #define BATTERY_LEVEL_MEDIUM            0x02
149 #define BATTERY_LEVEL_FULL              0x03
150
151 /*
152  * How many joysticks can be used with this library. Games that
153  * use the xinput library will not go over this number.
154  */
155
156 #define XUSER_MAX_COUNT                 4
157 #define XUSER_INDEX_ANY                 0x000000FF
158
159 /*
160  * Defines the structure of an xbox 360 joystick.
161  */
162
163 typedef struct _XINPUT_GAMEPAD {
164     WORD wButtons;
165     BYTE bLeftTrigger;
166     BYTE bRightTrigger;
167     SHORT sThumbLX;
168     SHORT sThumbLY;
169     SHORT sThumbRX;
170     SHORT sThumbRY;
171 } XINPUT_GAMEPAD, *PXINPUT_GAMEPAD;
172
173 typedef struct _XINPUT_STATE {
174     DWORD dwPacketNumber;
175     XINPUT_GAMEPAD Gamepad;
176 } XINPUT_STATE, *PXINPUT_STATE;
177
178 /*
179  * Defines the structure of how much vibration is set on both the
180  * right and left motors in a joystick. If you're not using a 360
181  * joystick you will have to map these to your device.
182  */
183
184 typedef struct _XINPUT_VIBRATION {
185     WORD wLeftMotorSpeed;
186     WORD wRightMotorSpeed;
187 } XINPUT_VIBRATION, *PXINPUT_VIBRATION;
188
189 /*
190  * Defines the structure for what kind of abilities the joystick has
191  * such abilities are things such as if the joystick has the ability
192  * to send and receive audio, if the joystick is in fact a driving
193  * wheel or perhaps if the joystick is some kind of dance pad or
194  * guitar.
195  */
196
197 typedef struct _XINPUT_CAPABILITIES {
198     BYTE Type;
199     BYTE SubType;
200     WORD Flags;
201     XINPUT_GAMEPAD Gamepad;
202     XINPUT_VIBRATION Vibration;
203 } XINPUT_CAPABILITIES, *PXINPUT_CAPABILITIES;
204
205 /*
206  * Defines the structure for a joystick input event which is
207  * retrieved using the function XInputGetKeystroke
208  */
209 typedef struct _XINPUT_KEYSTROKE {
210     WORD VirtualKey;
211     WCHAR Unicode;
212     WORD Flags;
213     BYTE UserIndex;
214     BYTE HidCode;
215 } XINPUT_KEYSTROKE, *PXINPUT_KEYSTROKE;
216
217 typedef struct _XINPUT_BATTERY_INFORMATION
218 {
```

```
219     BYTE BatteryType;
220     BYTE BatteryLevel;
221 } XINPUT_BATTERY_INFORMATION, *PXINPUT_BATTERY_INFORMATION;
222
223 #ifdef __cplusplus
224 extern "C" {
225 #endif
226
227 void WINAPI XInputEnable(WINBOOL);
228 DWORD WINAPI XInputSetState(DWORD, XINPUT_VIBRATION*);
229 DWORD WINAPI XInputGetState(DWORD, XINPUT_STATE*);
230 DWORD WINAPI XInputGetKeystroke(DWORD, DWORD, PXINPUT_KEYSTROKE);
231 DWORD WINAPI XInputGetCapabilities(DWORD, DWORD, XINPUT_CAPABILITIES*);
232 DWORD WINAPI XInputGetDSoundAudioDeviceGuids(DWORD, GUID*, GUID*);
233 DWORD WINAPI XInputGetBatteryInformation(DWORD, BYTE, XINPUT_BATTERY_INFORMATION*);
234
235 #ifdef __cplusplus
236 }
237 #endif
238
239 #endif /* __WINE_XINPUT_H */
```

27.9 nuklear.h

```
1 /*
215 */
216 #ifndef NK_SINGLE_FILE
217   #define NK_SINGLE_FILE
218 #endif
219
220 #ifndef NK_NUKLEAR_H_
221 #define NK_NUKLEAR_H_
222
223 #ifdef __cplusplus
224 extern "C" {
225 #endif
226 /*
227  * ==============================================================
228  *
229  *                          CONSTANTS
230  *
231  * ===============================================================
232  */
233 #define NK_UNDEFINED (-1.0f)
234 #define NK_UTF_INVALID 0xFFFD /* internal invalid utf8 rune */
235 #define NK_UTF_SIZE 4 /* describes the number of bytes a glyph consists of*/
236 #ifndef NK_INPUT_MAX
237   #define NK_INPUT_MAX 16
238 #endif
239 #ifndef NK_MAX_NUMBER_BUFFER
240   #define NK_MAX_NUMBER_BUFFER 64
241 #endif
242 #ifndef NK_SCROLLBAR_HIDING_TIMEOUT
243   #define NK_SCROLLBAR_HIDING_TIMEOUT 4.0f
244 #endif
245 /*
246  * ==============================================================
247  *
248  *                          HELPER
249  *
250  * ===============================================================
251  */
252 #ifndef NK_API
253   #ifdef NK_PRIVATE
254     #if (defined(__STDC_VERSION__) && (__STDC_VERSION__ >= 199409L))
255       #define NK_API static inline
256     #elif defined(__cplusplus)
257       #define NK_API static inline
258     #else
259       #define NK_API static
260     #endif
261   #else
262     #define NK_API extern
263   #endif
264 #endif
265 #ifndef NK_LIB
266   #ifdef NK_SINGLE_FILE
267     #define NK_LIB static
268   #else
269     #define NK_LIB extern
270   #endif
271 #endif
272
```

```
273 #define NK_INTERN static
274 #define NK_STORAGE static
275 #define NK_GLOBAL static
276
277 #define NK_FLAG(x) (1 « (x))
278 #define NK_STRINGIFY(x) #x
279 #define NK_MACRO_STRINGIFY(x) NK_STRINGIFY(x)
280 #define NK_STRING_JOIN_IMMEDIATE(arg1, arg2) arg1 ## arg2
281 #define NK_STRING_JOIN_DELAY(arg1, arg2) NK_STRING_JOIN_IMMEDIATE(arg1, arg2)
282 #define NK_STRING_JOIN(arg1, arg2) NK_STRING_JOIN_DELAY(arg1, arg2)
283
284 #ifdef _MSC_VER
285   #define NK_UNIQUE_NAME(name) NK_STRING_JOIN(name,__COUNTER__)
286 #else
287   #define NK_UNIQUE_NAME(name) NK_STRING_JOIN(name,__LINE__)
288 #endif
289
290 #ifndef NK_STATIC_ASSERT
291   #define NK_STATIC_ASSERT(exp) typedef char NK_UNIQUE_NAME(_dummy_array)[(exp)?1:-1]
292 #endif
293
294 #ifndef NK_FILE_LINE
295 #ifdef _MSC_VER
296   #define NK_FILE_LINE __FILE__ ":" NK_MACRO_STRINGIFY(__COUNTER__)
297 #else
298   #define NK_FILE_LINE __FILE__ ":" NK_MACRO_STRINGIFY(__LINE__)
299 #endif
300 #endif
301
302 #define NK_MIN(a,b) ((a) < (b) ? (a) : (b))
303 #define NK_MAX(a,b) ((a) < (b) ? (b) : (a))
304 #define NK_CLAMP(i,v,x) (NK_MAX(NK_MIN(v,x), i))
305
306 #ifdef NK_INCLUDE_STANDARD_VARARGS
307   #include <stdarg.h> /* valist, va_start, va_end, ... */
308   #if defined(_MSC_VER) && (_MSC_VER >= 1600) /* VS 2010 and above */
309     #include <sal.h>
310     #define NK_PRINTF_FORMAT_STRING _Printf_format_string_
311   #else
312     #define NK_PRINTF_FORMAT_STRING
313   #endif
314   #if defined(__GNUC__)
315     #define NK_PRINTF_VARARG_FUNC(fmtargnumber) __attribute__((format(__printf__, fmtargnumber,
     fmtargnumber+1)))
316     #define NK_PRINTF_VALIST_FUNC(fmtargnumber) __attribute__((format(__printf__, fmtargnumber, 0)))
317   #else
318     #define NK_PRINTF_VARARG_FUNC(fmtargnumber)
319     #define NK_PRINTF_VALIST_FUNC(fmtargnumber)
320   #endif
321 #endif
322
323 /*
324  * ==============================================================
325  *
326  *                          BASIC
327  *
328  * ==============================================================
329  */
330 #ifdef NK_INCLUDE_FIXED_TYPES
331  #include <stdint.h>
332  #define NK_INT8 int8_t
333  #define NK_UINT8 uint8_t
334  #define NK_INT16 int16_t
335  #define NK_UINT16 uint16_t
336  #define NK_INT32 int32_t
337  #define NK_UINT32 uint32_t
338  #define NK_SIZE_TYPE uintptr_t
339  #define NK_POINTER_TYPE uintptr_t
340 #else
341   #ifndef NK_INT8
342     #define NK_INT8 signed char
343   #endif
344   #ifndef NK_UINT8
345     #define NK_UINT8 unsigned char
346   #endif
347   #ifndef NK_INT16
348     #define NK_INT16 signed short
349   #endif
350   #ifndef NK_UINT16
351     #define NK_UINT16 unsigned short
352   #endif
353   #ifndef NK_INT32
354     #if defined(_MSC_VER)
355       #define NK_INT32 __int32
356     #else
357       #define NK_INT32 signed int
358     #endif
```

```
359   #endif
360   #ifndef NK_UINT32
361     #if defined(_MSC_VER)
362       #define NK_UINT32 unsigned __int32
363     #else
364       #define NK_UINT32 unsigned int
365     #endif
366   #endif
367   #ifndef NK_SIZE_TYPE
368     #if defined(_WIN64) && defined(_MSC_VER)
369       #define NK_SIZE_TYPE unsigned __int64
370     #elif (defined(_WIN32) || defined(WIN32)) && defined(_MSC_VER)
371       #define NK_SIZE_TYPE unsigned __int32
372     #elif defined(__GNUC__) || defined(__clang__)
373       #if defined(__x86_64__) || defined(__ppc64__)
374         #define NK_SIZE_TYPE unsigned long
375       #else
376         #define NK_SIZE_TYPE unsigned int
377       #endif
378     #else
379       #define NK_SIZE_TYPE unsigned long
380     #endif
381   #endif
382   #ifndef NK_POINTER_TYPE
383     #if defined(_WIN64) && defined(_MSC_VER)
384       #define NK_POINTER_TYPE unsigned __int64
385     #elif (defined(_WIN32) || defined(WIN32)) && defined(_MSC_VER)
386       #define NK_POINTER_TYPE unsigned __int32
387     #elif defined(__GNUC__) || defined(__clang__)
388       #if defined(__x86_64__) || defined(__ppc64__)
389         #define NK_POINTER_TYPE unsigned long
390       #else
391         #define NK_POINTER_TYPE unsigned int
392       #endif
393     #else
394       #define NK_POINTER_TYPE unsigned long
395     #endif
396   #endif
397 #endif
398
399 typedef NK_INT8 nk_char;
400 typedef NK_UINT8 nk_uchar;
401 typedef NK_UINT8 nk_byte;
402 typedef NK_INT16 nk_short;
403 typedef NK_UINT16 nk_ushort;
404 typedef NK_INT32 nk_int;
405 typedef NK_UINT32 nk_uint;
406 typedef NK_SIZE_TYPE nk_size;
407 typedef NK_POINTER_TYPE nk_ptr;
408
409 typedef nk_uint nk_hash;
410 typedef nk_uint nk_flags;
411 typedef nk_uint nk_rune;
412
413 /* Make sure correct type size:
414  * This will fire with a negative subscript error if the type sizes
415  * are set incorrectly by the compiler, and compile out if not */
416 NK_STATIC_ASSERT(sizeof(nk_short) == 2);
417 NK_STATIC_ASSERT(sizeof(nk_ushort) == 2);
418 NK_STATIC_ASSERT(sizeof(nk_uint) == 4);
419 NK_STATIC_ASSERT(sizeof(nk_int) == 4);
420 NK_STATIC_ASSERT(sizeof(nk_byte) == 1);
421 NK_STATIC_ASSERT(sizeof(nk_flags) >= 4);
422 NK_STATIC_ASSERT(sizeof(nk_rune) >= 4);
423 NK_STATIC_ASSERT(sizeof(nk_size) >= sizeof(void*));
424 NK_STATIC_ASSERT(sizeof(nk_ptr) >= sizeof(void*));
425
426 /* ==============================================================================
427  *
428  *                                  API
429  *
430  * ============================================================================== */
431 struct nk_buffer;
432 struct nk_allocator;
433 struct nk_command_buffer;
434 struct nk_draw_command;
435 struct nk_convert_config;
436 struct nk_style_item;
437 struct nk_text_edit;
438 struct nk_draw_list;
439 struct nk_user_font;
440 struct nk_panel;
441 struct nk_context;
442 struct nk_draw_vertex_layout_element;
443 struct nk_style_button;
444 struct nk_style_toggle;
445 struct nk_style_selectable;
```

```
446 struct nk_style_slide;
447 struct nk_style_progress;
448 struct nk_style_scrollbar;
449 struct nk_style_edit;
450 struct nk_style_property;
451 struct nk_style_chart;
452 struct nk_style_combo;
453 struct nk_style_tab;
454 struct nk_style_window_header;
455 struct nk_style_window;
456
457 enum {nk_false, nk_true};
458 struct nk_color {nk_byte r,g,b,a;};
459 struct nk_colorf {float r,g,b,a;};
460 struct nk_vec2 {float x,y;};
461 struct nk_vec2i {short x, y;};
462 struct nk_rect {float x,y,w,h;};
463 struct nk_recti {short x,y,w,h;};
464 typedef char nk_glyph[NK_UTF_SIZE];
465 typedef union {void *ptr; int id;} nk_handle;
466 struct nk_image {nk_handle handle;unsigned short w,h;unsigned short region[4];};
467 struct nk_cursor {struct nk_image img; struct nk_vec2 size, offset;};
468 struct nk_scroll {nk_uint x, y;};
469
470 enum nk_heading         {NK_UP, NK_RIGHT, NK_DOWN, NK_LEFT};
471 enum nk_button_behavior {NK_BUTTON_DEFAULT, NK_BUTTON_REPEATER};
472 enum nk_modify          {NK_FIXED = nk_false, NK_MODIFIABLE = nk_true};
473 enum nk_orientation     {NK_VERTICAL, NK_HORIZONTAL};
474 enum nk_collapse_states {NK_MINIMIZED = nk_false, NK_MAXIMIZED = nk_true};
475 enum nk_show_states     {NK_HIDDEN = nk_false, NK_SHOWN = nk_true};
476 enum nk_chart_type      {NK_CHART_LINES, NK_CHART_COLUMN, NK_CHART_MAX};
477 enum nk_chart_event     {NK_CHART_HOVERING = 0x01, NK_CHART_CLICKED = 0x02};
478 enum nk_color_format    {NK_RGB, NK_RGBA};
479 enum nk_popup_type      {NK_POPUP_STATIC, NK_POPUP_DYNAMIC};
480 enum nk_layout_format   {NK_DYNAMIC, NK_STATIC};
481 enum nk_tree_type       {NK_TREE_NODE, NK_TREE_TAB};
482
483 typedef void*(*nk_plugin_alloc)(nk_handle, void *old, nk_size);
484 typedef void (*nk_plugin_free)(nk_handle, void *old);
485 typedef int(*nk_plugin_filter)(const struct nk_text_edit*, nk_rune unicode);
486 typedef void(*nk_plugin_paste)(nk_handle, struct nk_text_edit*);
487 typedef void(*nk_plugin_copy)(nk_handle, const char*, int len);
488
489 struct nk_allocator {
490     nk_handle userdata;
491     nk_plugin_alloc alloc;
492     nk_plugin_free free;
493 };
494 enum nk_symbol_type {
495     NK_SYMBOL_NONE,
496     NK_SYMBOL_X,
497     NK_SYMBOL_UNDERSCORE,
498     NK_SYMBOL_CIRCLE_SOLID,
499     NK_SYMBOL_CIRCLE_OUTLINE,
500     NK_SYMBOL_RECT_SOLID,
501     NK_SYMBOL_RECT_OUTLINE,
502     NK_SYMBOL_TRIANGLE_UP,
503     NK_SYMBOL_TRIANGLE_DOWN,
504     NK_SYMBOL_TRIANGLE_LEFT,
505     NK_SYMBOL_TRIANGLE_RIGHT,
506     NK_SYMBOL_PLUS,
507     NK_SYMBOL_MINUS,
508     NK_SYMBOL_MAX
509 };
510 /* =============================================================================
511  *
512  *                                  CONTEXT
513  *
514  * =============================================================================*/
515 /*/// ### Context
547  */
548 #ifdef NK_INCLUDE_DEFAULT_ALLOCATOR
549 /*/// #### nk_init_default
564 */
565 NK_API int nk_init_default(struct nk_context*, const struct nk_user_font*);
566 #endif
567 /*/// #### nk_init_fixed
589 */
590 NK_API int nk_init_fixed(struct nk_context*, void *memory, nk_size size, const struct nk_user_font*);
591 /*/// #### nk_init
607 */
608 NK_API int nk_init(struct nk_context*, struct nk_allocator*, const struct nk_user_font*);
609 /*/// #### nk_init_custom
626 */
627 NK_API int nk_init_custom(struct nk_context*, struct nk_buffer *cmds, struct nk_buffer *pool, const
       struct nk_user_font*);
628 /*/// #### nk_clear
```

```
640 */
641 NK_API void nk_clear(struct nk_context*);
642 /*/// #### nk_free
653 */
654 NK_API void nk_free(struct nk_context*);
655 #ifdef NK_INCLUDE_COMMAND_USERDATA
656 /*/// #### nk_set_user_data
667 */
668 NK_API void nk_set_user_data(struct nk_context*, nk_handle handle);
669 #endif
670 /* ===============================================================================
671  *
672  *                                  INPUT
673  *
674  * ===============================================================================*/
675 /*/// ### Input
735 */
736 enum nk_keys {
737     NK_KEY_NONE,
738     NK_KEY_SHIFT,
739     NK_KEY_CTRL,
740     NK_KEY_DEL,
741     NK_KEY_ENTER,
742     NK_KEY_TAB,
743     NK_KEY_BACKSPACE,
744     NK_KEY_COPY,
745     NK_KEY_CUT,
746     NK_KEY_PASTE,
747     NK_KEY_UP,
748     NK_KEY_DOWN,
749     NK_KEY_LEFT,
750     NK_KEY_RIGHT,
751     /* Shortcuts: text field */
752     NK_KEY_TEXT_INSERT_MODE,
753     NK_KEY_TEXT_REPLACE_MODE,
754     NK_KEY_TEXT_RESET_MODE,
755     NK_KEY_TEXT_LINE_START,
756     NK_KEY_TEXT_LINE_END,
757     NK_KEY_TEXT_START,
758     NK_KEY_TEXT_END,
759     NK_KEY_TEXT_UNDO,
760     NK_KEY_TEXT_REDO,
761     NK_KEY_TEXT_SELECT_ALL,
762     NK_KEY_TEXT_WORD_LEFT,
763     NK_KEY_TEXT_WORD_RIGHT,
764     /* Shortcuts: scrollbar */
765     NK_KEY_SCROLL_START,
766     NK_KEY_SCROLL_END,
767     NK_KEY_SCROLL_DOWN,
768     NK_KEY_SCROLL_UP,
769     NK_KEY_MAX
770 };
771 enum nk_buttons {
772     NK_BUTTON_LEFT,
773     NK_BUTTON_MIDDLE,
774     NK_BUTTON_RIGHT,
775     NK_BUTTON_DOUBLE,
776     NK_BUTTON_MAX
777 };
778 /*/// #### nk_input_begin
789 */
790 NK_API void nk_input_begin(struct nk_context*);
791 /*/// #### nk_input_motion
803 */
804 NK_API void nk_input_motion(struct nk_context*, int x, int y);
805 /*/// #### nk_input_key
817 */
818 NK_API void nk_input_key(struct nk_context*, enum nk_keys, int down);
819 /*/// #### nk_input_button
833 */
834 NK_API void nk_input_button(struct nk_context*, enum nk_buttons, int x, int y, int down);
835 /*/// #### nk_input_scroll
848 */
849 NK_API void nk_input_scroll(struct nk_context*, struct nk_vec2 val);
850 /*/// #### nk_input_char
866 */
867 NK_API void nk_input_char(struct nk_context*, char);
868 /*/// #### nk_input_glyph
883 */
884 NK_API void nk_input_glyph(struct nk_context*, const nk_glyph);
885 /*/// #### nk_input_unicode
899 */
900 NK_API void nk_input_unicode(struct nk_context*, nk_rune);
901 /*/// #### nk_input_end
912 */
913 NK_API void nk_input_end(struct nk_context*);
914 /* ===============================================================================
```

```
915  *
916  *                                  DRAWING
917  *
918  * ===============================================================================*/
919 /*/// ### Drawing
1142 */
1143 enum nk_anti_aliasing {NK_ANTI_ALIASING_OFF, NK_ANTI_ALIASING_ON};
1144 enum nk_convert_result {
1145     NK_CONVERT_SUCCESS = 0,
1146     NK_CONVERT_INVALID_PARAM = 1,
1147     NK_CONVERT_COMMAND_BUFFER_FULL = NK_FLAG(1),
1148     NK_CONVERT_VERTEX_BUFFER_FULL = NK_FLAG(2),
1149     NK_CONVERT_ELEMENT_BUFFER_FULL = NK_FLAG(3)
1150 };
1151 struct nk_draw_null_texture {
1152     nk_handle texture; /* texture handle to a texture with a white pixel */
1153     struct nk_vec2 uv; /* coordinates to a white pixel in the texture  */
1154 };
1155 struct nk_convert_config {
1156     float global_alpha; /* global alpha value */
1157     enum nk_anti_aliasing line_AA; /* line anti-aliasing flag can be turned off if you are tight on
      memory */
1158     enum nk_anti_aliasing shape_AA; /* shape anti-aliasing flag can be turned off if you are tight on
      memory */
1159     unsigned circle_segment_count; /* number of segments used for circles: default to 22 */
1160     unsigned arc_segment_count; /* number of segments used for arcs: default to 22 */
1161     unsigned curve_segment_count; /* number of segments used for curves: default to 22 */
1162     struct nk_draw_null_texture null; /* handle to texture with a white pixel for shape drawing */
1163     const struct nk_draw_vertex_layout_element *vertex_layout; /* describes the vertex output format
      and packing */
1164     nk_size vertex_size; /* sizeof one vertex for vertex packing */
1165     nk_size vertex_alignment; /* vertex alignment: Can be obtained by NK_ALIGNOF */
1166 };
1167 /*/// #### nk__begin
1180 */
1181 NK_API const struct nk_command* nk__begin(struct nk_context*);
1182 /*/// #### nk__next
1195 */
1196 NK_API const struct nk_command* nk__next(struct nk_context*, const struct nk_command*);
1197 /*/// #### nk_foreach
1210 */
1211 #define nk_foreach(c, ctx) for((c) = nk__begin(ctx); (c) != 0; (c) = nk__next(ctx,c))
1212 #ifdef NK_INCLUDE_VERTEX_BUFFER_OUTPUT
1213 /*/// #### nk_convert
1241 */
1242 NK_API nk_flags nk_convert(struct nk_context*, struct nk_buffer *cmds, struct nk_buffer *vertices,
      struct nk_buffer *elements, const struct nk_convert_config*);
1243 /*/// #### nk__draw_begin
1256 */
1257 NK_API const struct nk_draw_command* nk__draw_begin(const struct nk_context*, const struct nk_buffer*);
1258 /*/// #### nk__draw_end
1271 */
1272 NK_API const struct nk_draw_command* nk__draw_end(const struct nk_context*, const struct nk_buffer*);
1273 /*/// #### nk__draw_next
1287 */
1288 NK_API const struct nk_draw_command* nk__draw_next(const struct nk_draw_command*, const struct
      nk_buffer*, const struct nk_context*);
1289 /*/// #### nk_draw_foreach
1301 */
1302 #define nk_draw_foreach(cmd,ctx, b) for((cmd)=nk__draw_begin(ctx, b); (cmd)!=0;
      (cmd)=nk__draw_next(cmd, b, ctx))
1303 #endif
1304 /* ===============================================================================
1305  *
1306  *                                  WINDOW
1307  *
1308  * ===============================================================================
1393 //
1415 //
1421 //
1427 */
1428 /*
1450 */
1451 enum nk_panel_flags {
1452     NK_WINDOW_BORDER            = NK_FLAG(0),
1453     NK_WINDOW_MOVABLE           = NK_FLAG(1),
1454     NK_WINDOW_SCALABLE          = NK_FLAG(2),
1455     NK_WINDOW_CLOSABLE          = NK_FLAG(3),
1456     NK_WINDOW_MINIMIZABLE       = NK_FLAG(4),
1457     NK_WINDOW_NO_SCROLLBAR      = NK_FLAG(5),
1458     NK_WINDOW_TITLE             = NK_FLAG(6),
1459     NK_WINDOW_SCROLL_AUTO_HIDE  = NK_FLAG(7),
1460     NK_WINDOW_BACKGROUND        = NK_FLAG(8),
1461     NK_WINDOW_SCALE_LEFT        = NK_FLAG(9),
1462     NK_WINDOW_NO_INPUT          = NK_FLAG(10)
1463 };
1464 /*/// #### nk_begin
```

```
1481 */
1482 NK_API int nk_begin(struct nk_context *ctx, const char *title, struct nk_rect bounds, nk_flags flags);
1483 /*/// #### nk_begin_titled
1501 */
1502 NK_API int nk_begin_titled(struct nk_context *ctx, const char *name, const char *title, struct nk_rect
       bounds, nk_flags flags);
1503 /*/// #### nk_end
1514 */
1515 NK_API void nk_end(struct nk_context *ctx);
1516 /*/// #### nk_window_find
1530 */
1531 NK_API struct nk_window *nk_window_find(struct nk_context *ctx, const char *name);
1532 /*/// #### nk_window_get_bounds
1546 */
1547 NK_API struct nk_rect nk_window_get_bounds(const struct nk_context *ctx);
1548 /*/// #### nk_window_get_position
1562 */
1563 NK_API struct nk_vec2 nk_window_get_position(const struct nk_context *ctx);
1564 /*/// #### nk_window_get_size
1578 */
1579 NK_API struct nk_vec2 nk_window_get_size(const struct nk_context*);
1580 /*/// #### nk_window_get_width
1594 */
1595 NK_API float nk_window_get_width(const struct nk_context*);
1596 /*/// #### nk_window_get_height
1610 */
1611 NK_API float nk_window_get_height(const struct nk_context*);
1612 /*/// #### nk_window_get_panel
1628 */
1629 NK_API struct nk_panel* nk_window_get_panel(struct nk_context*);
1630 /*/// #### nk_window_get_content_region
1647 */
1648 NK_API struct nk_rect nk_window_get_content_region(struct nk_context*);
1649 /*/// #### nk_window_get_content_region_min
1666 */
1667 NK_API struct nk_vec2 nk_window_get_content_region_min(struct nk_context*);
1668 /*/// #### nk_window_get_content_region_max
1685 */
1686 NK_API struct nk_vec2 nk_window_get_content_region_max(struct nk_context*);
1687 /*/// #### nk_window_get_content_region_size
1703 */
1704 NK_API struct nk_vec2 nk_window_get_content_region_size(struct nk_context*);
1705 /*/// #### nk_window_get_canvas
1722 */
1723 NK_API struct nk_command_buffer* nk_window_get_canvas(struct nk_context*);
1724 /*/// #### nk_window_get_scroll
1738 */
1739 NK_API void nk_window_get_scroll(struct nk_context*, nk_uint *offset_x, nk_uint *offset_y);
1740 /*/// #### nk_window_has_focus
1753 */
1754 NK_API int nk_window_has_focus(const struct nk_context*);
1755 /*/// #### nk_window_is_hovered
1768 */
1769 NK_API int nk_window_is_hovered(struct nk_context*);
1770 /*/// #### nk_window_is_collapsed
1783 */
1784 NK_API int nk_window_is_collapsed(struct nk_context *ctx, const char *name);
1785 /*/// #### nk_window_is_closed
1797 */
1798 NK_API int nk_window_is_closed(struct nk_context*, const char*);
1799 /*/// #### nk_window_is_hidden
1811 */
1812 NK_API int nk_window_is_hidden(struct nk_context*, const char*);
1813 /*/// #### nk_window_is_active
1825 */
1826 NK_API int nk_window_is_active(struct nk_context*, const char*);
1827 /*/// #### nk_window_is_any_hovered
1838 */
1839 NK_API int nk_window_is_any_hovered(struct nk_context*);
1840 /*/// #### nk_item_is_any_active
1853 */
1854 NK_API int nk_item_is_any_active(struct nk_context*);
1855 /*/// #### nk_window_set_bounds
1866 */
1867 NK_API void nk_window_set_bounds(struct nk_context*, const char *name, struct nk_rect bounds);
1868 /*/// #### nk_window_set_position
1879 */
1880 NK_API void nk_window_set_position(struct nk_context*, const char *name, struct nk_vec2 pos);
1881 /*/// #### nk_window_set_size
1892 */
1893 NK_API void nk_window_set_size(struct nk_context*, const char *name, struct nk_vec2);
1894 /*/// #### nk_window_set_focus
1904 */
1905 NK_API void nk_window_set_focus(struct nk_context*, const char *name);
1906 /*/// #### nk_window_set_scroll
1920 */
1921 NK_API void nk_window_set_scroll(struct nk_context*, nk_uint offset_x, nk_uint offset_y);
```

```
1922 /*/// #### nk_window_close
1932 */
1933 NK_API void nk_window_close(struct nk_context *ctx, const char *name);
1934 /*/// #### nk_window_collapse
1945 */
1946 NK_API void nk_window_collapse(struct nk_context*, const char *name, enum nk_collapse_states state);
1947 /*/// #### nk_window_collapse_if
1959 */
1960 NK_API void nk_window_collapse_if(struct nk_context*, const char *name, enum nk_collapse_states, int
       cond);
1961 /*/// #### nk_window_show
1972 */
1973 NK_API void nk_window_show(struct nk_context*, const char *name, enum nk_show_states);
1974 /*/// #### nk_window_show_if
1986 */
1987 NK_API void nk_window_show_if(struct nk_context*, const char *name, enum nk_show_states, int cond);
1988 /* =============================================================================
1989  *
1990  *                                  LAYOUT
1991  *
1992  * =============================================================================
2237 //
2244 //
2250 //
2254 //
2260 */
2261 /*/// #### nk_layout_set_min_row_height
2275 */
2276 NK_API void nk_layout_set_min_row_height(struct nk_context*, float height);
2277 /*/// #### nk_layout_reset_min_row_height
2286 */
2287 NK_API void nk_layout_reset_min_row_height(struct nk_context*);
2288 /*/// #### nk_layout_widget_bounds
2299 */
2300 NK_API struct nk_rect nk_layout_widget_bounds(struct nk_context*);
2301 /*/// #### nk_layout_ratio_from_pixel
2313 */
2314 NK_API float nk_layout_ratio_from_pixel(struct nk_context*, float pixel_width);
2315 /*/// #### nk_layout_row_dynamic
2328 */
2329 NK_API void nk_layout_row_dynamic(struct nk_context *ctx, float height, int cols);
2330 /*/// #### nk_layout_row_static
2344 */
2345 NK_API void nk_layout_row_static(struct nk_context *ctx, float height, int item_width, int cols);
2346 /*/// #### nk_layout_row_begin
2358 */
2359 NK_API void nk_layout_row_begin(struct nk_context *ctx, enum nk_layout_format fmt, float row_height,
       int cols);
2360 /*/// #### nk_layout_row_push
2370 */
2371 NK_API void nk_layout_row_push(struct nk_context*, float value);
2372 /*/// #### nk_layout_row_end
2381 */
2382 NK_API void nk_layout_row_end(struct nk_context*);
2383 /*/// #### nk_layout_row
2395 */
2396 NK_API void nk_layout_row(struct nk_context*, enum nk_layout_format, float height, int cols, const
       float *ratio);
2397 /*/// #### nk_layout_row_template_begin
2407 */
2408 NK_API void nk_layout_row_template_begin(struct nk_context*, float row_height);
2409 /*/// #### nk_layout_row_template_push_dynamic
2419 */
2420 NK_API void nk_layout_row_template_push_dynamic(struct nk_context*);
2421 /*/// #### nk_layout_row_template_push_variable
2431 */
2432 NK_API void nk_layout_row_template_push_variable(struct nk_context*, float min_width);
2433 /*/// #### nk_layout_row_template_push_static
2443 */
2444 NK_API void nk_layout_row_template_push_static(struct nk_context*, float width);
2445 /*/// #### nk_layout_row_template_end
2454 */
2455 NK_API void nk_layout_row_template_end(struct nk_context*);
2456 /*/// #### nk_layout_space_begin
2468 */
2469 NK_API void nk_layout_space_begin(struct nk_context*, enum nk_layout_format, float height, int
       widget_count);
2470 /*/// #### nk_layout_space_push
2480 */
2481 NK_API void nk_layout_space_push(struct nk_context*, struct nk_rect bounds);
2482 /*/// #### nk_layout_space_end
2491 */
2492 NK_API void nk_layout_space_end(struct nk_context*);
2493 /*/// #### nk_layout_space_bounds
2504 */
2505 NK_API struct nk_rect nk_layout_space_bounds(struct nk_context*);
2506 /*/// #### nk_layout_space_to_screen
```

```
2518 */
2519 NK_API struct nk_vec2 nk_layout_space_to_screen(struct nk_context*, struct nk_vec2);
2520 /*/// #### nk_layout_space_to_local
2532 */
2533 NK_API struct nk_vec2 nk_layout_space_to_local(struct nk_context*, struct nk_vec2);
2534 /*/// #### nk_layout_space_rect_to_screen
2546 */
2547 NK_API struct nk_rect nk_layout_space_rect_to_screen(struct nk_context*, struct nk_rect);
2548 /*/// #### nk_layout_space_rect_to_local
2560 */
2561 NK_API struct nk_rect nk_layout_space_rect_to_local(struct nk_context*, struct nk_rect);
2562 /* ==============================================================================
2563  *
2564  *                                  GROUP
2565  *
2566  * ==============================================================================
2649 */
2650 /*/// #### nk_group_begin
2663 */
2664 NK_API int nk_group_begin(struct nk_context*, const char *title, nk_flags);
2665 /*/// #### nk_group_begin_titled
2679 */
2680 NK_API int nk_group_begin_titled(struct nk_context*, const char *name, const char *title, nk_flags);
2681 /*/// #### nk_group_end
2690 */
2691 NK_API void nk_group_end(struct nk_context*);
2692 /*/// #### nk_group_scrolled_offset_begin
2708 */
2709 NK_API int nk_group_scrolled_offset_begin(struct nk_context*, nk_uint *x_offset, nk_uint *y_offset,
      const char *title, nk_flags flags);
2710 /*/// #### nk_group_scrolled_begin
2725 */
2726 NK_API int nk_group_scrolled_begin(struct nk_context*, struct nk_scroll *off, const char *title,
      nk_flags);
2727 /*/// #### nk_group_scrolled_end
2736 */
2737 NK_API void nk_group_scrolled_end(struct nk_context*);
2738 /*/// #### nk_group_get_scroll
2750 */
2751 NK_API void nk_group_get_scroll(struct nk_context*, const char *id, nk_uint *x_offset, nk_uint
      *y_offset);
2752 /*/// #### nk_group_set_scroll
2764 */
2765 NK_API void nk_group_set_scroll(struct nk_context*, const char *id, nk_uint x_offset, nk_uint
      y_offset);
2766 /* ==============================================================================
2767  *
2768  *                                  TREE
2769  *
2770  * ==============================================================================
2825 //
2835 */
2836 /*/// #### nk_tree_push
2856 */
2857 #define nk_tree_push(ctx, type, title, state) nk_tree_push_hashed(ctx, type, title, state,
      NK_FILE_LINE,nk_strlen(NK_FILE_LINE),__LINE__)
2858 /*/// #### nk_tree_push_id
2873 */
2874 #define nk_tree_push_id(ctx, type, title, state, id) nk_tree_push_hashed(ctx, type, title, state,
      NK_FILE_LINE,nk_strlen(NK_FILE_LINE),id)
2875 /*/// #### nk_tree_push_hashed
2893 */
2894 NK_API int nk_tree_push_hashed(struct nk_context*, enum nk_tree_type, const char *title, enum
      nk_collapse_states initial_state, const char *hash, int len,int seed);
2895 /*/// #### nk_tree_image_push
2906 //
2916 */
2917 #define nk_tree_image_push(ctx, type, img, title, state) nk_tree_image_push_hashed(ctx, type, img,
      title, state, NK_FILE_LINE,nk_strlen(NK_FILE_LINE),__LINE__)
2918 /*/// #### nk_tree_image_push_id
2936 */
2937 #define nk_tree_image_push_id(ctx, type, img, title, state, id) nk_tree_image_push_hashed(ctx, type,
      img, title, state, NK_FILE_LINE,nk_strlen(NK_FILE_LINE),id)
2938 /*/// #### nk_tree_image_push_hashed
2957 */
2958 NK_API int nk_tree_image_push_hashed(struct nk_context*, enum nk_tree_type, struct nk_image, const char
      *title, enum nk_collapse_states initial_state, const char *hash, int len,int seed);
2959 /*/// #### nk_tree_pop
2968 */
2969 NK_API void nk_tree_pop(struct nk_context*);
2970 /*/// #### nk_tree_state_push
2984 */
2985 NK_API int nk_tree_state_push(struct nk_context*, enum nk_tree_type, const char *title, enum
      nk_collapse_states *state);
2986 /*/// #### nk_tree_state_image_push
3001 */
3002 NK_API int nk_tree_state_image_push(struct nk_context*, enum nk_tree_type, struct nk_image, const char
```

```
    *title, enum nk_collapse_states *state);
/*/// #### nk_tree_state_pop
*/
NK_API void nk_tree_state_pop(struct nk_context*);

#define nk_tree_element_push(ctx, type, title, state, sel) nk_tree_element_push_hashed(ctx, type,
    title, state, sel, NK_FILE_LINE,nk_strlen(NK_FILE_LINE),__LINE__)
#define nk_tree_element_push_id(ctx, type, title, state, sel, id) nk_tree_element_push_hashed(ctx,
    type, title, state, sel, NK_FILE_LINE,nk_strlen(NK_FILE_LINE),id)
NK_API int nk_tree_element_push_hashed(struct nk_context*, enum nk_tree_type, const char *title, enum
    nk_collapse_states initial_state, int *selected, const char *hash, int len, int seed);
NK_API int nk_tree_element_image_push_hashed(struct nk_context*, enum nk_tree_type, struct nk_image,
    const char *title, enum nk_collapse_states initial_state, int *selected, const char *hash, int
    len,int seed);
NK_API void nk_tree_element_pop(struct nk_context*);

/* ==============================================================================
 *
 *                                  LIST VIEW
 *
 * ============================================================================== */
struct nk_list_view {
/* public: */
    int begin, end, count;
/* private: */
    int total_height;
    struct nk_context *ctx;
    nk_uint *scroll_pointer;
    nk_uint scroll_value;
};
NK_API int nk_list_view_begin(struct nk_context*, struct nk_list_view *out, const char *id, nk_flags,
    int row_height, int row_count);
NK_API void nk_list_view_end(struct nk_list_view*);
/* ==============================================================================
 *
 *                                  WIDGET
 *
 * ============================================================================== */
enum nk_widget_layout_states {
    NK_WIDGET_INVALID, /* The widget cannot be seen and is completely out of view */
    NK_WIDGET_VALID, /* The widget is completely inside the window and can be updated and drawn */
    NK_WIDGET_ROM /* The widget is partially visible and cannot be updated */
};
enum nk_widget_states {
    NK_WIDGET_STATE_MODIFIED    = NK_FLAG(1),
    NK_WIDGET_STATE_INACTIVE    = NK_FLAG(2), /* widget is neither active nor hovered */
    NK_WIDGET_STATE_ENTERED     = NK_FLAG(3), /* widget has been hovered on the current frame */
    NK_WIDGET_STATE_HOVER       = NK_FLAG(4), /* widget is being hovered */
    NK_WIDGET_STATE_ACTIVED     = NK_FLAG(5),/* widget is currently activated */
    NK_WIDGET_STATE_LEFT        = NK_FLAG(6), /* widget is from this frame on not hovered anymore */
    NK_WIDGET_STATE_HOVERED     = NK_WIDGET_STATE_HOVER|NK_WIDGET_STATE_MODIFIED, /* widget is being
    hovered */
    NK_WIDGET_STATE_ACTIVE      = NK_WIDGET_STATE_ACTIVED|NK_WIDGET_STATE_MODIFIED /* widget is
    currently activated */
};
NK_API enum nk_widget_layout_states nk_widget(struct nk_rect*, const struct nk_context*);
NK_API enum nk_widget_layout_states nk_widget_fitting(struct nk_rect*, struct nk_context*, struct
    nk_vec2);
NK_API struct nk_rect nk_widget_bounds(struct nk_context*);
NK_API struct nk_vec2 nk_widget_position(struct nk_context*);
NK_API struct nk_vec2 nk_widget_size(struct nk_context*);
NK_API float nk_widget_width(struct nk_context*);
NK_API float nk_widget_height(struct nk_context*);
NK_API int nk_widget_is_hovered(struct nk_context*);
NK_API int nk_widget_is_mouse_clicked(struct nk_context*, enum nk_buttons);
NK_API int nk_widget_has_mouse_click_down(struct nk_context*, enum nk_buttons, int down);
NK_API void nk_spacing(struct nk_context*, int cols);
/* ==============================================================================
 *
 *                                  TEXT
 *
 * ============================================================================== */
enum nk_text_align {
    NK_TEXT_ALIGN_LEFT        = 0x01,
    NK_TEXT_ALIGN_CENTERED    = 0x02,
    NK_TEXT_ALIGN_RIGHT       = 0x04,
    NK_TEXT_ALIGN_TOP         = 0x08,
    NK_TEXT_ALIGN_MIDDLE      = 0x10,
    NK_TEXT_ALIGN_BOTTOM      = 0x20
};
enum nk_text_alignment {
    NK_TEXT_LEFT        = NK_TEXT_ALIGN_MIDDLE|NK_TEXT_ALIGN_LEFT,
    NK_TEXT_CENTERED    = NK_TEXT_ALIGN_MIDDLE|NK_TEXT_ALIGN_CENTERED,
    NK_TEXT_RIGHT       = NK_TEXT_ALIGN_MIDDLE|NK_TEXT_ALIGN_RIGHT
};
NK_API void nk_text(struct nk_context*, const char*, int, nk_flags);
NK_API void nk_text_colored(struct nk_context*, const char*, int, nk_flags, struct nk_color);
```

```
3088 NK_API void nk_text_wrap(struct nk_context*, const char*, int);
3089 NK_API void nk_text_wrap_colored(struct nk_context*, const char*, int, struct nk_color);
3090 NK_API void nk_label(struct nk_context*, const char*, nk_flags align);
3091 NK_API void nk_label_colored(struct nk_context*, const char*, nk_flags align, struct nk_color);
3092 NK_API void nk_label_wrap(struct nk_context*, const char*);
3093 NK_API void nk_label_colored_wrap(struct nk_context*, const char*, struct nk_color);
3094 NK_API void nk_image(struct nk_context*, struct nk_image);
3095 NK_API void nk_image_color(struct nk_context*, struct nk_image, struct nk_color);
3096 #ifdef NK_INCLUDE_STANDARD_VARARGS
3097 NK_API void nk_labelf(struct nk_context*, nk_flags, NK_PRINTF_FORMAT_STRING const char*, ...)
       NK_PRINTF_VARARG_FUNC(3);
3098 NK_API void nk_labelf_colored(struct nk_context*, nk_flags, struct nk_color, NK_PRINTF_FORMAT_STRING
       const char*,...) NK_PRINTF_VARARG_FUNC(4);
3099 NK_API void nk_labelf_wrap(struct nk_context*, NK_PRINTF_FORMAT_STRING const char*,...)
       NK_PRINTF_VARARG_FUNC(2);
3100 NK_API void nk_labelf_colored_wrap(struct nk_context*, struct nk_color, NK_PRINTF_FORMAT_STRING const
       char*,...) NK_PRINTF_VARARG_FUNC(3);
3101 NK_API void nk_labelfv(struct nk_context*, nk_flags, NK_PRINTF_FORMAT_STRING const char*, va_list)
       NK_PRINTF_VALIST_FUNC(3);
3102 NK_API void nk_labelfv_colored(struct nk_context*, nk_flags, struct nk_color, NK_PRINTF_FORMAT_STRING
       const char*, va_list) NK_PRINTF_VALIST_FUNC(4);
3103 NK_API void nk_labelfv_wrap(struct nk_context*, NK_PRINTF_FORMAT_STRING const char*, va_list)
       NK_PRINTF_VALIST_FUNC(2);
3104 NK_API void nk_labelfv_colored_wrap(struct nk_context*, struct nk_color, NK_PRINTF_FORMAT_STRING const
       char*, va_list) NK_PRINTF_VALIST_FUNC(3);
3105 NK_API void nk_value_bool(struct nk_context*, const char *prefix, int);
3106 NK_API void nk_value_int(struct nk_context*, const char *prefix, int);
3107 NK_API void nk_value_uint(struct nk_context*, const char *prefix, unsigned int);
3108 NK_API void nk_value_float(struct nk_context*, const char *prefix, float);
3109 NK_API void nk_value_color_byte(struct nk_context*, const char *prefix, struct nk_color);
3110 NK_API void nk_value_color_float(struct nk_context*, const char *prefix, struct nk_color);
3111 NK_API void nk_value_color_hex(struct nk_context*, const char *prefix, struct nk_color);
3112 #endif
3113 /* =============================================================================
3114  *
3115  *                                  BUTTON
3116  *
3117  * ============================================================================= */
3118 NK_API int nk_button_text(struct nk_context*, const char *title, int len);
3119 NK_API int nk_button_label(struct nk_context*, const char *title);
3120 NK_API int nk_button_color(struct nk_context*, struct nk_color);
3121 NK_API int nk_button_symbol(struct nk_context*, enum nk_symbol_type);
3122 NK_API int nk_button_image(struct nk_context*, struct nk_image img);
3123 NK_API int nk_button_symbol_label(struct nk_context*, enum nk_symbol_type, const char*, nk_flags
       text_alignment);
3124 NK_API int nk_button_symbol_text(struct nk_context*, enum nk_symbol_type, const char*, int, nk_flags
       alignment);
3125 NK_API int nk_button_image_label(struct nk_context*, struct nk_image img, const char*, nk_flags
       text_alignment);
3126 NK_API int nk_button_image_text(struct nk_context*, struct nk_image img, const char*, int, nk_flags
       alignment);
3127 NK_API int nk_button_text_styled(struct nk_context*, const struct nk_style_button*, const char *title,
       int len);
3128 NK_API int nk_button_label_styled(struct nk_context*, const struct nk_style_button*, const char
       *title);
3129 NK_API int nk_button_symbol_styled(struct nk_context*, const struct nk_style_button*, enum
       nk_symbol_type);
3130 NK_API int nk_button_image_styled(struct nk_context*, const struct nk_style_button*, struct nk_image
       img);
3131 NK_API int nk_button_symbol_text_styled(struct nk_context*,const struct nk_style_button*, enum
       nk_symbol_type, const char*, int, nk_flags alignment);
3132 NK_API int nk_button_symbol_label_styled(struct nk_context *ctx, const struct nk_style_button *style,
       enum nk_symbol_type symbol, const char *title, nk_flags align);
3133 NK_API int nk_button_image_label_styled(struct nk_context*,const struct nk_style_button*, struct
       nk_image img, const char*, nk_flags text_alignment);
3134 NK_API int nk_button_image_text_styled(struct nk_context*,const struct nk_style_button*, struct
       nk_image img, const char*, int, nk_flags alignment);
3135 NK_API void nk_button_set_behavior(struct nk_context*, enum nk_button_behavior);
3136 NK_API int nk_button_push_behavior(struct nk_context*, enum nk_button_behavior);
3137 NK_API int nk_button_pop_behavior(struct nk_context*);
3138 /* =============================================================================
3139  *
3140  *                                  CHECKBOX
3141  *
3142  * ============================================================================= */
3143 NK_API int nk_check_label(struct nk_context*, const char*, int active);
3144 NK_API int nk_check_text(struct nk_context*, const char*, int,int active);
3145 NK_API unsigned nk_check_flags_label(struct nk_context*, const char*, unsigned int flags, unsigned int
       value);
3146 NK_API unsigned nk_check_flags_text(struct nk_context*, const char*, int, unsigned int flags, unsigned
       int value);
3147 NK_API int nk_checkbox_label(struct nk_context*, const char*, int *active);
3148 NK_API int nk_checkbox_text(struct nk_context*, const char*, int, int *active);
3149 NK_API int nk_checkbox_flags_label(struct nk_context*, const char*, unsigned int *flags, unsigned int
       value);
3150 NK_API int nk_checkbox_flags_text(struct nk_context*, const char*, int, unsigned int *flags, unsigned
       int value);
```

```
3151 /* ===============================================================================
3152  *
3153  *                                  RADIO BUTTON
3154  *
3155  * =============================================================================== */
3156 NK_API int nk_radio_label(struct nk_context*, const char*, int *active);
3157 NK_API int nk_radio_text(struct nk_context*, const char*, int, int *active);
3158 NK_API int nk_option_label(struct nk_context*, const char*, int active);
3159 NK_API int nk_option_text(struct nk_context*, const char*, int, int active);
3160 /* ===============================================================================
3161  *
3162  *                                  SELECTABLE
3163  *
3164  * =============================================================================== */
3165 NK_API int nk_selectable_label(struct nk_context*, const char*, nk_flags align, int *value);
3166 NK_API int nk_selectable_text(struct nk_context*, const char*, int, nk_flags align, int *value);
3167 NK_API int nk_selectable_image_label(struct nk_context*,struct nk_image,  const char*, nk_flags align,
      int *value);
3168 NK_API int nk_selectable_image_text(struct nk_context*,struct nk_image, const char*, int, nk_flags
      align, int *value);
3169 NK_API int nk_selectable_symbol_label(struct nk_context*,enum nk_symbol_type,  const char*, nk_flags
      align, int *value);
3170 NK_API int nk_selectable_symbol_text(struct nk_context*,enum nk_symbol_type, const char*, int, nk_flags
      align, int *value);
3171
3172 NK_API int nk_select_label(struct nk_context*, const char*, nk_flags align, int value);
3173 NK_API int nk_select_text(struct nk_context*, const char*, int, nk_flags align, int value);
3174 NK_API int nk_select_image_label(struct nk_context*, struct nk_image,const char*, nk_flags align, int
      value);
3175 NK_API int nk_select_image_text(struct nk_context*, struct nk_image,const char*, int, nk_flags align,
      int value);
3176 NK_API int nk_select_symbol_label(struct nk_context*,enum nk_symbol_type,  const char*, nk_flags align,
      int value);
3177 NK_API int nk_select_symbol_text(struct nk_context*,enum nk_symbol_type, const char*, int, nk_flags
      align, int value);
3178
3179 /* ===============================================================================
3180  *
3181  *                                  SLIDER
3182  *
3183  * =============================================================================== */
3184 NK_API float nk_slide_float(struct nk_context*, float min, float val, float max, float step);
3185 NK_API int nk_slide_int(struct nk_context*, int min, int val, int max, int step);
3186 NK_API int nk_slider_float(struct nk_context*, float min, float *val, float max, float step);
3187 NK_API int nk_slider_int(struct nk_context*, int min, int *val, int max, int step);
3188 /* ===============================================================================
3189  *
3190  *                                  PROGRESSBAR
3191  *
3192  * =============================================================================== */
3193 NK_API int nk_progress(struct nk_context*, nk_size *cur, nk_size max, int modifyable);
3194 NK_API nk_size nk_prog(struct nk_context*, nk_size cur, nk_size max, int modifyable);
3195
3196 /* ===============================================================================
3197  *
3198  *                                  COLOR PICKER
3199  *
3200  * =============================================================================== */
3201 NK_API struct nk_colorf nk_color_picker(struct nk_context*, struct nk_colorf, enum nk_color_format);
3202 NK_API int nk_color_pick(struct nk_context*, struct nk_colorf*, enum nk_color_format);
3203 /* ===============================================================================
3204  *
3205  *                                  PROPERTIES
3206  *
3207  * =============================================================================== */
3278 */
3279 /*/// #### nk_property_int
3298 */
3299 NK_API void nk_property_int(struct nk_context*, const char *name, int min, int *val, int max, int step,
      float inc_per_pixel);
3300 /*/// #### nk_property_float
3319 */
3320 NK_API void nk_property_float(struct nk_context*, const char *name, float min, float *val, float max,
      float step, float inc_per_pixel);
3321 /*/// #### nk_property_double
3340 */
3341 NK_API void nk_property_double(struct nk_context*, const char *name, double min, double *val, double
      max, double step, float inc_per_pixel);
3342 /*/// #### nk_propertyi
3363 */
3364 NK_API int nk_propertyi(struct nk_context*, const char *name, int min, int val, int max, int step,
      float inc_per_pixel);
3365 /*/// #### nk_propertyf
3386 */
3387 NK_API float nk_propertyf(struct nk_context*, const char *name, float min, float val, float max, float
      step, float inc_per_pixel);
3388 /*/// #### nk_propertyd
```

```
3409 */
3410 NK_API double nk_propertyd(struct nk_context*, const char *name, double min, double val, double max,
       double step, float inc_per_pixel);
3411 /* ==============================================================================
3412  *
3413  *                                  TEXT EDIT
3414  *
3415  * ============================================================================== */
3416 enum nk_edit_flags {
3417     NK_EDIT_DEFAULT                 = 0,
3418     NK_EDIT_READ_ONLY               = NK_FLAG(0),
3419     NK_EDIT_AUTO_SELECT             = NK_FLAG(1),
3420     NK_EDIT_SIG_ENTER               = NK_FLAG(2),
3421     NK_EDIT_ALLOW_TAB               = NK_FLAG(3),
3422     NK_EDIT_NO_CURSOR               = NK_FLAG(4),
3423     NK_EDIT_SELECTABLE              = NK_FLAG(5),
3424     NK_EDIT_CLIPBOARD               = NK_FLAG(6),
3425     NK_EDIT_CTRL_ENTER_NEWLINE      = NK_FLAG(7),
3426     NK_EDIT_NO_HORIZONTAL_SCROLL    = NK_FLAG(8),
3427     NK_EDIT_ALWAYS_INSERT_MODE      = NK_FLAG(9),
3428     NK_EDIT_MULTILINE               = NK_FLAG(10),
3429     NK_EDIT_GOTO_END_ON_ACTIVATE    = NK_FLAG(11)
3430 };
3431 enum nk_edit_types {
3432     NK_EDIT_SIMPLE  = NK_EDIT_ALWAYS_INSERT_MODE,
3433     NK_EDIT_FIELD   = NK_EDIT_SIMPLE|NK_EDIT_SELECTABLE|NK_EDIT_CLIPBOARD,
3434     NK_EDIT_BOX     = NK_EDIT_ALWAYS_INSERT_MODE| NK_EDIT_SELECTABLE|
     NK_EDIT_MULTILINE|NK_EDIT_ALLOW_TAB|NK_EDIT_CLIPBOARD,
3435     NK_EDIT_EDITOR  = NK_EDIT_SELECTABLE|NK_EDIT_MULTILINE|NK_EDIT_ALLOW_TAB| NK_EDIT_CLIPBOARD
3436 };
3437 enum nk_edit_events {
3438     NK_EDIT_ACTIVE      = NK_FLAG(0), /* edit widget is currently being modified */
3439     NK_EDIT_INACTIVE    = NK_FLAG(1), /* edit widget is not active and is not being modified */
3440     NK_EDIT_ACTIVATED   = NK_FLAG(2), /* edit widget went from state inactive to state active */
3441     NK_EDIT_DEACTIVATED = NK_FLAG(3), /* edit widget went from state active to state inactive */
3442     NK_EDIT_COMMITED    = NK_FLAG(4) /* edit widget has received an enter and lost focus */
3443 };
3444 NK_API nk_flags nk_edit_string(struct nk_context*, nk_flags, char *buffer, int *len, int max,
       nk_plugin_filter);
3445 NK_API nk_flags nk_edit_string_zero_terminated(struct nk_context*, nk_flags, char *buffer, int max,
       nk_plugin_filter);
3446 NK_API nk_flags nk_edit_buffer(struct nk_context*, nk_flags, struct nk_text_edit*, nk_plugin_filter);
3447 NK_API void nk_edit_focus(struct nk_context*, nk_flags flags);
3448 NK_API void nk_edit_unfocus(struct nk_context*);
3449 /* ==============================================================================
3450  *
3451  *                                  CHART
3452  *
3453  * ============================================================================== */
3454 NK_API int nk_chart_begin(struct nk_context*, enum nk_chart_type, int num, float min, float max);
3455 NK_API int nk_chart_begin_colored(struct nk_context*, enum nk_chart_type, struct nk_color, struct
       nk_color active, int num, float min, float max);
3456 NK_API void nk_chart_add_slot(struct nk_context *ctx, const enum nk_chart_type, int count, float
       min_value, float max_value);
3457 NK_API void nk_chart_add_slot_colored(struct nk_context *ctx, const enum nk_chart_type, struct
       nk_color, struct nk_color active, int count, float min_value, float max_value);
3458 NK_API nk_flags nk_chart_push(struct nk_context*, float);
3459 NK_API nk_flags nk_chart_push_slot(struct nk_context*, float, int);
3460 NK_API void nk_chart_end(struct nk_context*);
3461 NK_API void nk_plot(struct nk_context*, enum nk_chart_type, const float *values, int count, int
       offset);
3462 NK_API void nk_plot_function(struct nk_context*, enum nk_chart_type, void *userdata,
       float(*value_getter)(void* user, int index), int count, int offset);
3463 /* ==============================================================================
3464  *
3465  *                                  POPUP
3466  *
3467  * ============================================================================== */
3468 NK_API int nk_popup_begin(struct nk_context*, enum nk_popup_type, const char*, nk_flags, struct nk_rect
       bounds);
3469 NK_API void nk_popup_close(struct nk_context*);
3470 NK_API void nk_popup_end(struct nk_context*);
3471 NK_API void nk_popup_get_scroll(struct nk_context*, nk_uint *offset_x, nk_uint *offset_y);
3472 NK_API void nk_popup_set_scroll(struct nk_context*, nk_uint offset_x, nk_uint offset_y);
3473 /* ==============================================================================
3474  *
3475  *                                  COMBOBOX
3476  *
3477  * ============================================================================== */
3478 NK_API int nk_combo(struct nk_context*, const char **items, int count, int selected, int item_height,
       struct nk_vec2 size);
3479 NK_API int nk_combo_separator(struct nk_context*, const char *items_separated_by_separator, int
       separator, int selected, int count, int item_height, struct nk_vec2 size);
3480 NK_API int nk_combo_string(struct nk_context*, const char *items_separated_by_zeros, int selected, int
       count, int item_height, struct nk_vec2 size);
3481 NK_API int nk_combo_callback(struct nk_context*, void(*item_getter)(void*, int, const char**), void
       *userdata, int selected, int count, int item_height, struct nk_vec2 size);
```

```
3482 NK_API void nk_combobox(struct nk_context*, const char **items, int count, int *selected, int
       item_height, struct nk_vec2 size);
3483 NK_API void nk_combobox_string(struct nk_context*, const char *items_separated_by_zeros, int *selected,
       int count, int item_height, struct nk_vec2 size);
3484 NK_API void nk_combobox_separator(struct nk_context*, const char *items_separated_by_separator, int
       separator,int *selected, int count, int item_height, struct nk_vec2 size);
3485 NK_API void nk_combobox_callback(struct nk_context*, void(*item_getter)(void*, int, const char**),
       void*, int *selected, int count, int item_height, struct nk_vec2 size);
3486 /* =============================================================================
3487  *
3488  *                                  ABSTRACT COMBOBOX
3489  *
3490  * ============================================================================= */
3491 NK_API int nk_combo_begin_text(struct nk_context*, const char *selected, int, struct nk_vec2 size);
3492 NK_API int nk_combo_begin_label(struct nk_context*, const char *selected, struct nk_vec2 size);
3493 NK_API int nk_combo_begin_color(struct nk_context*, struct nk_color color, struct nk_vec2 size);
3494 NK_API int nk_combo_begin_symbol(struct nk_context*,  enum nk_symbol_type,  struct nk_vec2 size);
3495 NK_API int nk_combo_begin_symbol_label(struct nk_context*, const char *selected, enum nk_symbol_type,
       struct nk_vec2 size);
3496 NK_API int nk_combo_begin_symbol_text(struct nk_context*, const char *selected, int, enum
       nk_symbol_type, struct nk_vec2 size);
3497 NK_API int nk_combo_begin_image(struct nk_context*, struct nk_image img,  struct nk_vec2 size);
3498 NK_API int nk_combo_begin_image_label(struct nk_context*, const char *selected, struct nk_image, struct
       nk_vec2 size);
3499 NK_API int nk_combo_begin_image_text(struct nk_context*,  const char *selected, int, struct nk_image,
       struct nk_vec2 size);
3500 NK_API int nk_combo_item_label(struct nk_context*, const char*, nk_flags alignment);
3501 NK_API int nk_combo_item_text(struct nk_context*, const char*,int, nk_flags alignment);
3502 NK_API int nk_combo_item_image_label(struct nk_context*, struct nk_image, const char*, nk_flags
       alignment);
3503 NK_API int nk_combo_item_image_text(struct nk_context*, struct nk_image, const char*, int,nk_flags
       alignment);
3504 NK_API int nk_combo_item_symbol_label(struct nk_context*, enum nk_symbol_type, const char*, nk_flags
       alignment);
3505 NK_API int nk_combo_item_symbol_text(struct nk_context*, enum nk_symbol_type, const char*, int,
       nk_flags alignment);
3506 NK_API void nk_combo_close(struct nk_context*);
3507 NK_API void nk_combo_end(struct nk_context*);
3508 /* =============================================================================
3509  *
3510  *                                  CONTEXTUAL
3511  *
3512  * ============================================================================= */
3513 NK_API int nk_contextual_begin(struct nk_context*, nk_flags, struct nk_vec2, struct nk_rect
       trigger_bounds);
3514 NK_API int nk_contextual_item_text(struct nk_context*, const char*, int,nk_flags align);
3515 NK_API int nk_contextual_item_label(struct nk_context*, const char*, nk_flags align);
3516 NK_API int nk_contextual_item_image_label(struct nk_context*, struct nk_image, const char*, nk_flags
       alignment);
3517 NK_API int nk_contextual_item_image_text(struct nk_context*, struct nk_image, const char*, int len,
       nk_flags alignment);
3518 NK_API int nk_contextual_item_symbol_label(struct nk_context*, enum nk_symbol_type, const char*,
       nk_flags alignment);
3519 NK_API int nk_contextual_item_symbol_text(struct nk_context*, enum nk_symbol_type, const char*, int,
       nk_flags alignment);
3520 NK_API void nk_contextual_close(struct nk_context*);
3521 NK_API void nk_contextual_end(struct nk_context*);
3522 /* =============================================================================
3523  *
3524  *                                  TOOLTIP
3525  *
3526  * ============================================================================= */
3527 NK_API void nk_tooltip(struct nk_context*, const char*);
3528 #ifdef NK_INCLUDE_STANDARD_VARARGS
3529 NK_API void nk_tooltipf(struct nk_context*, NK_PRINTF_FORMAT_STRING const char*, ...)
       NK_PRINTF_VARARG_FUNC(2);
3530 NK_API void nk_tooltipfv(struct nk_context*, NK_PRINTF_FORMAT_STRING const char*, va_list)
       NK_PRINTF_VALIST_FUNC(2);
3531 #endif
3532 NK_API int nk_tooltip_begin(struct nk_context*, float width);
3533 NK_API void nk_tooltip_end(struct nk_context*);
3534 /* =============================================================================
3535  *
3536  *                                  MENU
3537  *
3538  * ============================================================================= */
3539 NK_API void nk_menubar_begin(struct nk_context*);
3540 NK_API void nk_menubar_end(struct nk_context*);
3541 NK_API int nk_menu_begin_text(struct nk_context*, const char* title, int title_len, nk_flags align,
       struct nk_vec2 size);
3542 NK_API int nk_menu_begin_label(struct nk_context*, const char*, nk_flags align, struct nk_vec2 size);
3543 NK_API int nk_menu_begin_image(struct nk_context*, const char*, struct nk_image, struct nk_vec2 size);
3544 NK_API int nk_menu_begin_image_text(struct nk_context*, const char*, int,nk_flags align,struct
       nk_image, struct nk_vec2 size);
3545 NK_API int nk_menu_begin_image_label(struct nk_context*, const char*, nk_flags align,struct nk_image,
       struct nk_vec2 size);
3546 NK_API int nk_menu_begin_symbol(struct nk_context*, const char*, enum nk_symbol_type, struct nk_vec2
```

```
    size);
3547 NK_API int nk_menu_begin_symbol_text(struct nk_context*, const char*, int,nk_flags align,enum
    nk_symbol_type, struct nk_vec2 size);
3548 NK_API int nk_menu_begin_symbol_label(struct nk_context*, const char*, nk_flags align,enum
    nk_symbol_type, struct nk_vec2 size);
3549 NK_API int nk_menu_item_text(struct nk_context*, const char*, int,nk_flags align);
3550 NK_API int nk_menu_item_label(struct nk_context*, const char*, nk_flags alignment);
3551 NK_API int nk_menu_item_image_label(struct nk_context*, struct nk_image, const char*, nk_flags
    alignment);
3552 NK_API int nk_menu_item_image_text(struct nk_context*, struct nk_image, const char*, int len, nk_flags
    alignment);
3553 NK_API int nk_menu_item_symbol_text(struct nk_context*, enum nk_symbol_type, const char*, int, nk_flags
    alignment);
3554 NK_API int nk_menu_item_symbol_label(struct nk_context*, enum nk_symbol_type, const char*, nk_flags
    alignment);
3555 NK_API void nk_menu_close(struct nk_context*);
3556 NK_API void nk_menu_end(struct nk_context*);
3557 /* ==============================================================================
3558  *
3559  *                                  STYLE
3560  *
3561  * ============================================================================== */
3562 enum nk_style_colors {
3563     NK_COLOR_TEXT,
3564     NK_COLOR_WINDOW,
3565     NK_COLOR_HEADER,
3566     NK_COLOR_BORDER,
3567     NK_COLOR_BUTTON,
3568     NK_COLOR_BUTTON_HOVER,
3569     NK_COLOR_BUTTON_ACTIVE,
3570     NK_COLOR_TOGGLE,
3571     NK_COLOR_TOGGLE_HOVER,
3572     NK_COLOR_TOGGLE_CURSOR,
3573     NK_COLOR_SELECT,
3574     NK_COLOR_SELECT_ACTIVE,
3575     NK_COLOR_SLIDER,
3576     NK_COLOR_SLIDER_CURSOR,
3577     NK_COLOR_SLIDER_CURSOR_HOVER,
3578     NK_COLOR_SLIDER_CURSOR_ACTIVE,
3579     NK_COLOR_PROPERTY,
3580     NK_COLOR_EDIT,
3581     NK_COLOR_EDIT_CURSOR,
3582     NK_COLOR_COMBO,
3583     NK_COLOR_CHART,
3584     NK_COLOR_CHART_COLOR,
3585     NK_COLOR_CHART_COLOR_HIGHLIGHT,
3586     NK_COLOR_SCROLLBAR,
3587     NK_COLOR_SCROLLBAR_CURSOR,
3588     NK_COLOR_SCROLLBAR_CURSOR_HOVER,
3589     NK_COLOR_SCROLLBAR_CURSOR_ACTIVE,
3590     NK_COLOR_TAB_HEADER,
3591     NK_COLOR_COUNT
3592 };
3593 enum nk_style_cursor {
3594     NK_CURSOR_ARROW,
3595     NK_CURSOR_TEXT,
3596     NK_CURSOR_MOVE,
3597     NK_CURSOR_RESIZE_VERTICAL,
3598     NK_CURSOR_RESIZE_HORIZONTAL,
3599     NK_CURSOR_RESIZE_TOP_LEFT_DOWN_RIGHT,
3600     NK_CURSOR_RESIZE_TOP_RIGHT_DOWN_LEFT,
3601     NK_CURSOR_COUNT
3602 };
3603 NK_API void nk_style_default(struct nk_context*);
3604 NK_API void nk_style_from_table(struct nk_context*, const struct nk_color*);
3605 NK_API void nk_style_load_cursor(struct nk_context*, enum nk_style_cursor, const struct nk_cursor*);
3606 NK_API void nk_style_load_all_cursors(struct nk_context*, struct nk_cursor*);
3607 NK_API const char* nk_style_get_color_by_name(enum nk_style_colors);
3608 NK_API void nk_style_set_font(struct nk_context*, const struct nk_user_font*);
3609 NK_API int nk_style_set_cursor(struct nk_context*, enum nk_style_cursor);
3610 NK_API void nk_style_show_cursor(struct nk_context*);
3611 NK_API void nk_style_hide_cursor(struct nk_context*);
3612
3613 NK_API int nk_style_push_font(struct nk_context*, const struct nk_user_font*);
3614 NK_API int nk_style_push_float(struct nk_context*, float*, float);
3615 NK_API int nk_style_push_vec2(struct nk_context*, struct nk_vec2*, struct nk_vec2);
3616 NK_API int nk_style_push_style_item(struct nk_context*, struct nk_style_item*, struct nk_style_item);
3617 NK_API int nk_style_push_flags(struct nk_context*, nk_flags*, nk_flags);
3618 NK_API int nk_style_push_color(struct nk_context*, struct nk_color*, struct nk_color);
3619
3620 NK_API int nk_style_pop_font(struct nk_context*);
3621 NK_API int nk_style_pop_float(struct nk_context*);
3622 NK_API int nk_style_pop_vec2(struct nk_context*);
3623 NK_API int nk_style_pop_style_item(struct nk_context*);
3624 NK_API int nk_style_pop_flags(struct nk_context*);
3625 NK_API int nk_style_pop_color(struct nk_context*);
3626 /* ==============================================================================
```

```
3627  *
3628  *                                  COLOR
3629  *
3630  * ============================================================================== */
3631 NK_API struct nk_color nk_rgb(int r, int g, int b);
3632 NK_API struct nk_color nk_rgb_iv(const int *rgb);
3633 NK_API struct nk_color nk_rgb_bv(const nk_byte* rgb);
3634 NK_API struct nk_color nk_rgb_f(float r, float g, float b);
3635 NK_API struct nk_color nk_rgb_fv(const float *rgb);
3636 NK_API struct nk_color nk_rgb_cf(struct nk_colorf c);
3637 NK_API struct nk_color nk_rgb_hex(const char *rgb);
3638
3639 NK_API struct nk_color nk_rgba(int r, int g, int b, int a);
3640 NK_API struct nk_color nk_rgba_u32(nk_uint);
3641 NK_API struct nk_color nk_rgba_iv(const int *rgba);
3642 NK_API struct nk_color nk_rgba_bv(const nk_byte *rgba);
3643 NK_API struct nk_color nk_rgba_f(float r, float g, float b, float a);
3644 NK_API struct nk_color nk_rgba_fv(const float *rgba);
3645 NK_API struct nk_color nk_rgba_cf(struct nk_colorf c);
3646 NK_API struct nk_color nk_rgba_hex(const char *rgb);
3647
3648 NK_API struct nk_colorf nk_hsva_colorf(float h, float s, float v, float a);
3649 NK_API struct nk_colorf nk_hsva_colorfv(float *c);
3650 NK_API void nk_colorf_hsva_f(float *out_h, float *out_s, float *out_v, float *out_a, struct nk_colorf
      in);
3651 NK_API void nk_colorf_hsva_fv(float *hsva, struct nk_colorf in);
3652
3653 NK_API struct nk_color nk_hsv(int h, int s, int v);
3654 NK_API struct nk_color nk_hsv_iv(const int *hsv);
3655 NK_API struct nk_color nk_hsv_bv(const nk_byte *hsv);
3656 NK_API struct nk_color nk_hsv_f(float h, float s, float v);
3657 NK_API struct nk_color nk_hsv_fv(const float *hsv);
3658
3659 NK_API struct nk_color nk_hsva(int h, int s, int v, int a);
3660 NK_API struct nk_color nk_hsva_iv(const int *hsva);
3661 NK_API struct nk_color nk_hsva_bv(const nk_byte *hsva);
3662 NK_API struct nk_color nk_hsva_f(float h, float s, float v, float a);
3663 NK_API struct nk_color nk_hsva_fv(const float *hsva);
3664
3665 /* color (conversion nuklear --> user) */
3666 NK_API void nk_color_f(float *r, float *g, float *b, float *a, struct nk_color);
3667 NK_API void nk_color_fv(float *rgba_out, struct nk_color);
3668 NK_API struct nk_colorf nk_color_cf(struct nk_color);
3669 NK_API void nk_color_d(double *r, double *g, double *b, double *a, struct nk_color);
3670 NK_API void nk_color_dv(double *rgba_out, struct nk_color);
3671
3672 NK_API nk_uint nk_color_u32(struct nk_color);
3673 NK_API void nk_color_hex_rgba(char *output, struct nk_color);
3674 NK_API void nk_color_hex_rgb(char *output, struct nk_color);
3675
3676 NK_API void nk_color_hsv_i(int *out_h, int *out_s, int *out_v, struct nk_color);
3677 NK_API void nk_color_hsv_b(nk_byte *out_h, nk_byte *out_s, nk_byte *out_v, struct nk_color);
3678 NK_API void nk_color_hsv_iv(int *hsv_out, struct nk_color);
3679 NK_API void nk_color_hsv_bv(nk_byte *hsv_out, struct nk_color);
3680 NK_API void nk_color_hsv_f(float *out_h, float *out_s, float *out_v, struct nk_color);
3681 NK_API void nk_color_hsv_fv(float *hsv_out, struct nk_color);
3682
3683 NK_API void nk_color_hsva_i(int *h, int *s, int *v, int *a, struct nk_color);
3684 NK_API void nk_color_hsva_b(nk_byte *h, nk_byte *s, nk_byte *v, nk_byte *a, struct nk_color);
3685 NK_API void nk_color_hsva_iv(int *hsva_out, struct nk_color);
3686 NK_API void nk_color_hsva_bv(nk_byte *hsva_out, struct nk_color);
3687 NK_API void nk_color_hsva_f(float *out_h, float *out_s, float *out_v, float *out_a, struct nk_color);
3688 NK_API void nk_color_hsva_fv(float *hsva_out, struct nk_color);
3689 /* ============================================================================
3690  *
3691  *                                  IMAGE
3692  *
3693  * ============================================================================== */
3694 NK_API nk_handle nk_handle_ptr(void*);
3695 NK_API nk_handle nk_handle_id(int);
3696 NK_API struct nk_image nk_image_handle(nk_handle);
3697 NK_API struct nk_image nk_image_ptr(void*);
3698 NK_API struct nk_image nk_image_id(int);
3699 NK_API int nk_image_is_subimage(const struct nk_image* img);
3700 NK_API struct nk_image nk_subimage_ptr(void*, unsigned short w, unsigned short h, struct nk_rect
      sub_region);
3701 NK_API struct nk_image nk_subimage_id(int, unsigned short w, unsigned short h, struct nk_rect
      sub_region);
3702 NK_API struct nk_image nk_subimage_handle(nk_handle, unsigned short w, unsigned short h, struct nk_rect
      sub_region);
3703 /* ============================================================================
3704  *
3705  *                                  MATH
3706  *
3707  * ============================================================================== */
3708 NK_API nk_hash nk_murmur_hash(const void *key, int len, nk_hash seed);
3709 NK_API void nk_triangle_from_direction(struct nk_vec2 *result, struct nk_rect r, float pad_x, float
```

```
    pad_y, enum nk_heading);
3710
3711 NK_API struct nk_vec2 nk_vec2(float x, float y);
3712 NK_API struct nk_vec2 nk_vec2i(int x, int y);
3713 NK_API struct nk_vec2 nk_vec2v(const float *xy);
3714 NK_API struct nk_vec2 nk_vec2iv(const int *xy);
3715
3716 NK_API struct nk_rect nk_get_null_rect(void);
3717 NK_API struct nk_rect nk_rect(float x, float y, float w, float h);
3718 NK_API struct nk_rect nk_recti(int x, int y, int w, int h);
3719 NK_API struct nk_rect nk_recta(struct nk_vec2 pos, struct nk_vec2 size);
3720 NK_API struct nk_rect nk_rectv(const float *xywh);
3721 NK_API struct nk_rect nk_rectiv(const int *xywh);
3722 NK_API struct nk_vec2 nk_rect_pos(struct nk_rect);
3723 NK_API struct nk_vec2 nk_rect_size(struct nk_rect);
3724 /* ============================================================================
3725  *
3726  *                                  STRING
3727  *
3728  * ============================================================================ */
3729 NK_API int nk_strlen(const char *str);
3730 NK_API int nk_stricmp(const char *s1, const char *s2);
3731 NK_API int nk_stricmpn(const char *s1, const char *s2, int n);
3732 NK_API int nk_strtoi(const char *str, const char **endptr);
3733 NK_API float nk_strtof(const char *str, const char **endptr);
3734 NK_API double nk_strtod(const char *str, const char **endptr);
3735 NK_API int nk_strfilter(const char *text, const char *regexp);
3736 NK_API int nk_strmatch_fuzzy_string(char const *str, char const *pattern, int *out_score);
3737 NK_API int nk_strmatch_fuzzy_text(const char *txt, int txt_len, const char *pattern, int *out_score);
3738 /* ============================================================================
3739  *
3740  *                                  UTF-8
3741  *
3742  * ============================================================================ */
3743 NK_API int nk_utf_decode(const char*, nk_rune*, int);
3744 NK_API int nk_utf_encode(nk_rune, char*, int);
3745 NK_API int nk_utf_len(const char*, int byte_len);
3746 NK_API const char* nk_utf_at(const char *buffer, int length, int index, nk_rune *unicode, int *len);
3747 /* ===============================================================
3748  *
3749  *                          FONT
3750  *
3751  * ===============================================================*/
3752 /*  Font handling in this library was designed to be quite customizable and lets
3753     you decide what you want to use and what you want to provide. There are three
3754     different ways to use the font atlas. The first two will use your font
3755     handling scheme and only requires essential data to run nuklear. The next
3756     slightly more advanced features is font handling with vertex buffer output.
3757     Finally the most complex API wise is using nuklear's font baking API.
3758
3759     1.) Using your own implementation without vertex buffer output
3760     --------------------------------------------------------------
3761     So first up the easiest way to do font handling is by just providing a
3762     'nk_user_font' struct which only requires the height in pixel of the used
3763     font and a callback to calculate the width of a string. This way of handling
3764     fonts is best fitted for using the normal draw shape command API where you
3765     do all the text drawing yourself and the library does not require any kind
3766     of deeper knowledge about which font handling mechanism you use.
3767     IMPORTANT: the 'nk_user_font' pointer provided to nuklear has to persist
3768     over the complete life time! I know this sucks but it is currently the only
3769     way to switch between fonts.
3770
3771         float your_text_width_calculation(nk_handle handle, float height, const char *text, int len)
3772         {
3773             your_font_type *type = handle.ptr;
3774             float text_width = ...;
3775             return text_width;
3776         }
3777
3778         struct nk_user_font font;
3779         font.userdata.ptr = &your_font_class_or_struct;
3780         font.height = your_font_height;
3781         font.width = your_text_width_calculation;
3782
3783         struct nk_context ctx;
3784         nk_init_default(&ctx, &font);
3785
3786     2.) Using your own implementation with vertex buffer output
3787     --------------------------------------------------------------
3788     While the first approach works fine if you don't want to use the optional
3789     vertex buffer output it is not enough if you do. To get font handling working
3790     for these cases you have to provide two additional parameters inside the
3791     'nk_user_font'. First a texture atlas handle used to draw text as subimages
3792     of a bigger font atlas texture and a callback to query a character's glyph
3793     information (offset, size, ...). So it is still possible to provide your own
3794     font and use the vertex buffer output.
3795
```

```
3796         float your_text_width_calculation(nk_handle handle, float height, const char *text, int len)
3797         {
3798             your_font_type *type = handle.ptr;
3799             float text_width = ...;
3800             return text_width;
3801         }
3802         void query_your_font_glyph(nk_handle handle, float font_height, struct nk_user_font_glyph
     *glyph, nk_rune codepoint, nk_rune next_codepoint)
3803         {
3804             your_font_type *type = handle.ptr;
3805             glyph.width = ...;
3806             glyph.height = ...;
3807             glyph.xadvance = ...;
3808             glyph.uv[0].x = ...;
3809             glyph.uv[0].y = ...;
3810             glyph.uv[1].x = ...;
3811             glyph.uv[1].y = ...;
3812             glyph.offset.x = ...;
3813             glyph.offset.y = ...;
3814         }
3815
3816         struct nk_user_font font;
3817         font.userdata.ptr = &your_font_class_or_struct;
3818         font.height = your_font_height;
3819         font.width = your_text_width_calculation;
3820         font.query = query_your_font_glyph;
3821         font.texture.id = your_font_texture;
3822
3823         struct nk_context ctx;
3824         nk_init_default(&ctx, &font);
3825
3826     3.) Nuklear font baker
3827     ------------------------------------
3828     The final approach if you do not have a font handling functionality or don't
3829     want to use it in this library is by using the optional font baker.
3830     The font baker APIs can be used to create a font plus font atlas texture
3831     and can be used with or without the vertex buffer output.
3832
3833     It still uses the `nk_user_font` struct and the two different approaches
3834     previously stated still work. The font baker is not located inside
3835     `nk_context` like all other systems since it can be understood as more of
3836     an extension to nuklear and does not really depend on any `nk_context` state.
3837
3838     Font baker need to be initialized first by one of the nk_font_atlas_init_xxx
3839     functions. If you don't care about memory just call the default version
3840     `nk_font_atlas_init_default` which will allocate all memory from the standard library.
3841     If you want to control memory allocation but you don't care if the allocated
3842     memory is temporary and therefore can be freed directly after the baking process
3843     is over or permanent you can call `nk_font_atlas_init`.
3844
3845     After successfully initializing the font baker you can add Truetype(.ttf) fonts from
3846     different sources like memory or from file by calling one of the `nk_font_atlas_add_xxx`.
3847     functions. Adding font will permanently store each font, font config and ttf memory block(!)
3848     inside the font atlas and allows to reuse the font atlas. If you don't want to reuse
3849     the font baker by for example adding additional fonts you can call
3850     `nk_font_atlas_cleanup` after the baking process is over (after calling nk_font_atlas_end).
3851
3852     As soon as you added all fonts you wanted you can now start the baking process
3853     for every selected glyph to image by calling `nk_font_atlas_bake`.
3854     The baking process returns image memory, width and height which can be used to
3855     either create your own image object or upload it to any graphics library.
3856     No matter which case you finally have to call `nk_font_atlas_end` which
3857     will free all temporary memory including the font atlas image so make sure
3858     you created our texture beforehand. `nk_font_atlas_end` requires a handle
3859     to your font texture or object and optionally fills a `struct nk_draw_null_texture`
3860     which can be used for the optional vertex output. If you don't want it just
3861     set the argument to `NULL`.
3862
3863     At this point you are done and if you don't want to reuse the font atlas you
3864     can call `nk_font_atlas_cleanup` to free all truetype blobs and configuration
3865     memory. Finally if you don't use the font atlas and any of it's fonts anymore
3866     you need to call `nk_font_atlas_clear` to free all memory still being used.
3867
3868         struct nk_font_atlas atlas;
3869         nk_font_atlas_init_default(&atlas);
3870         nk_font_atlas_begin(&atlas);
3871         nk_font *font = nk_font_atlas_add_from_file(&atlas, "Path/To/Your/TTF_Font.ttf", 13, 0);
3872         nk_font *font2 = nk_font_atlas_add_from_file(&atlas, "Path/To/Your/TTF_Font2.ttf", 16, 0);
3873         const void* img = nk_font_atlas_bake(&atlas, &img_width, &img_height, NK_FONT_ATLAS_RGBA32);
3874         nk_font_atlas_end(&atlas, nk_handle_id(texture), 0);
3875
3876         struct nk_context ctx;
3877         nk_init_default(&ctx, &font->handle);
3878         while (1) {
3879
3880         }
3881         nk_font_atlas_clear(&atlas);
```

```
3882
3883     The font baker API is probably the most complex API inside this library and
3884     I would suggest reading some of my examples `example/` to get a grip on how
3885     to use the font atlas. There are a number of details I left out. For example
3886     how to merge fonts, configure a font with `nk_font_config` to use other languages,
3887     use another texture coordinate format and a lot more:
3888
3889         struct nk_font_config cfg = nk_font_config(font_pixel_height);
3890         cfg.merge_mode = nk_false or nk_true;
3891         cfg.range = nk_font_korean_glyph_ranges();
3892         cfg.coord_type = NK_COORD_PIXEL;
3893         nk_font *font = nk_font_atlas_add_from_file(&atlas, "Path/To/Your/TTF_Font.ttf", 13, &cfg);
3894
3895 */
3896 struct nk_user_font_glyph;
3897 typedef float(*nk_text_width_f)(nk_handle, float h, const char*, int len);
3898 typedef void(*nk_query_font_glyph_f)(nk_handle handle, float font_height,
3899                                      struct nk_user_font_glyph *glyph,
3900                                      nk_rune codepoint, nk_rune next_codepoint);
3901
3902 #if defined(NK_INCLUDE_VERTEX_BUFFER_OUTPUT) || defined(NK_INCLUDE_SOFTWARE_FONT)
3903 struct nk_user_font_glyph {
3904     struct nk_vec2 uv[2];
3905     /* texture coordinates */
3906     struct nk_vec2 offset;
3907     /* offset between top left and glyph */
3908     float width, height;
3909     /* size of the glyph  */
3910     float xadvance;
3911     /* offset to the next glyph */
3912 };
3913 #endif
3914
3915 struct nk_user_font {
3916     nk_handle userdata;
3917     /* user provided font handle */
3918     float height;
3919     /* max height of the font */
3920     nk_text_width_f width;
3921     /* font string width in pixel callback */
3922 #ifdef NK_INCLUDE_VERTEX_BUFFER_OUTPUT
3923     nk_query_font_glyph_f query;
3924     /* font glyph callback to query drawing info */
3925     nk_handle texture;
3926     /* texture handle to the used font atlas or texture */
3927 #endif
3928 };
3929
3930 #ifdef NK_INCLUDE_FONT_BAKING
3931 enum nk_font_coord_type {
3932     NK_COORD_UV, /* texture coordinates inside font glyphs are clamped between 0-1 */
3933     NK_COORD_PIXEL /* texture coordinates inside font glyphs are in absolute pixel */
3934 };
3935
3936 struct nk_font;
3937 struct nk_baked_font {
3938     float height;
3939     /* height of the font  */
3940     float ascent, descent;
3941     /* font glyphs ascent and descent  */
3942     nk_rune glyph_offset;
3943     /* glyph array offset inside the font glyph baking output array  */
3944     nk_rune glyph_count;
3945     /* number of glyphs of this font inside the glyph baking array output */
3946     const nk_rune *ranges;
3947     /* font codepoint ranges as pairs of (from/to) and 0 as last element */
3948 };
3949
3950 struct nk_font_config {
3951     struct nk_font_config *next;
3952     /* NOTE: only used internally */
3953     void *ttf_blob;
3954     /* pointer to loaded TTF file memory block.
3955      * NOTE: not needed for nk_font_atlas_add_from_memory and nk_font_atlas_add_from_file. */
3956     nk_size ttf_size;
3957     /* size of the loaded TTF file memory block
3958      * NOTE: not needed for nk_font_atlas_add_from_memory and nk_font_atlas_add_from_file. */
3959
3960     unsigned char ttf_data_owned_by_atlas;
3961     /* used inside font atlas: default to: 0*/
3962     unsigned char merge_mode;
3963     /* merges this font into the last font */
3964     unsigned char pixel_snap;
3965     /* align every character to pixel boundary (if true set oversample (1,1)) */
3966     unsigned char oversample_v, oversample_h;
3967     /* rasterize at hight quality for sub-pixel position */
3968     unsigned char padding[3];
```

```
3969
3970     float size;
3971     /* baked pixel height of the font */
3972     enum nk_font_coord_type coord_type;
3973     /* texture coordinate format with either pixel or UV coordinates */
3974     struct nk_vec2 spacing;
3975     /* extra pixel spacing between glyphs  */
3976     const nk_rune *range;
3977     /* list of unicode ranges (2 values per range, zero terminated) */
3978     struct nk_baked_font *font;
3979     /* font to setup in the baking process: NOTE: not needed for font atlas */
3980     nk_rune fallback_glyph;
3981     /* fallback glyph to use if a given rune is not found */
3982     struct nk_font_config *n;
3983     struct nk_font_config *p;
3984 };
3985
3986 struct nk_font_glyph {
3987     nk_rune codepoint;
3988     float xadvance;
3989     float x0, y0, x1, y1, w, h;
3990     float u0, v0, u1, v1;
3991 };
3992
3993 struct nk_font {
3994     struct nk_font *next;
3995     struct nk_user_font handle;
3996     struct nk_baked_font info;
3997     float scale;
3998     struct nk_font_glyph *glyphs;
3999     const struct nk_font_glyph *fallback;
4000     nk_rune fallback_codepoint;
4001     nk_handle texture;
4002     struct nk_font_config *config;
4003 };
4004
4005 enum nk_font_atlas_format {
4006     NK_FONT_ATLAS_ALPHA8,
4007     NK_FONT_ATLAS_RGBA32
4008 };
4009
4010 struct nk_font_atlas {
4011     void *pixel;
4012     int tex_width;
4013     int tex_height;
4014
4015     struct nk_allocator permanent;
4016     struct nk_allocator temporary;
4017
4018     struct nk_recti custom;
4019     struct nk_cursor cursors[NK_CURSOR_COUNT];
4020
4021     int glyph_count;
4022     struct nk_font_glyph *glyphs;
4023     struct nk_font *default_font;
4024     struct nk_font *fonts;
4025     struct nk_font_config *config;
4026     int font_num;
4027 };
4028
4029 /* some language glyph codepoint ranges */
4030 NK_API const nk_rune *nk_font_default_glyph_ranges(void);
4031 NK_API const nk_rune *nk_font_chinese_glyph_ranges(void);
4032 NK_API const nk_rune *nk_font_cyrillic_glyph_ranges(void);
4033 NK_API const nk_rune *nk_font_korean_glyph_ranges(void);
4034
4035 #ifdef NK_INCLUDE_DEFAULT_ALLOCATOR
4036 NK_API void nk_font_atlas_init_default(struct nk_font_atlas*);
4037 #endif
4038 NK_API void nk_font_atlas_init(struct nk_font_atlas*, struct nk_allocator*);
4039 NK_API void nk_font_atlas_init_custom(struct nk_font_atlas*, struct nk_allocator *persistent, struct
       nk_allocator *transient);
4040 NK_API void nk_font_atlas_begin(struct nk_font_atlas*);
4041 NK_API struct nk_font_config nk_font_config(float pixel_height);
4042 NK_API struct nk_font *nk_font_atlas_add(struct nk_font_atlas*, const struct nk_font_config*);
4043 #ifdef NK_INCLUDE_DEFAULT_FONT
4044 NK_API struct nk_font* nk_font_atlas_add_default(struct nk_font_atlas*, float height, const struct
       nk_font_config*);
4045 #endif
4046 NK_API struct nk_font* nk_font_atlas_add_from_memory(struct nk_font_atlas *atlas, void *memory, nk_size
       size, float height, const struct nk_font_config *config);
4047 #ifdef NK_INCLUDE_STANDARD_IO
4048 NK_API struct nk_font* nk_font_atlas_add_from_file(struct nk_font_atlas *atlas, const char *file_path,
       float height, const struct nk_font_config*);
4049 #endif
4050 NK_API struct nk_font *nk_font_atlas_add_compressed(struct nk_font_atlas*, void *memory, nk_size size,
       float height, const struct nk_font_config*);
```

```
4051 NK_API struct nk_font* nk_font_atlas_add_compressed_base85(struct nk_font_atlas*, const char *data,
       float height, const struct nk_font_config *config);
4052 NK_API const void* nk_font_atlas_bake(struct nk_font_atlas*, int *width, int *height, enum
       nk_font_atlas_format);
4053 NK_API void nk_font_atlas_end(struct nk_font_atlas*, nk_handle tex, struct nk_draw_null_texture*);
4054 NK_API const struct nk_font_glyph* nk_font_find_glyph(struct nk_font*, nk_rune unicode);
4055 NK_API void nk_font_atlas_cleanup(struct nk_font_atlas *atlas);
4056 NK_API void nk_font_atlas_clear(struct nk_font_atlas*);
4057
4058 #endif
4059
4060 /* ==============================================================
4061  *
4062  *                          MEMORY BUFFER
4063  *
4064  * ==============================================================*/
4065 /*  A basic (double)-buffer with linear allocation and resetting as only
4066     freeing policy. The buffer's main purpose is to control all memory management
4067     inside the GUI toolkit and still leave memory control as much as possible in
4068     the hand of the user while also making sure the library is easy to use if
4069     not as much control is needed.
4070     In general all memory inside this library can be provided from the user in
4071     three different ways.
4072
4073     The first way and the one providing most control is by just passing a fixed
4074     size memory block. In this case all control lies in the hand of the user
4075     since he can exactly control where the memory comes from and how much memory
4076     the library should consume. Of course using the fixed size API removes the
4077     ability to automatically resize a buffer if not enough memory is provided so
4078     you have to take over the resizing. While being a fixed sized buffer sounds
4079     quite limiting, it is very effective in this library since the actual memory
4080     consumption is quite stable and has a fixed upper bound for a lot of cases.
4081
4082     If you don't want to think about how much memory the library should allocate
4083     at all time or have a very dynamic UI with unpredictable memory consumption
4084     habits but still want control over memory allocation you can use the dynamic
4085     allocator based API. The allocator consists of two callbacks for allocating
4086     and freeing memory and optional userdata so you can plugin your own allocator.
4087
4088     The final and easiest way can be used by defining
4089     NK_INCLUDE_DEFAULT_ALLOCATOR which uses the standard library memory
4090     allocation functions malloc and free and takes over complete control over
4091     memory in this library.
4092 */
4093 struct nk_memory_status {
4094     void *memory;
4095     unsigned int type;
4096     nk_size size;
4097     nk_size allocated;
4098     nk_size needed;
4099     nk_size calls;
4100 };
4101
4102 enum nk_allocation_type {
4103     NK_BUFFER_FIXED,
4104     NK_BUFFER_DYNAMIC
4105 };
4106
4107 enum nk_buffer_allocation_type {
4108     NK_BUFFER_FRONT,
4109     NK_BUFFER_BACK,
4110     NK_BUFFER_MAX
4111 };
4112
4113 struct nk_buffer_marker {
4114     int active;
4115     nk_size offset;
4116 };
4117
4118 struct nk_memory {void *ptr;nk_size size;};
4119 struct nk_buffer {
4120     struct nk_buffer_marker marker[NK_BUFFER_MAX];
4121     /* buffer marker to free a buffer to a certain offset */
4122     struct nk_allocator pool;
4123     /* allocator callback for dynamic buffers */
4124     enum nk_allocation_type type;
4125     /* memory management type */
4126     struct nk_memory memory;
4127     /* memory and size of the current memory block */
4128     float grow_factor;
4129     /* growing factor for dynamic memory management */
4130     nk_size allocated;
4131     /* total amount of memory allocated */
4132     nk_size needed;
4133     /* totally consumed memory given that enough memory is present */
4134     nk_size calls;
4135     /* number of allocation calls */
```

```
4136     nk_size size;
4137     /* current size of the buffer */
4138 };
4139
4140 #ifdef NK_INCLUDE_DEFAULT_ALLOCATOR
4141 NK_API void nk_buffer_init_default(struct nk_buffer*);
4142 #endif
4143 NK_API void nk_buffer_init(struct nk_buffer*, const struct nk_allocator*, nk_size size);
4144 NK_API void nk_buffer_init_fixed(struct nk_buffer*, void *memory, nk_size size);
4145 NK_API void nk_buffer_info(struct nk_memory_status*, struct nk_buffer*);
4146 NK_API void nk_buffer_push(struct nk_buffer*, enum nk_buffer_allocation_type type, const void *memory,
       nk_size size, nk_size align);
4147 NK_API void nk_buffer_mark(struct nk_buffer*, enum nk_buffer_allocation_type type);
4148 NK_API void nk_buffer_reset(struct nk_buffer*, enum nk_buffer_allocation_type type);
4149 NK_API void nk_buffer_clear(struct nk_buffer*);
4150 NK_API void nk_buffer_free(struct nk_buffer*);
4151 NK_API void *nk_buffer_memory(struct nk_buffer*);
4152 NK_API const void *nk_buffer_memory_const(const struct nk_buffer*);
4153 NK_API nk_size nk_buffer_total(struct nk_buffer*);
4154
4155 /* ==============================================================
4156  *
4157  *                          STRING
4158  *
4159  * ==============================================================*/
4160 /*  Basic string buffer which is only used in context with the text editor
4161  *  to manage and manipulate dynamic or fixed size string content. This is _NOT_
4162  *  the default string handling method. The only instance you should have any contact
4163  *  with this API is if you interact with an `nk_text_edit` object inside one of the
4164  *  copy and paste functions and even there only for more advanced cases. */
4165 struct nk_str {
4166     struct nk_buffer buffer;
4167     int len; /* in codepoints/runes/glyphs */
4168 };
4169
4170 #ifdef NK_INCLUDE_DEFAULT_ALLOCATOR
4171 NK_API void nk_str_init_default(struct nk_str*);
4172 #endif
4173 NK_API void nk_str_init(struct nk_str*, const struct nk_allocator*, nk_size size);
4174 NK_API void nk_str_init_fixed(struct nk_str*, void *memory, nk_size size);
4175 NK_API void nk_str_clear(struct nk_str*);
4176 NK_API void nk_str_free(struct nk_str*);
4177
4178 NK_API int nk_str_append_text_char(struct nk_str*, const char*, int);
4179 NK_API int nk_str_append_str_char(struct nk_str*, const char*);
4180 NK_API int nk_str_append_text_utf8(struct nk_str*, const char*, int);
4181 NK_API int nk_str_append_str_utf8(struct nk_str*, const char*);
4182 NK_API int nk_str_append_text_runes(struct nk_str*, const nk_rune*, int);
4183 NK_API int nk_str_append_str_runes(struct nk_str*, const nk_rune*);
4184
4185 NK_API int nk_str_insert_at_char(struct nk_str*, int pos, const char*, int);
4186 NK_API int nk_str_insert_at_rune(struct nk_str*, int pos, const char*, int);
4187
4188 NK_API int nk_str_insert_text_char(struct nk_str*, int pos, const char*, int);
4189 NK_API int nk_str_insert_str_char(struct nk_str*, int pos, const char*);
4190 NK_API int nk_str_insert_text_utf8(struct nk_str*, int pos, const char*, int);
4191 NK_API int nk_str_insert_str_utf8(struct nk_str*, int pos, const char*);
4192 NK_API int nk_str_insert_text_runes(struct nk_str*, int pos, const nk_rune*, int);
4193 NK_API int nk_str_insert_str_runes(struct nk_str*, int pos, const nk_rune*);
4194
4195 NK_API void nk_str_remove_chars(struct nk_str*, int len);
4196 NK_API void nk_str_remove_runes(struct nk_str *str, int len);
4197 NK_API void nk_str_delete_chars(struct nk_str*, int pos, int len);
4198 NK_API void nk_str_delete_runes(struct nk_str*, int pos, int len);
4199
4200 NK_API char *nk_str_at_char(struct nk_str*, int pos);
4201 NK_API char *nk_str_at_rune(struct nk_str*, int pos, nk_rune *unicode, int *len);
4202 NK_API nk_rune nk_str_rune_at(const struct nk_str*, int pos);
4203 NK_API const char *nk_str_at_char_const(const struct nk_str*, int pos);
4204 NK_API const char *nk_str_at_const(const struct nk_str*, int pos, nk_rune *unicode, int *len);
4205
4206 NK_API char *nk_str_get(struct nk_str*);
4207 NK_API const char *nk_str_get_const(const struct nk_str*);
4208 NK_API int nk_str_len(struct nk_str*);
4209 NK_API int nk_str_len_char(struct nk_str*);
4210
4211 /*==============================================================
4212  *
4213  *                      TEXT EDITOR
4214  *
4215  * ==============================================================*/
4216 /* Editing text in this library is handled by either `nk_edit_string` or
4217  * `nk_edit_buffer`. But like almost everything in this library there are multiple
4218  * ways of doing it and a balance between control and ease of use with memory
4219  * as well as functionality controlled by flags.
4220  *
4221  * This library generally allows three different levels of memory control:
```

```
4222  * First of is the most basic way of just providing a simple char array with
4223  * string length. This method is probably the easiest way of handling simple
4224  * user text input. Main upside is complete control over memory while the biggest
4225  * downside in comparison with the other two approaches is missing undo/redo.
4226  *
4227  * For UIs that require undo/redo the second way was created. It is based on
4228  * a fixed size nk_text_edit struct, which has an internal undo/redo stack.
4229  * This is mainly useful if you want something more like a text editor but don't want
4230  * to have a dynamically growing buffer.
4231  *
4232  * The final way is using a dynamically growing nk_text_edit struct, which
4233  * has both a default version if you don't care where memory comes from and an
4234  * allocator version if you do. While the text editor is quite powerful for its
4235  * complexity I would not recommend editing gigabytes of data with it.
4236  * It is rather designed for uses cases which make sense for a GUI library not for
4237  * an full blown text editor.
4238  */
4239 #ifndef NK_TEXTEDIT_UNDOSTATECOUNT
4240 #define NK_TEXTEDIT_UNDOSTATECOUNT     99
4241 #endif
4242
4243 #ifndef NK_TEXTEDIT_UNDOCHARCOUNT
4244 #define NK_TEXTEDIT_UNDOCHARCOUNT      999
4245 #endif
4246
4247 struct nk_text_edit;
4248 struct nk_clipboard {
4249     nk_handle userdata;
4250     nk_plugin_paste paste;
4251     nk_plugin_copy copy;
4252 };
4253
4254 struct nk_text_undo_record {
4255    int where;
4256    short insert_length;
4257    short delete_length;
4258    short char_storage;
4259 };
4260
4261 struct nk_text_undo_state {
4262    struct nk_text_undo_record undo_rec[NK_TEXTEDIT_UNDOSTATECOUNT];
4263    nk_rune undo_char[NK_TEXTEDIT_UNDOCHARCOUNT];
4264    short undo_point;
4265    short redo_point;
4266    short undo_char_point;
4267    short redo_char_point;
4268 };
4269
4270 enum nk_text_edit_type {
4271     NK_TEXT_EDIT_SINGLE_LINE,
4272     NK_TEXT_EDIT_MULTI_LINE
4273 };
4274
4275 enum nk_text_edit_mode {
4276     NK_TEXT_EDIT_MODE_VIEW,
4277     NK_TEXT_EDIT_MODE_INSERT,
4278     NK_TEXT_EDIT_MODE_REPLACE
4279 };
4280
4281 struct nk_text_edit {
4282     struct nk_clipboard clip;
4283     struct nk_str string;
4284     nk_plugin_filter filter;
4285     struct nk_vec2 scrollbar;
4286
4287     int cursor;
4288     int select_start;
4289     int select_end;
4290     unsigned char mode;
4291     unsigned char cursor_at_end_of_line;
4292     unsigned char initialized;
4293     unsigned char has_preferred_x;
4294     unsigned char single_line;
4295     unsigned char active;
4296     unsigned char padding1;
4297     float preferred_x;
4298     struct nk_text_undo_state undo;
4299 };
4300
4301 /* filter function */
4302 NK_API int nk_filter_default(const struct nk_text_edit*, nk_rune unicode);
4303 NK_API int nk_filter_ascii(const struct nk_text_edit*, nk_rune unicode);
4304 NK_API int nk_filter_float(const struct nk_text_edit*, nk_rune unicode);
4305 NK_API int nk_filter_decimal(const struct nk_text_edit*, nk_rune unicode);
4306 NK_API int nk_filter_hex(const struct nk_text_edit*, nk_rune unicode);
4307 NK_API int nk_filter_oct(const struct nk_text_edit*, nk_rune unicode);
4308 NK_API int nk_filter_binary(const struct nk_text_edit*, nk_rune unicode);
```

```
4309
4310 /* text editor */
4311 #ifdef NK_INCLUDE_DEFAULT_ALLOCATOR
4312 NK_API void nk_textedit_init_default(struct nk_text_edit*);
4313 #endif
4314 NK_API void nk_textedit_init(struct nk_text_edit*, struct nk_allocator*, nk_size size);
4315 NK_API void nk_textedit_init_fixed(struct nk_text_edit*, void *memory, nk_size size);
4316 NK_API void nk_textedit_free(struct nk_text_edit*);
4317 NK_API void nk_textedit_text(struct nk_text_edit*, const char*, int total_len);
4318 NK_API void nk_textedit_delete(struct nk_text_edit*, int where, int len);
4319 NK_API void nk_textedit_delete_selection(struct nk_text_edit*);
4320 NK_API void nk_textedit_select_all(struct nk_text_edit*);
4321 NK_API int nk_textedit_cut(struct nk_text_edit*);
4322 NK_API int nk_textedit_paste(struct nk_text_edit*, char const*, int len);
4323 NK_API void nk_textedit_undo(struct nk_text_edit*);
4324 NK_API void nk_textedit_redo(struct nk_text_edit*);
4325
4326 /* ===============================================================
4327  *
4328  *                          DRAWING
4329  *
4330  * ===============================================================*/
4331 /*  This library was designed to be render backend agnostic so it does
4332     not draw anything to screen. Instead all drawn shapes, widgets
4333     are made of, are buffered into memory and make up a command queue.
4334     Each frame therefore fills the command buffer with draw commands
4335     that then need to be executed by the user and his own render backend.
4336     After that the command buffer needs to be cleared and a new frame can be
4337     started. It is probably important to note that the command buffer is the main
4338     drawing API and the optional vertex buffer API only takes this format and
4339     converts it into a hardware accessible format.
4340
4341     To use the command queue to draw your own widgets you can access the
4342     command buffer of each window by calling `nk_window_get_canvas` after
4343     previously having called `nk_begin`:
4344
4345         void draw_red_rectangle_widget(struct nk_context *ctx)
4346         {
4347             struct nk_command_buffer *canvas;
4348             struct nk_input *input = &ctx->input;
4349             canvas = nk_window_get_canvas(ctx);
4350
4351             struct nk_rect space;
4352             enum nk_widget_layout_states state;
4353             state = nk_widget(&space, ctx);
4354             if (!state) return;
4355
4356             if (state != NK_WIDGET_ROM)
4357                 update_your_widget_by_user_input(...);
4358             nk_fill_rect(canvas, space, 0, nk_rgb(255,0,0));
4359         }
4360
4361         if (nk_begin(...)) {
4362             nk_layout_row_dynamic(ctx, 25, 1);
4363             draw_red_rectangle_widget(ctx);
4364         }
4365         nk_end(..)
4366
4367     Important to know if you want to create your own widgets is the `nk_widget`
4368     call. It allocates space on the panel reserved for this widget to be used,
4369     but also returns the state of the widget space. If your widget is not seen and does
4370     not have to be updated it is '0' and you can just return. If it only has
4371     to be drawn the state will be `NK_WIDGET_ROM` otherwise you can do both
4372     update and draw your widget. The reason for separating is to only draw and
4373     update what is actually necessary which is crucial for performance.
4374 */
4375 enum nk_command_type {
4376     NK_COMMAND_NOP,
4377     NK_COMMAND_SCISSOR,
4378     NK_COMMAND_LINE,
4379     NK_COMMAND_CURVE,
4380     NK_COMMAND_RECT,
4381     NK_COMMAND_RECT_FILLED,
4382     NK_COMMAND_RECT_MULTI_COLOR,
4383     NK_COMMAND_CIRCLE,
4384     NK_COMMAND_CIRCLE_FILLED,
4385     NK_COMMAND_ARC,
4386     NK_COMMAND_ARC_FILLED,
4387     NK_COMMAND_TRIANGLE,
4388     NK_COMMAND_TRIANGLE_FILLED,
4389     NK_COMMAND_POLYGON,
4390     NK_COMMAND_POLYGON_FILLED,
4391     NK_COMMAND_POLYLINE,
4392     NK_COMMAND_TEXT,
4393     NK_COMMAND_IMAGE,
4394     NK_COMMAND_CUSTOM
4395 };
```

```
4396
4397 /* command base and header of every command inside the buffer */
4398 struct nk_command {
4399     enum nk_command_type type;
4400     nk_size next;
4401 #ifdef NK_INCLUDE_COMMAND_USERDATA
4402     nk_handle userdata;
4403 #endif
4404 };
4405
4406 struct nk_command_scissor {
4407     struct nk_command header;
4408     short x, y;
4409     unsigned short w, h;
4410 };
4411
4412 struct nk_command_line {
4413     struct nk_command header;
4414     unsigned short line_thickness;
4415     struct nk_vec2i begin;
4416     struct nk_vec2i end;
4417     struct nk_color color;
4418 };
4419
4420 struct nk_command_curve {
4421     struct nk_command header;
4422     unsigned short line_thickness;
4423     struct nk_vec2i begin;
4424     struct nk_vec2i end;
4425     struct nk_vec2i ctrl[2];
4426     struct nk_color color;
4427 };
4428
4429 struct nk_command_rect {
4430     struct nk_command header;
4431     unsigned short rounding;
4432     unsigned short line_thickness;
4433     short x, y;
4434     unsigned short w, h;
4435     struct nk_color color;
4436 };
4437
4438 struct nk_command_rect_filled {
4439     struct nk_command header;
4440     unsigned short rounding;
4441     short x, y;
4442     unsigned short w, h;
4443     struct nk_color color;
4444 };
4445
4446 struct nk_command_rect_multi_color {
4447     struct nk_command header;
4448     short x, y;
4449     unsigned short w, h;
4450     struct nk_color left;
4451     struct nk_color top;
4452     struct nk_color bottom;
4453     struct nk_color right;
4454 };
4455
4456 struct nk_command_triangle {
4457     struct nk_command header;
4458     unsigned short line_thickness;
4459     struct nk_vec2i a;
4460     struct nk_vec2i b;
4461     struct nk_vec2i c;
4462     struct nk_color color;
4463 };
4464
4465 struct nk_command_triangle_filled {
4466     struct nk_command header;
4467     struct nk_vec2i a;
4468     struct nk_vec2i b;
4469     struct nk_vec2i c;
4470     struct nk_color color;
4471 };
4472
4473 struct nk_command_circle {
4474     struct nk_command header;
4475     short x, y;
4476     unsigned short line_thickness;
4477     unsigned short w, h;
4478     struct nk_color color;
4479 };
4480
4481 struct nk_command_circle_filled {
4482     struct nk_command header;
```

```
4483     short x, y;
4484     unsigned short w, h;
4485     struct nk_color color;
4486 };
4487
4488 struct nk_command_arc {
4489     struct nk_command header;
4490     short cx, cy;
4491     unsigned short r;
4492     unsigned short line_thickness;
4493     float a[2];
4494     struct nk_color color;
4495 };
4496
4497 struct nk_command_arc_filled {
4498     struct nk_command header;
4499     short cx, cy;
4500     unsigned short r;
4501     float a[2];
4502     struct nk_color color;
4503 };
4504
4505 struct nk_command_polygon {
4506     struct nk_command header;
4507     struct nk_color color;
4508     unsigned short line_thickness;
4509     unsigned short point_count;
4510     struct nk_vec2i points[1];
4511 };
4512
4513 struct nk_command_polygon_filled {
4514     struct nk_command header;
4515     struct nk_color color;
4516     unsigned short point_count;
4517     struct nk_vec2i points[1];
4518 };
4519
4520 struct nk_command_polyline {
4521     struct nk_command header;
4522     struct nk_color color;
4523     unsigned short line_thickness;
4524     unsigned short point_count;
4525     struct nk_vec2i points[1];
4526 };
4527
4528 struct nk_command_image {
4529     struct nk_command header;
4530     short x, y;
4531     unsigned short w, h;
4532     struct nk_image img;
4533     struct nk_color col;
4534 };
4535
4536 typedef void (*nk_command_custom_callback)(void *canvas, short x,short y,
4537     unsigned short w, unsigned short h, nk_handle callback_data);
4538 struct nk_command_custom {
4539     struct nk_command header;
4540     short x, y;
4541     unsigned short w, h;
4542     nk_handle callback_data;
4543     nk_command_custom_callback callback;
4544 };
4545
4546 struct nk_command_text {
4547     struct nk_command header;
4548     const struct nk_user_font *font;
4549     struct nk_color background;
4550     struct nk_color foreground;
4551     short x, y;
4552     unsigned short w, h;
4553     float height;
4554     int length;
4555     char string[1];
4556 };
4557
4558 enum nk_command_clipping {
4559     NK_CLIPPING_OFF = nk_false,
4560     NK_CLIPPING_ON = nk_true
4561 };
4562
4563 struct nk_command_buffer {
4564     struct nk_buffer *base;
4565     struct nk_rect clip;
4566     int use_clipping;
4567     nk_handle userdata;
4568     nk_size begin, end, last;
4569 };
```

```
4570
4571 /* shape outlines */
4572 NK_API void nk_stroke_line(struct nk_command_buffer *b, float x0, float y0, float x1, float y1, float
       line_thickness, struct nk_color);
4573 NK_API void nk_stroke_curve(struct nk_command_buffer*, float, float, float, float, float, float, float,
       float, float line_thickness, struct nk_color);
4574 NK_API void nk_stroke_rect(struct nk_command_buffer*, struct nk_rect, float rounding, float
       line_thickness, struct nk_color);
4575 NK_API void nk_stroke_circle(struct nk_command_buffer*, struct nk_rect, float line_thickness, struct
       nk_color);
4576 NK_API void nk_stroke_arc(struct nk_command_buffer*, float cx, float cy, float radius, float a_min,
       float a_max, float line_thickness, struct nk_color);
4577 NK_API void nk_stroke_triangle(struct nk_command_buffer*, float, float, float, float, float, float,
       float line_thichness, struct nk_color);
4578 NK_API void nk_stroke_polyline(struct nk_command_buffer*, float *points, int point_count, float
       line_thickness, struct nk_color col);
4579 NK_API void nk_stroke_polygon(struct nk_command_buffer*, float*, int point_count, float line_thickness,
       struct nk_color);
4580
4581 /* filled shades */
4582 NK_API void nk_fill_rect(struct nk_command_buffer*, struct nk_rect, float rounding, struct nk_color);
4583 NK_API void nk_fill_rect_multi_color(struct nk_command_buffer*, struct nk_rect, struct nk_color left,
       struct nk_color top, struct nk_color right, struct nk_color bottom);
4584 NK_API void nk_fill_circle(struct nk_command_buffer*, struct nk_rect, struct nk_color);
4585 NK_API void nk_fill_arc(struct nk_command_buffer*, float cx, float cy, float radius, float a_min, float
       a_max, struct nk_color);
4586 NK_API void nk_fill_triangle(struct nk_command_buffer*, float x0, float y0, float x1, float y1, float
       x2, float y2, struct nk_color);
4587 NK_API void nk_fill_polygon(struct nk_command_buffer*, float*, int point_count, struct nk_color);
4588
4589 /* misc */
4590 NK_API void nk_draw_image(struct nk_command_buffer*, struct nk_rect, const struct nk_image*, struct
       nk_color);
4591 NK_API void nk_draw_text(struct nk_command_buffer*, struct nk_rect, const char *text, int len, const
       struct nk_user_font*, struct nk_color, struct nk_color);
4592 NK_API void nk_push_scissor(struct nk_command_buffer*, struct nk_rect);
4593 NK_API void nk_push_custom(struct nk_command_buffer*, struct nk_rect, nk_command_custom_callback,
       nk_handle usr);
4594
4595 /* ===============================================================
4596  *
4597  *                          INPUT
4598  *
4599  * ===============================================================*/
4600 struct nk_mouse_button {
4601     int down;
4602     unsigned int clicked;
4603     struct nk_vec2 clicked_pos;
4604 };
4605 struct nk_mouse {
4606     struct nk_mouse_button buttons[NK_BUTTON_MAX];
4607     struct nk_vec2 pos;
4608     struct nk_vec2 prev;
4609     struct nk_vec2 delta;
4610     struct nk_vec2 scroll_delta;
4611     unsigned char grab;
4612     unsigned char grabbed;
4613     unsigned char ungrab;
4614 };
4615
4616 struct nk_key {
4617     int down;
4618     unsigned int clicked;
4619 };
4620 struct nk_keyboard {
4621     struct nk_key keys[NK_KEY_MAX];
4622     char text[NK_INPUT_MAX];
4623     int text_len;
4624 };
4625
4626 struct nk_input {
4627     struct nk_keyboard keyboard;
4628     struct nk_mouse mouse;
4629 };
4630
4631 NK_API int nk_input_has_mouse_click(const struct nk_input*, enum nk_buttons);
4632 NK_API int nk_input_has_mouse_click_in_rect(const struct nk_input*, enum nk_buttons, struct nk_rect);
4633 NK_API int nk_input_has_mouse_click_down_in_rect(const struct nk_input*, enum nk_buttons, struct
       nk_rect, int down);
4634 NK_API int nk_input_is_mouse_click_in_rect(const struct nk_input*, enum nk_buttons, struct nk_rect);
4635 NK_API int nk_input_is_mouse_click_down_in_rect(const struct nk_input *i, enum nk_buttons id, struct
       nk_rect b, int down);
4636 NK_API int nk_input_any_mouse_click_in_rect(const struct nk_input*, struct nk_rect);
4637 NK_API int nk_input_is_mouse_prev_hovering_rect(const struct nk_input*, struct nk_rect);
4638 NK_API int nk_input_is_mouse_hovering_rect(const struct nk_input*, struct nk_rect);
4639 NK_API int nk_input_mouse_clicked(const struct nk_input*, enum nk_buttons, struct nk_rect);
4640 NK_API int nk_input_is_mouse_down(const struct nk_input*, enum nk_buttons);
```

```
4641 NK_API int nk_input_is_mouse_pressed(const struct nk_input*, enum nk_buttons);
4642 NK_API int nk_input_is_mouse_released(const struct nk_input*, enum nk_buttons);
4643 NK_API int nk_input_is_key_pressed(const struct nk_input*, enum nk_keys);
4644 NK_API int nk_input_is_key_released(const struct nk_input*, enum nk_keys);
4645 NK_API int nk_input_is_key_down(const struct nk_input*, enum nk_keys);
4646
4647 /* ===============================================================
4648  *
4649  *                          DRAW LIST
4650  *
4651  * ===============================================================*/
4652 #ifdef NK_INCLUDE_VERTEX_BUFFER_OUTPUT
4653 /*  The optional vertex buffer draw list provides a 2D drawing context
4654     with antialiasing functionality which takes basic filled or outlined shapes
4655     or a path and outputs vertexes, elements and draw commands.
4656     The actual draw list API is not required to be used directly while using this
4657     library since converting the default library draw command output is done by
4658     just calling `nk_convert` but I decided to still make this library accessible
4659     since it can be useful.
4660
4661     The draw list is based on a path buffering and polygon and polyline
4662     rendering API which allows a lot of ways to draw 2D content to screen.
4663     In fact it is probably more powerful than needed but allows even more crazy
4664     things than this library provides by default.
4665 */
4666 #ifdef NK_UINT_DRAW_INDEX
4667 typedef nk_uint nk_draw_index;
4668 #else
4669 typedef nk_ushort nk_draw_index;
4670 #endif
4671 enum nk_draw_list_stroke {
4672     NK_STROKE_OPEN = nk_false,
4673     /* build up path has no connection back to the beginning */
4674     NK_STROKE_CLOSED = nk_true
4675     /* build up path has a connection back to the beginning */
4676 };
4677
4678 enum nk_draw_vertex_layout_attribute {
4679     NK_VERTEX_POSITION,
4680     NK_VERTEX_COLOR,
4681     NK_VERTEX_TEXCOORD,
4682     NK_VERTEX_ATTRIBUTE_COUNT
4683 };
4684
4685 enum nk_draw_vertex_layout_format {
4686     NK_FORMAT_SCHAR,
4687     NK_FORMAT_SSHORT,
4688     NK_FORMAT_SINT,
4689     NK_FORMAT_UCHAR,
4690     NK_FORMAT_USHORT,
4691     NK_FORMAT_UINT,
4692     NK_FORMAT_FLOAT,
4693     NK_FORMAT_DOUBLE,
4694
4695 NK_FORMAT_COLOR_BEGIN,
4696     NK_FORMAT_R8G8B8 = NK_FORMAT_COLOR_BEGIN,
4697     NK_FORMAT_R16G15B16,
4698     NK_FORMAT_R32G32B32,
4699
4700     NK_FORMAT_R8G8B8A8,
4701     NK_FORMAT_B8G8R8A8,
4702     NK_FORMAT_R16G15B16A16,
4703     NK_FORMAT_R32G32B32A32,
4704     NK_FORMAT_R32G32B32A32_FLOAT,
4705     NK_FORMAT_R32G32B32A32_DOUBLE,
4706
4707     NK_FORMAT_RGB32,
4708     NK_FORMAT_RGBA32,
4709 NK_FORMAT_COLOR_END = NK_FORMAT_RGBA32,
4710     NK_FORMAT_COUNT
4711 };
4712
4713 #define NK_VERTEX_LAYOUT_END NK_VERTEX_ATTRIBUTE_COUNT,NK_FORMAT_COUNT,0
4714 struct nk_draw_vertex_layout_element {
4715     enum nk_draw_vertex_layout_attribute attribute;
4716     enum nk_draw_vertex_layout_format format;
4717     nk_size offset;
4718 };
4719
4720 struct nk_draw_command {
4721     unsigned int elem_count;
4722     /* number of elements in the current draw batch */
4723     struct nk_rect clip_rect;
4724     /* current screen clipping rectangle */
4725     nk_handle texture;
4726     /* current texture to set */
4727 #ifdef NK_INCLUDE_COMMAND_USERDATA
```

```
4728     nk_handle userdata;
4729 #endif
4730 };
4731
4732 struct nk_draw_list {
4733     struct nk_rect clip_rect;
4734     struct nk_vec2 circle_vtx[12];
4735     struct nk_convert_config config;
4736
4737     struct nk_buffer *buffer;
4738     struct nk_buffer *vertices;
4739     struct nk_buffer *elements;
4740
4741     unsigned int element_count;
4742     unsigned int vertex_count;
4743     unsigned int cmd_count;
4744     nk_size cmd_offset;
4745
4746     unsigned int path_count;
4747     unsigned int path_offset;
4748
4749     enum nk_anti_aliasing line_AA;
4750     enum nk_anti_aliasing shape_AA;
4751
4752 #ifdef NK_INCLUDE_COMMAND_USERDATA
4753     nk_handle userdata;
4754 #endif
4755 };
4756
4757 /* draw list */
4758 NK_API void nk_draw_list_init(struct nk_draw_list*);
4759 NK_API void nk_draw_list_setup(struct nk_draw_list*, const struct nk_convert_config*, struct nk_buffer
      *cmds, struct nk_buffer *vertices, struct nk_buffer *elements, enum nk_anti_aliasing line_aa,enum
      nk_anti_aliasing shape_aa);
4760
4761 /* drawing */
4762 #define nk_draw_list_foreach(cmd, can, b) for((cmd)=nk__draw_list_begin(can, b); (cmd)!=0;
      (cmd)=nk__draw_list_next(cmd, b, can))
4763 NK_API const struct nk_draw_command* nk__draw_list_begin(const struct nk_draw_list*, const struct
      nk_buffer*);
4764 NK_API const struct nk_draw_command* nk__draw_list_next(const struct nk_draw_command*, const struct
      nk_buffer*, const struct nk_draw_list*);
4765 NK_API const struct nk_draw_command* nk__draw_list_end(const struct nk_draw_list*, const struct
      nk_buffer*);
4766
4767 /* path */
4768 NK_API void nk_draw_list_path_clear(struct nk_draw_list*);
4769 NK_API void nk_draw_list_path_line_to(struct nk_draw_list*, struct nk_vec2 pos);
4770 NK_API void nk_draw_list_path_arc_to_fast(struct nk_draw_list*, struct nk_vec2 center, float radius,
      int a_min, int a_max);
4771 NK_API void nk_draw_list_path_arc_to(struct nk_draw_list*, struct nk_vec2 center, float radius, float
      a_min, float a_max, unsigned int segments);
4772 NK_API void nk_draw_list_path_rect_to(struct nk_draw_list*, struct nk_vec2 a, struct nk_vec2 b, float
      rounding);
4773 NK_API void nk_draw_list_path_curve_to(struct nk_draw_list*, struct nk_vec2 p2, struct nk_vec2 p3,
      struct nk_vec2 p4, unsigned int num_segments);
4774 NK_API void nk_draw_list_path_fill(struct nk_draw_list*, struct nk_color);
4775 NK_API void nk_draw_list_path_stroke(struct nk_draw_list*, struct nk_color, enum nk_draw_list_stroke
      closed, float thickness);
4776
4777 /* stroke */
4778 NK_API void nk_draw_list_stroke_line(struct nk_draw_list*, struct nk_vec2 a, struct nk_vec2 b, struct
      nk_color, float thickness);
4779 NK_API void nk_draw_list_stroke_rect(struct nk_draw_list*, struct nk_rect rect, struct nk_color, float
      rounding, float thickness);
4780 NK_API void nk_draw_list_stroke_triangle(struct nk_draw_list*, struct nk_vec2 a, struct nk_vec2 b,
      struct nk_vec2 c, struct nk_color, float thickness);
4781 NK_API void nk_draw_list_stroke_circle(struct nk_draw_list*, struct nk_vec2 center, float radius,
      struct nk_color, unsigned int segs, float thickness);
4782 NK_API void nk_draw_list_stroke_curve(struct nk_draw_list*, struct nk_vec2 p0, struct nk_vec2 cp0,
      struct nk_vec2 cp1, struct nk_vec2 p1, struct nk_color, unsigned int segments, float thickness);
4783 NK_API void nk_draw_list_stroke_poly_line(struct nk_draw_list*, const struct nk_vec2 *pnts, const
      unsigned int cnt, struct nk_color, enum nk_draw_list_stroke, float thickness, enum nk_anti_aliasing);
4784
4785 /* fill */
4786 NK_API void nk_draw_list_fill_rect(struct nk_draw_list*, struct nk_rect rect, struct nk_color, float
      rounding);
4787 NK_API void nk_draw_list_fill_rect_multi_color(struct nk_draw_list*, struct nk_rect rect, struct
      nk_color left, struct nk_color top, struct nk_color right, struct nk_color bottom);
4788 NK_API void nk_draw_list_fill_triangle(struct nk_draw_list*, struct nk_vec2 a, struct nk_vec2 b, struct
      nk_vec2 c, struct nk_color);
4789 NK_API void nk_draw_list_fill_circle(struct nk_draw_list*, struct nk_vec2 center, float radius, struct
      nk_color col, unsigned int segs);
4790 NK_API void nk_draw_list_fill_poly_convex(struct nk_draw_list*, const struct nk_vec2 *points, const
      unsigned int count, struct nk_color, enum nk_anti_aliasing);
4791
4792 /* misc */
```

```
4793 NK_API void nk_draw_list_add_image(struct nk_draw_list*, struct nk_image texture, struct nk_rect rect,
       struct nk_color);
4794 NK_API void nk_draw_list_add_text(struct nk_draw_list*, const struct nk_user_font*, struct nk_rect,
       const char *text, int len, float font_height, struct nk_color);
4795 #ifdef NK_INCLUDE_COMMAND_USERDATA
4796 NK_API void nk_draw_list_push_userdata(struct nk_draw_list*, nk_handle userdata);
4797 #endif
4798
4799 #endif
4800
4801 /* ===============================================================
4802  *
4803  *                          GUI
4804  *
4805  * ===============================================================*/
4806 enum nk_style_item_type {
4807     NK_STYLE_ITEM_COLOR,
4808     NK_STYLE_ITEM_IMAGE
4809 };
4810
4811 union nk_style_item_data {
4812     struct nk_image image;
4813     struct nk_color color;
4814 };
4815
4816 struct nk_style_item {
4817     enum nk_style_item_type type;
4818     union nk_style_item_data data;
4819 };
4820
4821 struct nk_style_text {
4822     struct nk_color color;
4823     struct nk_vec2 padding;
4824 };
4825
4826 struct nk_style_button {
4827     /* background */
4828     struct nk_style_item normal;
4829     struct nk_style_item hover;
4830     struct nk_style_item active;
4831     struct nk_color border_color;
4832
4833     /* text */
4834     struct nk_color text_background;
4835     struct nk_color text_normal;
4836     struct nk_color text_hover;
4837     struct nk_color text_active;
4838     nk_flags text_alignment;
4839
4840     /* properties */
4841     float border;
4842     float rounding;
4843     struct nk_vec2 padding;
4844     struct nk_vec2 image_padding;
4845     struct nk_vec2 touch_padding;
4846
4847     /* optional user callbacks */
4848     nk_handle userdata;
4849     void(*draw_begin)(struct nk_command_buffer*, nk_handle userdata);
4850     void(*draw_end)(struct nk_command_buffer*, nk_handle userdata);
4851 };
4852
4853 struct nk_style_toggle {
4854     /* background */
4855     struct nk_style_item normal;
4856     struct nk_style_item hover;
4857     struct nk_style_item active;
4858     struct nk_color border_color;
4859
4860     /* cursor */
4861     struct nk_style_item cursor_normal;
4862     struct nk_style_item cursor_hover;
4863
4864     /* text */
4865     struct nk_color text_normal;
4866     struct nk_color text_hover;
4867     struct nk_color text_active;
4868     struct nk_color text_background;
4869     nk_flags text_alignment;
4870
4871     /* properties */
4872     struct nk_vec2 padding;
4873     struct nk_vec2 touch_padding;
4874     float spacing;
4875     float border;
4876
4877     /* optional user callbacks */
```

```
4878     nk_handle userdata;
4879     void(*draw_begin)(struct nk_command_buffer*, nk_handle);
4880     void(*draw_end)(struct nk_command_buffer*, nk_handle);
4881 };
4882
4883 struct nk_style_selectable {
4884     /* background (inactive) */
4885     struct nk_style_item normal;
4886     struct nk_style_item hover;
4887     struct nk_style_item pressed;
4888
4889     /* background (active) */
4890     struct nk_style_item normal_active;
4891     struct nk_style_item hover_active;
4892     struct nk_style_item pressed_active;
4893
4894     /* text color (inactive) */
4895     struct nk_color text_normal;
4896     struct nk_color text_hover;
4897     struct nk_color text_pressed;
4898
4899     /* text color (active) */
4900     struct nk_color text_normal_active;
4901     struct nk_color text_hover_active;
4902     struct nk_color text_pressed_active;
4903     struct nk_color text_background;
4904     nk_flags text_alignment;
4905
4906     /* properties */
4907     float rounding;
4908     struct nk_vec2 padding;
4909     struct nk_vec2 touch_padding;
4910     struct nk_vec2 image_padding;
4911
4912     /* optional user callbacks */
4913     nk_handle userdata;
4914     void(*draw_begin)(struct nk_command_buffer*, nk_handle);
4915     void(*draw_end)(struct nk_command_buffer*, nk_handle);
4916 };
4917
4918 struct nk_style_slider {
4919     /* background */
4920     struct nk_style_item normal;
4921     struct nk_style_item hover;
4922     struct nk_style_item active;
4923     struct nk_color border_color;
4924
4925     /* background bar */
4926     struct nk_color bar_normal;
4927     struct nk_color bar_hover;
4928     struct nk_color bar_active;
4929     struct nk_color bar_filled;
4930
4931     /* cursor */
4932     struct nk_style_item cursor_normal;
4933     struct nk_style_item cursor_hover;
4934     struct nk_style_item cursor_active;
4935
4936     /* properties */
4937     float border;
4938     float rounding;
4939     float bar_height;
4940     struct nk_vec2 padding;
4941     struct nk_vec2 spacing;
4942     struct nk_vec2 cursor_size;
4943
4944     /* optional buttons */
4945     int show_buttons;
4946     struct nk_style_button inc_button;
4947     struct nk_style_button dec_button;
4948     enum nk_symbol_type inc_symbol;
4949     enum nk_symbol_type dec_symbol;
4950
4951     /* optional user callbacks */
4952     nk_handle userdata;
4953     void(*draw_begin)(struct nk_command_buffer*, nk_handle);
4954     void(*draw_end)(struct nk_command_buffer*, nk_handle);
4955 };
4956
4957 struct nk_style_progress {
4958     /* background */
4959     struct nk_style_item normal;
4960     struct nk_style_item hover;
4961     struct nk_style_item active;
4962     struct nk_color border_color;
4963
4964     /* cursor */
```

```
4965     struct nk_style_item cursor_normal;
4966     struct nk_style_item cursor_hover;
4967     struct nk_style_item cursor_active;
4968     struct nk_color cursor_border_color;
4969
4970     /* properties */
4971     float rounding;
4972     float border;
4973     float cursor_border;
4974     float cursor_rounding;
4975     struct nk_vec2 padding;
4976
4977     /* optional user callbacks */
4978     nk_handle userdata;
4979     void(*draw_begin)(struct nk_command_buffer*, nk_handle);
4980     void(*draw_end)(struct nk_command_buffer*, nk_handle);
4981 };
4982
4983 struct nk_style_scrollbar {
4984     /* background */
4985     struct nk_style_item normal;
4986     struct nk_style_item hover;
4987     struct nk_style_item active;
4988     struct nk_color border_color;
4989
4990     /* cursor */
4991     struct nk_style_item cursor_normal;
4992     struct nk_style_item cursor_hover;
4993     struct nk_style_item cursor_active;
4994     struct nk_color cursor_border_color;
4995
4996     /* properties */
4997     float border;
4998     float rounding;
4999     float border_cursor;
5000     float rounding_cursor;
5001     struct nk_vec2 padding;
5002
5003     /* optional buttons */
5004     int show_buttons;
5005     struct nk_style_button inc_button;
5006     struct nk_style_button dec_button;
5007     enum nk_symbol_type inc_symbol;
5008     enum nk_symbol_type dec_symbol;
5009
5010     /* optional user callbacks */
5011     nk_handle userdata;
5012     void(*draw_begin)(struct nk_command_buffer*, nk_handle);
5013     void(*draw_end)(struct nk_command_buffer*, nk_handle);
5014 };
5015
5016 struct nk_style_edit {
5017     /* background */
5018     struct nk_style_item normal;
5019     struct nk_style_item hover;
5020     struct nk_style_item active;
5021     struct nk_color border_color;
5022     struct nk_style_scrollbar scrollbar;
5023
5024     /* cursor  */
5025     struct nk_color cursor_normal;
5026     struct nk_color cursor_hover;
5027     struct nk_color cursor_text_normal;
5028     struct nk_color cursor_text_hover;
5029
5030     /* text (unselected) */
5031     struct nk_color text_normal;
5032     struct nk_color text_hover;
5033     struct nk_color text_active;
5034
5035     /* text (selected) */
5036     struct nk_color selected_normal;
5037     struct nk_color selected_hover;
5038     struct nk_color selected_text_normal;
5039     struct nk_color selected_text_hover;
5040
5041     /* properties */
5042     float border;
5043     float rounding;
5044     float cursor_size;
5045     struct nk_vec2 scrollbar_size;
5046     struct nk_vec2 padding;
5047     float row_padding;
5048 };
5049
5050 struct nk_style_property {
5051     /* background */
```

```
5052     struct nk_style_item normal;
5053     struct nk_style_item hover;
5054     struct nk_style_item active;
5055     struct nk_color border_color;
5056
5057     /* text */
5058     struct nk_color label_normal;
5059     struct nk_color label_hover;
5060     struct nk_color label_active;
5061
5062     /* symbols */
5063     enum nk_symbol_type sym_left;
5064     enum nk_symbol_type sym_right;
5065
5066     /* properties */
5067     float border;
5068     float rounding;
5069     struct nk_vec2 padding;
5070
5071     struct nk_style_edit edit;
5072     struct nk_style_button inc_button;
5073     struct nk_style_button dec_button;
5074
5075     /* optional user callbacks */
5076     nk_handle userdata;
5077     void(*draw_begin)(struct nk_command_buffer*, nk_handle);
5078     void(*draw_end)(struct nk_command_buffer*, nk_handle);
5079 };
5080
5081 struct nk_style_chart {
5082     /* colors */
5083     struct nk_style_item background;
5084     struct nk_color border_color;
5085     struct nk_color selected_color;
5086     struct nk_color color;
5087
5088     /* properties */
5089     float border;
5090     float rounding;
5091     struct nk_vec2 padding;
5092 };
5093
5094 struct nk_style_combo {
5095     /* background */
5096     struct nk_style_item normal;
5097     struct nk_style_item hover;
5098     struct nk_style_item active;
5099     struct nk_color border_color;
5100
5101     /* label */
5102     struct nk_color label_normal;
5103     struct nk_color label_hover;
5104     struct nk_color label_active;
5105
5106     /* symbol */
5107     struct nk_color symbol_normal;
5108     struct nk_color symbol_hover;
5109     struct nk_color symbol_active;
5110
5111     /* button */
5112     struct nk_style_button button;
5113     enum nk_symbol_type sym_normal;
5114     enum nk_symbol_type sym_hover;
5115     enum nk_symbol_type sym_active;
5116
5117     /* properties */
5118     float border;
5119     float rounding;
5120     struct nk_vec2 content_padding;
5121     struct nk_vec2 button_padding;
5122     struct nk_vec2 spacing;
5123 };
5124
5125 struct nk_style_tab {
5126     /* background */
5127     struct nk_style_item background;
5128     struct nk_color border_color;
5129     struct nk_color text;
5130
5131     /* button */
5132     struct nk_style_button tab_maximize_button;
5133     struct nk_style_button tab_minimize_button;
5134     struct nk_style_button node_maximize_button;
5135     struct nk_style_button node_minimize_button;
5136     enum nk_symbol_type sym_minimize;
5137     enum nk_symbol_type sym_maximize;
5138
```

```
5139     /* properties */
5140     float border;
5141     float rounding;
5142     float indent;
5143     struct nk_vec2 padding;
5144     struct nk_vec2 spacing;
5145 };
5146
5147 enum nk_style_header_align {
5148     NK_HEADER_LEFT,
5149     NK_HEADER_RIGHT
5150 };
5151 struct nk_style_window_header {
5152     /* background */
5153     struct nk_style_item normal;
5154     struct nk_style_item hover;
5155     struct nk_style_item active;
5156
5157     /* button */
5158     struct nk_style_button close_button;
5159     struct nk_style_button minimize_button;
5160     enum nk_symbol_type close_symbol;
5161     enum nk_symbol_type minimize_symbol;
5162     enum nk_symbol_type maximize_symbol;
5163
5164     /* title */
5165     struct nk_color label_normal;
5166     struct nk_color label_hover;
5167     struct nk_color label_active;
5168
5169     /* properties */
5170     enum nk_style_header_align align;
5171     struct nk_vec2 padding;
5172     struct nk_vec2 label_padding;
5173     struct nk_vec2 spacing;
5174 };
5175
5176 struct nk_style_window {
5177     struct nk_style_window_header header;
5178     struct nk_style_item fixed_background;
5179     struct nk_color background;
5180
5181     struct nk_color border_color;
5182     struct nk_color popup_border_color;
5183     struct nk_color combo_border_color;
5184     struct nk_color contextual_border_color;
5185     struct nk_color menu_border_color;
5186     struct nk_color group_border_color;
5187     struct nk_color tooltip_border_color;
5188     struct nk_style_item scaler;
5189
5190     float border;
5191     float combo_border;
5192     float contextual_border;
5193     float menu_border;
5194     float group_border;
5195     float tooltip_border;
5196     float popup_border;
5197     float min_row_height_padding;
5198
5199     float rounding;
5200     struct nk_vec2 spacing;
5201     struct nk_vec2 scrollbar_size;
5202     struct nk_vec2 min_size;
5203
5204     struct nk_vec2 padding;
5205     struct nk_vec2 group_padding;
5206     struct nk_vec2 popup_padding;
5207     struct nk_vec2 combo_padding;
5208     struct nk_vec2 contextual_padding;
5209     struct nk_vec2 menu_padding;
5210     struct nk_vec2 tooltip_padding;
5211 };
5212
5213 struct nk_style {
5214     const struct nk_user_font *font;
5215     const struct nk_cursor *cursors[NK_CURSOR_COUNT];
5216     const struct nk_cursor *cursor_active;
5217     struct nk_cursor *cursor_last;
5218     int cursor_visible;
5219
5220     struct nk_style_text text;
5221     struct nk_style_button button;
5222     struct nk_style_button contextual_button;
5223     struct nk_style_button menu_button;
5224     struct nk_style_toggle option;
5225     struct nk_style_toggle checkbox;
```

```
5226     struct nk_style_selectable selectable;
5227     struct nk_style_slider slider;
5228     struct nk_style_progress progress;
5229     struct nk_style_property property;
5230     struct nk_style_edit edit;
5231     struct nk_style_chart chart;
5232     struct nk_style_scrollbar scrollh;
5233     struct nk_style_scrollbar scrollv;
5234     struct nk_style_tab tab;
5235     struct nk_style_combo combo;
5236     struct nk_style_window window;
5237 };
5238
5239 NK_API struct nk_style_item nk_style_item_image(struct nk_image img);
5240 NK_API struct nk_style_item nk_style_item_color(struct nk_color);
5241 NK_API struct nk_style_item nk_style_item_hide(void);
5242
5243 /*==============================================================
5244  *                          PANEL
5245  * ==============================================================*/
5246 #ifndef NK_MAX_LAYOUT_ROW_TEMPLATE_COLUMNS
5247 #define NK_MAX_LAYOUT_ROW_TEMPLATE_COLUMNS 16
5248 #endif
5249 #ifndef NK_CHART_MAX_SLOT
5250 #define NK_CHART_MAX_SLOT 4
5251 #endif
5252
5253 enum nk_panel_type {
5254     NK_PANEL_NONE       = 0,
5255     NK_PANEL_WINDOW     = NK_FLAG(0),
5256     NK_PANEL_GROUP      = NK_FLAG(1),
5257     NK_PANEL_POPUP      = NK_FLAG(2),
5258     NK_PANEL_CONTEXTUAL = NK_FLAG(4),
5259     NK_PANEL_COMBO      = NK_FLAG(5),
5260     NK_PANEL_MENU       = NK_FLAG(6),
5261     NK_PANEL_TOOLTIP    = NK_FLAG(7)
5262 };
5263 enum nk_panel_set {
5264     NK_PANEL_SET_NONBLOCK = NK_PANEL_CONTEXTUAL|NK_PANEL_COMBO|NK_PANEL_MENU|NK_PANEL_TOOLTIP,
5265     NK_PANEL_SET_POPUP = NK_PANEL_SET_NONBLOCK|NK_PANEL_POPUP,
5266     NK_PANEL_SET_SUB = NK_PANEL_SET_POPUP|NK_PANEL_GROUP
5267 };
5268
5269 struct nk_chart_slot {
5270     enum nk_chart_type type;
5271     struct nk_color color;
5272     struct nk_color highlight;
5273     float min, max, range;
5274     int count;
5275     struct nk_vec2 last;
5276     int index;
5277 };
5278
5279 struct nk_chart {
5280     int slot;
5281     float x, y, w, h;
5282     struct nk_chart_slot slots[NK_CHART_MAX_SLOT];
5283 };
5284
5285 enum nk_panel_row_layout_type {
5286     NK_LAYOUT_DYNAMIC_FIXED = 0,
5287     NK_LAYOUT_DYNAMIC_ROW,
5288     NK_LAYOUT_DYNAMIC_FREE,
5289     NK_LAYOUT_DYNAMIC,
5290     NK_LAYOUT_STATIC_FIXED,
5291     NK_LAYOUT_STATIC_ROW,
5292     NK_LAYOUT_STATIC_FREE,
5293     NK_LAYOUT_STATIC,
5294     NK_LAYOUT_TEMPLATE,
5295     NK_LAYOUT_COUNT
5296 };
5297 struct nk_row_layout {
5298     enum nk_panel_row_layout_type type;
5299     int index;
5300     float height;
5301     float min_height;
5302     int columns;
5303     const float *ratio;
5304     float item_width;
5305     float item_height;
5306     float item_offset;
5307     float filled;
5308     struct nk_rect item;
5309     int tree_depth;
5310     float templates[NK_MAX_LAYOUT_ROW_TEMPLATE_COLUMNS];
5311 };
5312
```

```
5313 struct nk_popup_buffer {
5314     nk_size begin;
5315     nk_size parent;
5316     nk_size last;
5317     nk_size end;
5318     int active;
5319 };
5320
5321 struct nk_menu_state {
5322     float x, y, w, h;
5323     struct nk_scroll offset;
5324 };
5325
5326 struct nk_panel {
5327     enum nk_panel_type type;
5328     nk_flags flags;
5329     struct nk_rect bounds;
5330     nk_uint *offset_x;
5331     nk_uint *offset_y;
5332     float at_x, at_y, max_x;
5333     float footer_height;
5334     float header_height;
5335     float border;
5336     unsigned int has_scrolling;
5337     struct nk_rect clip;
5338     struct nk_menu_state menu;
5339     struct nk_row_layout row;
5340     struct nk_chart chart;
5341     struct nk_command_buffer *buffer;
5342     struct nk_panel *parent;
5343 };
5344
5345 /*==============================================================
5346  *                          WINDOW
5347  * ==============================================================*/
5348 #ifndef NK_WINDOW_MAX_NAME
5349 #define NK_WINDOW_MAX_NAME 64
5350 #endif
5351
5352 struct nk_table;
5353 enum nk_window_flags {
5354     NK_WINDOW_PRIVATE       = NK_FLAG(11),
5355     NK_WINDOW_DYNAMIC       = NK_WINDOW_PRIVATE,
5356     /* special window type growing up in height while being filled to a certain maximum height */
5357     NK_WINDOW_ROM           = NK_FLAG(12),
5358     /* sets window widgets into a read only mode and does not allow input changes */
5359     NK_WINDOW_NOT_INTERACTIVE = NK_WINDOW_ROM|NK_WINDOW_NO_INPUT,
5360     /* prevents all interaction caused by input to either window or widgets inside */
5361     NK_WINDOW_HIDDEN        = NK_FLAG(13),
5362     /* Hides window and stops any window interaction and drawing */
5363     NK_WINDOW_CLOSED        = NK_FLAG(14),
5364     /* Directly closes and frees the window at the end of the frame */
5365     NK_WINDOW_MINIMIZED     = NK_FLAG(15),
5366     /* marks the window as minimized */
5367     NK_WINDOW_REMOVE_ROM    = NK_FLAG(16)
5368     /* Removes read only mode at the end of the window */
5369 };
5370
5371 struct nk_popup_state {
5372     struct nk_window *win;
5373     enum nk_panel_type type;
5374     struct nk_popup_buffer buf;
5375     nk_hash name;
5376     int active;
5377     unsigned combo_count;
5378     unsigned con_count, con_old;
5379     unsigned active_con;
5380     struct nk_rect header;
5381 };
5382
5383 struct nk_edit_state {
5384     nk_hash name;
5385     unsigned int seq;
5386     unsigned int old;
5387     int active, prev;
5388     int cursor;
5389     int sel_start;
5390     int sel_end;
5391     struct nk_scroll scrollbar;
5392     unsigned char mode;
5393     unsigned char single_line;
5394 };
5395
5396 struct nk_property_state {
5397     int active, prev;
5398     char buffer[NK_MAX_NUMBER_BUFFER];
5399     int length;
```

```
5400     int cursor;
5401     int select_start;
5402     int select_end;
5403     nk_hash name;
5404     unsigned int seq;
5405     unsigned int old;
5406     int state;
5407 };
5408
5409 struct nk_window {
5410     unsigned int seq;
5411     nk_hash name;
5412     char name_string[NK_WINDOW_MAX_NAME];
5413     nk_flags flags;
5414
5415     struct nk_rect bounds;
5416     struct nk_scroll scrollbar;
5417     struct nk_command_buffer buffer;
5418     struct nk_panel *layout;
5419     float scrollbar_hiding_timer;
5420
5421     /* persistent widget state */
5422     struct nk_property_state property;
5423     struct nk_popup_state popup;
5424     struct nk_edit_state edit;
5425     unsigned int scrolled;
5426
5427     struct nk_table *tables;
5428     unsigned int table_count;
5429
5430     /* window list hooks */
5431     struct nk_window *next;
5432     struct nk_window *prev;
5433     struct nk_window *parent;
5434 };
5435
5436 /*==============================================================
5437  *                          STACK
5438  * ==============================================================*/
5439 /* The style modifier stack can be used to temporarily change a
5440  * property inside `nk_style`. For example if you want a special
5441  * red button you can temporarily push the old button color onto a stack
5442  * draw the button with a red color and then you just pop the old color
5443  * back from the stack:
5444  *
5445  *      nk_style_push_style_item(ctx, &ctx->style.button.normal, nk_style_item_color(nk_rgb(255,0,0)));
5446  *      nk_style_push_style_item(ctx, &ctx->style.button.hover, nk_style_item_color(nk_rgb(255,0,0)));
5447  *      nk_style_push_style_item(ctx, &ctx->style.button.active, nk_style_item_color(nk_rgb(255,0,0)));
5448  *      nk_style_push_vec2(ctx, &cx->style.button.padding, nk_vec2(2,2));
5449  *
5450  *      nk_button(...);
5451  *
5452  *      nk_style_pop_style_item(ctx);
5453  *      nk_style_pop_style_item(ctx);
5454  *      nk_style_pop_style_item(ctx);
5455  *      nk_style_pop_vec2(ctx);
5456  *
5457  * Nuklear has a stack for style_items, float properties, vector properties,
5458  * flags, colors, fonts and for button_behavior. Each has it's own fixed size stack
5459  * which can be changed at compile time.
5460  */
5461 #ifndef NK_BUTTON_BEHAVIOR_STACK_SIZE
5462 #define NK_BUTTON_BEHAVIOR_STACK_SIZE 8
5463 #endif
5464
5465 #ifndef NK_FONT_STACK_SIZE
5466 #define NK_FONT_STACK_SIZE 8
5467 #endif
5468
5469 #ifndef NK_STYLE_ITEM_STACK_SIZE
5470 #define NK_STYLE_ITEM_STACK_SIZE 16
5471 #endif
5472
5473 #ifndef NK_FLOAT_STACK_SIZE
5474 #define NK_FLOAT_STACK_SIZE 32
5475 #endif
5476
5477 #ifndef NK_VECTOR_STACK_SIZE
5478 #define NK_VECTOR_STACK_SIZE 16
5479 #endif
5480
5481 #ifndef NK_FLAGS_STACK_SIZE
5482 #define NK_FLAGS_STACK_SIZE 32
5483 #endif
5484
5485 #ifndef NK_COLOR_STACK_SIZE
5486 #define NK_COLOR_STACK_SIZE 32
```

```
5487 #endif
5488
5489 #define NK_CONFIGURATION_STACK_TYPE(prefix, name, type)\
5490     struct nk_config_stack_##name##_element {\
5491         prefix##_##type *address;\
5492         prefix##_##type old_value;\
5493     }
5494 #define NK_CONFIG_STACK(type,size)\
5495     struct nk_config_stack_##type {\
5496         int head;\
5497         struct nk_config_stack_##type##_element elements[size];\
5498     }
5499
5500 #define nk_float float
5501 NK_CONFIGURATION_STACK_TYPE(struct nk, style_item, style_item);
5502 NK_CONFIGURATION_STACK_TYPE(nk ,float, float);
5503 NK_CONFIGURATION_STACK_TYPE(struct nk, vec2, vec2);
5504 NK_CONFIGURATION_STACK_TYPE(nk ,flags, flags);
5505 NK_CONFIGURATION_STACK_TYPE(struct nk, color, color);
5506 NK_CONFIGURATION_STACK_TYPE(const struct nk, user_font, user_font*);
5507 NK_CONFIGURATION_STACK_TYPE(enum nk, button_behavior, button_behavior);
5508
5509 NK_CONFIG_STACK(style_item, NK_STYLE_ITEM_STACK_SIZE);
5510 NK_CONFIG_STACK(float, NK_FLOAT_STACK_SIZE);
5511 NK_CONFIG_STACK(vec2, NK_VECTOR_STACK_SIZE);
5512 NK_CONFIG_STACK(flags, NK_FLAGS_STACK_SIZE);
5513 NK_CONFIG_STACK(color, NK_COLOR_STACK_SIZE);
5514 NK_CONFIG_STACK(user_font, NK_FONT_STACK_SIZE);
5515 NK_CONFIG_STACK(button_behavior, NK_BUTTON_BEHAVIOR_STACK_SIZE);
5516
5517 struct nk_configuration_stacks {
5518     struct nk_config_stack_style_item style_items;
5519     struct nk_config_stack_float floats;
5520     struct nk_config_stack_vec2 vectors;
5521     struct nk_config_stack_flags flags;
5522     struct nk_config_stack_color colors;
5523     struct nk_config_stack_user_font fonts;
5524     struct nk_config_stack_button_behavior button_behaviors;
5525 };
5526
5527 /*==============================================================
5528  *                          CONTEXT
5529  * =============================================================*/
5530 #define NK_VALUE_PAGE_CAPACITY \
5531     (((NK_MAX(sizeof(struct nk_window),sizeof(struct nk_panel)) / sizeof(nk_uint))) / 2)
5532
5533 struct nk_table {
5534     unsigned int seq;
5535     unsigned int size;
5536     nk_hash keys[NK_VALUE_PAGE_CAPACITY];
5537     nk_uint values[NK_VALUE_PAGE_CAPACITY];
5538     struct nk_table *next, *prev;
5539 };
5540
5541 union nk_page_data {
5542     struct nk_table tbl;
5543     struct nk_panel pan;
5544     struct nk_window win;
5545 };
5546
5547 struct nk_page_element {
5548     union nk_page_data data;
5549     struct nk_page_element *next;
5550     struct nk_page_element *prev;
5551 };
5552
5553 struct nk_page {
5554     unsigned int size;
5555     struct nk_page *next;
5556     struct nk_page_element win[1];
5557 };
5558
5559 struct nk_pool {
5560     struct nk_allocator alloc;
5561     enum nk_allocation_type type;
5562     unsigned int page_count;
5563     struct nk_page *pages;
5564     struct nk_page_element *freelist;
5565     unsigned capacity;
5566     nk_size size;
5567     nk_size cap;
5568 };
5569
5570 struct nk_context {
5571 /* public: can be accessed freely */
5572     struct nk_input input;
5573     struct nk_style style;
```

```
5574     struct nk_buffer memory;
5575     struct nk_clipboard clip;
5576     nk_flags last_widget_state;
5577     enum nk_button_behavior button_behavior;
5578     struct nk_configuration_stacks stacks;
5579     float delta_time_seconds;
5580
5581 /* private:
5582     should only be accessed if you
5583     know what you are doing */
5584 #ifdef NK_INCLUDE_VERTEX_BUFFER_OUTPUT
5585     struct nk_draw_list draw_list;
5586 #endif
5587 #ifdef NK_INCLUDE_COMMAND_USERDATA
5588     nk_handle userdata;
5589 #endif
5590     /* text editor objects are quite big because of an internal
5591      * undo/redo stack. Therefore it does not make sense to have one for
5592      * each window for temporary use cases, so I only provide *one* instance
5593      * for all windows. This works because the content is cleared anyway */
5594     struct nk_text_edit text_edit;
5595     /* draw buffer used for overlay drawing operation like cursor */
5596     struct nk_command_buffer overlay;
5597
5598     /* windows */
5599     int build;
5600     int use_pool;
5601     struct nk_pool pool;
5602     struct nk_window *begin;
5603     struct nk_window *end;
5604     struct nk_window *active;
5605     struct nk_window *current;
5606     struct nk_page_element *freelist;
5607     unsigned int count;
5608     unsigned int seq;
5609 };
5610
5611 /* ==============================================================
5612  *                          MATH
5613  * ============================================================== */
5614 #define NK_PI 3.141592654f
5615 #define NK_UTF_INVALID 0xFFFD
5616 #define NK_MAX_FLOAT_PRECISION 2
5617
5618 #define NK_UNUSED(x) ((void)(x))
5619 #define NK_SATURATE(x) (NK_MAX(0, NK_MIN(1.0f, x)))
5620 #define NK_LEN(a) (sizeof(a)/sizeof(a)[0])
5621 #define NK_ABS(a) (((a) < 0) ? -(a) : (a))
5622 #define NK_BETWEEN(x, a, b) ((a) <= (x) && (x) < (b))
5623 #define NK_INBOX(px, py, x, y, w, h)\
5624     (NK_BETWEEN(px,x,x+w) && NK_BETWEEN(py,y,y+h))
5625 #define NK_INTERSECT(x0, y0, w0, h0, x1, y1, w1, h1) \
5626     (!(((x1 > (x0 + w0)) || ((x1 + w1) < x0) || (y1 > (y0 + h0)) || (y1 + h1) < y0)))
5627 #define NK_CONTAINS(x, y, w, h, bx, by, bw, bh)\
5628     (NK_INBOX(x,y, bx, by, bw, bh) && NK_INBOX(x+w,y+h, bx, by, bw, bh))
5629
5630 #define nk_vec2_sub(a, b) nk_vec2((a).x - (b).x, (a).y - (b).y)
5631 #define nk_vec2_add(a, b) nk_vec2((a).x + (b).x, (a).y + (b).y)
5632 #define nk_vec2_len_sqr(a) ((a).x*(a).x+(a).y*(a).y)
5633 #define nk_vec2_muls(a, t) nk_vec2((a).x * (t), (a).y * (t))
5634
5635 #define nk_ptr_add(t, p, i) ((t*)((void*)((nk_byte*)(p) + (i))))
5636 #define nk_ptr_add_const(t, p, i) ((const t*)((const void*)((const nk_byte*)(p) + (i))))
5637 #define nk_zero_struct(s) nk_zero(&s, sizeof(s))
5638
5639 /* ==============================================================
5640  *                          ALIGNMENT
5641  * ============================================================== */
5642 /* Pointer to Integer type conversion for pointer alignment */
5643 #if defined(__PTRDIFF_TYPE__) /* This case should work for GCC*/
5644 # define NK_UINT_TO_PTR(x) ((void*)(__PTRDIFF_TYPE__)(x))
5645 # define NK_PTR_TO_UINT(x) ((nk_size)(__PTRDIFF_TYPE__)(x))
5646 #elif !defined(__GNUC__) /* works for compilers other than LLVM */
5647 # define NK_UINT_TO_PTR(x) ((void*)&((char*)0)[x])
5648 # define NK_PTR_TO_UINT(x) ((nk_size)(((char*)x)-(char*)0))
5649 #elif defined(NK_USE_FIXED_TYPES) /* used if we have <stdint.h> */
5650 # define NK_UINT_TO_PTR(x) ((void*)(uintptr_t)(x))
5651 # define NK_PTR_TO_UINT(x) ((uintptr_t)(x))
5652 #else /* generates warning but works */
5653 # define NK_UINT_TO_PTR(x) ((void*)(x))
5654 # define NK_PTR_TO_UINT(x) ((nk_size)(x))
5655 #endif
5656
5657 #define NK_ALIGN_PTR(x, mask)\
5658     (NK_UINT_TO_PTR((NK_PTR_TO_UINT((nk_byte*)(x) + (mask-1)) & ~(mask-1))))
5659 #define NK_ALIGN_PTR_BACK(x, mask)\
5660     (NK_UINT_TO_PTR((NK_PTR_TO_UINT((nk_byte*)(x)) & ~(mask-1))))
```

```
5661
5662 #define NK_OFFSETOF(st,m) ((nk_ptr)&(((st*)0)->m))
5663 #define NK_CONTAINER_OF(ptr,type,member)\
5664     (type*)((void*)((char*)(1 ? (ptr): &((type*)0)->member) - NK_OFFSETOF(type, member)))
5665
5666 #ifdef __cplusplus
5667 }
5668 #endif
5669
5670 #ifdef __cplusplus
5671 template<typename T> struct nk_alignof;
5672 template<typename T, int size_diff> struct nk_helper{enum {value = size_diff};};
5673 template<typename T> struct nk_helper<T,0>{enum {value = nk_alignof<T>::value};};
5674 template<typename T> struct nk_alignof{struct Big {T x; char c;}; enum {
5675     diff = sizeof(Big) - sizeof(T), value = nk_helper<Big, diff>::value};};
5676 #define NK_ALIGNOF(t) (nk_alignof<t>::value)
5677 #elif defined(_MSC_VER)
5678 #define NK_ALIGNOF(t) (__alignof(t))
5679 #else
5680 #define NK_ALIGNOF(t) ((char*)(&((struct {char c; t _h;}*)0)->_h) - (char*)0)
5681 #endif
5682
5683 #endif /* NK_NUKLEAR_H_ */
5684
5685 #ifdef NK_IMPLEMENTATION
5686
5687 #ifndef NK_INTERNAL_H
5688 #define NK_INTERNAL_H
5689
5690 #ifndef NK_POOL_DEFAULT_CAPACITY
5691 #define NK_POOL_DEFAULT_CAPACITY 16
5692 #endif
5693
5694 #ifndef NK_DEFAULT_COMMAND_BUFFER_SIZE
5695 #define NK_DEFAULT_COMMAND_BUFFER_SIZE (4*1024)
5696 #endif
5697
5698 #ifndef NK_BUFFER_DEFAULT_INITIAL_SIZE
5699 #define NK_BUFFER_DEFAULT_INITIAL_SIZE (4*1024)
5700 #endif
5701
5702 /* standard library headers */
5703 #ifdef NK_INCLUDE_DEFAULT_ALLOCATOR
5704 #include <stdlib.h> /* malloc, free */
5705 #endif
5706 #ifdef NK_INCLUDE_STANDARD_IO
5707 #include <stdio.h> /* fopen, fclose,... */
5708 #endif
5709 #ifndef NK_ASSERT
5710 #include <assert.h>
5711 #define NK_ASSERT(expr) assert(expr)
5712 #endif
5713
5714 #ifndef NK_MEMSET
5715 #define NK_MEMSET nk_memset
5716 #endif
5717 #ifndef NK_MEMCPY
5718 #define NK_MEMCPY nk_memcopy
5719 #endif
5720 #ifndef NK_SQRT
5721 #define NK_SQRT nk_sqrt
5722 #endif
5723 #ifndef NK_SIN
5724 #define NK_SIN nk_sin
5725 #endif
5726 #ifndef NK_COS
5727 #define NK_COS nk_cos
5728 #endif
5729 #ifndef NK_STRTOD
5730 #define NK_STRTOD nk_strtod
5731 #endif
5732 #ifndef NK_DTOA
5733 #define NK_DTOA nk_dtoa
5734 #endif
5735
5736 #define NK_DEFAULT (-1)
5737
5738 #ifndef NK_VSNPRINTF
5739 /* If your compiler does support `vsnprintf` I would highly recommend
5740  * defining this to vsnprintf instead since `vsprintf` is basically
5741  * unbelievable unsafe and should *NEVER* be used. But I have to support
5742  * it since C89 only provides this unsafe version. */
5743   #if (defined(__STDC_VERSION__) && (__STDC_VERSION__ >= 199901L)) ||\
5744       (defined(__cplusplus) && (__cplusplus >= 201103L)) || \
5745       (defined(_POSIX_C_SOURCE) && (_POSIX_C_SOURCE >= 200112L)) ||\
5746       (defined(_XOPEN_SOURCE) && (_XOPEN_SOURCE >= 500)) ||\
5747        defined(_ISOC99_SOURCE) || defined(_BSD_SOURCE)
```

```
5748        #define NK_VSNPRINTF(s,n,f,a) vsnprintf(s,n,f,a)
5749    #else
5750      #define NK_VSNPRINTF(s,n,f,a) vsprintf(s,f,a)
5751    #endif
5752 #endif
5753
5754 #define NK_SCHAR_MIN (-127)
5755 #define NK_SCHAR_MAX 127
5756 #define NK_UCHAR_MIN 0
5757 #define NK_UCHAR_MAX 256
5758 #define NK_SSHORT_MIN (-32767)
5759 #define NK_SSHORT_MAX 32767
5760 #define NK_USHORT_MIN 0
5761 #define NK_USHORT_MAX 65535
5762 #define NK_SINT_MIN (-2147483647)
5763 #define NK_SINT_MAX 2147483647
5764 #define NK_UINT_MIN 0
5765 #define NK_UINT_MAX 4294967295u
5766
5767 /* Make sure correct type size:
5768  * This will fire with a negative subscript error if the type sizes
5769  * are set incorrectly by the compiler, and compile out if not */
5770 NK_STATIC_ASSERT(sizeof(nk_size) >= sizeof(void*));
5771 NK_STATIC_ASSERT(sizeof(nk_ptr) == sizeof(void*));
5772 NK_STATIC_ASSERT(sizeof(nk_flags) >= 4);
5773 NK_STATIC_ASSERT(sizeof(nk_rune) >= 4);
5774 NK_STATIC_ASSERT(sizeof(nk_ushort) == 2);
5775 NK_STATIC_ASSERT(sizeof(nk_short) == 2);
5776 NK_STATIC_ASSERT(sizeof(nk_uint) == 4);
5777 NK_STATIC_ASSERT(sizeof(nk_int) == 4);
5778 NK_STATIC_ASSERT(sizeof(nk_byte) == 1);
5779
5780 NK_GLOBAL const struct nk_rect nk_null_rect = {-8192.0f, -8192.0f, 16384, 16384};
5781 #define NK_FLOAT_PRECISION 0.00000000000001
5782
5783 NK_GLOBAL const struct nk_color nk_red = {255,0,0,255};
5784 NK_GLOBAL const struct nk_color nk_green = {0,255,0,255};
5785 NK_GLOBAL const struct nk_color nk_blue = {0,0,255,255};
5786 NK_GLOBAL const struct nk_color nk_white = {255,255,255,255};
5787 NK_GLOBAL const struct nk_color nk_black = {0,0,0,255};
5788 NK_GLOBAL const struct nk_color nk_yellow = {255,255,0,255};
5789
5790 /* widget */
5791 #define nk_widget_state_reset(s)\
5792     if ((*(s)) & NK_WIDGET_STATE_MODIFIED)\
5793         (*(s)) = NK_WIDGET_STATE_INACTIVE|NK_WIDGET_STATE_MODIFIED;\
5794     else (*(s)) = NK_WIDGET_STATE_INACTIVE;
5795
5796 /* math */
5797 NK_LIB float nk_inv_sqrt(float n);
5798 NK_LIB float nk_sqrt(float x);
5799 NK_LIB float nk_sin(float x);
5800 NK_LIB float nk_cos(float x);
5801 NK_LIB nk_uint nk_round_up_pow2(nk_uint v);
5802 NK_LIB struct nk_rect nk_shrink_rect(struct nk_rect r, float amount);
5803 NK_LIB struct nk_rect nk_pad_rect(struct nk_rect r, struct nk_vec2 pad);
5804 NK_LIB void nk_unify(struct nk_rect *clip, const struct nk_rect *a, float x0, float y0, float x1, float
       y1);
5805 NK_LIB double nk_pow(double x, int n);
5806 NK_LIB int nk_ifloord(double x);
5807 NK_LIB int nk_ifloorf(float x);
5808 NK_LIB int nk_iceilf(float x);
5809 NK_LIB int nk_log10(double n);
5810
5811 /* util */
5812 enum {NK_DO_NOT_STOP_ON_NEW_LINE, NK_STOP_ON_NEW_LINE};
5813 NK_LIB int nk_is_lower(int c);
5814 NK_LIB int nk_is_upper(int c);
5815 NK_LIB int nk_to_upper(int c);
5816 NK_LIB int nk_to_lower(int c);
5817 NK_LIB void* nk_memcopy(void *dst, const void *src, nk_size n);
5818 NK_LIB void nk_memset(void *ptr, int c0, nk_size size);
5819 NK_LIB void nk_zero(void *ptr, nk_size size);
5820 NK_LIB char *nk_itoa(char *s, long n);
5821 NK_LIB int nk_string_float_limit(char *string, int prec);
5822 NK_LIB char *nk_dtoa(char *s, double n);
5823 NK_LIB int nk_text_clamp(const struct nk_user_font *font, const char *text, int text_len, float space,
       int *glyphs, float *text_width, nk_rune *sep_list, int sep_count);
5824 NK_LIB struct nk_vec2 nk_text_calculate_text_bounds(const struct nk_user_font *font, const char *begin,
       int byte_len, float row_height, const char **remaining, struct nk_vec2 *out_offset, int *glyphs, int
       op);
5825 #ifdef NK_INCLUDE_STANDARD_VARARGS
5826 NK_LIB int nk_strfmt(char *buf, int buf_size, const char *fmt, va_list args);
5827 #endif
5828 #ifdef NK_INCLUDE_STANDARD_IO
5829 NK_LIB char *nk_file_load(const char* path, nk_size* siz, struct nk_allocator *alloc);
5830 #endif
```

```
5831
5832 /* buffer */
5833 #ifdef NK_INCLUDE_DEFAULT_ALLOCATOR
5834 NK_LIB void* nk_malloc(nk_handle unused, void *old,nk_size size);
5835 NK_LIB void nk_mfree(nk_handle unused, void *ptr);
5836 #endif
5837 NK_LIB void* nk_buffer_align(void *unaligned, nk_size align, nk_size *alignment, enum
       nk_buffer_allocation_type type);
5838 NK_LIB void* nk_buffer_alloc(struct nk_buffer *b, enum nk_buffer_allocation_type type, nk_size size,
       nk_size align);
5839 NK_LIB void* nk_buffer_realloc(struct nk_buffer *b, nk_size capacity, nk_size *size);
5840
5841 /* draw */
5842 NK_LIB void nk_command_buffer_init(struct nk_command_buffer *cb, struct nk_buffer *b, enum
       nk_command_clipping clip);
5843 NK_LIB void nk_command_buffer_reset(struct nk_command_buffer *b);
5844 NK_LIB void* nk_command_buffer_push(struct nk_command_buffer* b, enum nk_command_type t, nk_size size);
5845 NK_LIB void nk_draw_symbol(struct nk_command_buffer *out, enum nk_symbol_type type, struct nk_rect
       content, struct nk_color background, struct nk_color foreground, float border_width, const struct
       nk_user_font *font);
5846
5847 /* buffering */
5848 NK_LIB void nk_start_buffer(struct nk_context *ctx, struct nk_command_buffer *b);
5849 NK_LIB void nk_start(struct nk_context *ctx, struct nk_window *win);
5850 NK_LIB void nk_start_popup(struct nk_context *ctx, struct nk_window *win);
5851 NK_LIB void nk_finish_popup(struct nk_context *ctx, struct nk_window*);
5852 NK_LIB void nk_finish_buffer(struct nk_context *ctx, struct nk_command_buffer *b);
5853 NK_LIB void nk_finish(struct nk_context *ctx, struct nk_window *w);
5854 NK_LIB void nk_build(struct nk_context *ctx);
5855
5856 /* text editor */
5857 NK_LIB void nk_textedit_clear_state(struct nk_text_edit *state, enum nk_text_edit_type type,
       nk_plugin_filter filter);
5858 NK_LIB void nk_textedit_click(struct nk_text_edit *state, float x, float y, const struct nk_user_font
       *font, float row_height);
5859 NK_LIB void nk_textedit_drag(struct nk_text_edit *state, float x, float y, const struct nk_user_font
       *font, float row_height);
5860 NK_LIB void nk_textedit_key(struct nk_text_edit *state, enum nk_keys key, int shift_mod, const struct
       nk_user_font *font, float row_height);
5861
5862 /* window */
5863 enum nk_window_insert_location {
5864     NK_INSERT_BACK, /* inserts window into the back of list (front of screen) */
5865     NK_INSERT_FRONT /* inserts window into the front of list (back of screen) */
5866 };
5867 NK_LIB void *nk_create_window(struct nk_context *ctx);
5868 NK_LIB void nk_remove_window(struct nk_context*, struct nk_window*);
5869 NK_LIB void nk_free_window(struct nk_context *ctx, struct nk_window *win);
5870 NK_LIB struct nk_window *nk_find_window(struct nk_context *ctx, nk_hash hash, const char *name);
5871 NK_LIB void nk_insert_window(struct nk_context *ctx, struct nk_window *win, enum
       nk_window_insert_location loc);
5872
5873 /* pool */
5874 NK_LIB void nk_pool_init(struct nk_pool *pool, struct nk_allocator *alloc, unsigned int capacity);
5875 NK_LIB void nk_pool_free(struct nk_pool *pool);
5876 NK_LIB void nk_pool_init_fixed(struct nk_pool *pool, void *memory, nk_size size);
5877 NK_LIB struct nk_page_element *nk_pool_alloc(struct nk_pool *pool);
5878
5879 /* page-element */
5880 NK_LIB struct nk_page_element* nk_create_page_element(struct nk_context *ctx);
5881 NK_LIB void nk_link_page_element_into_freelist(struct nk_context *ctx, struct nk_page_element *elem);
5882 NK_LIB void nk_free_page_element(struct nk_context *ctx, struct nk_page_element *elem);
5883
5884 /* table */
5885 NK_LIB struct nk_table* nk_create_table(struct nk_context *ctx);
5886 NK_LIB void nk_remove_table(struct nk_window *win, struct nk_table *tbl);
5887 NK_LIB void nk_free_table(struct nk_context *ctx, struct nk_table *tbl);
5888 NK_LIB void nk_push_table(struct nk_window *win, struct nk_table *tbl);
5889 NK_LIB nk_uint *nk_add_value(struct nk_context *ctx, struct nk_window *win, nk_hash name, nk_uint
       value);
5890 NK_LIB nk_uint *nk_find_value(struct nk_window *win, nk_hash name);
5891
5892 /* panel */
5893 NK_LIB void *nk_create_panel(struct nk_context *ctx);
5894 NK_LIB void nk_free_panel(struct nk_context*, struct nk_panel *pan);
5895 NK_LIB int nk_panel_has_header(nk_flags flags, const char *title);
5896 NK_LIB struct nk_vec2 nk_panel_get_padding(const struct nk_style *style, enum nk_panel_type type);
5897 NK_LIB float nk_panel_get_border(const struct nk_style *style, nk_flags flags, enum nk_panel_type
       type);
5898 NK_LIB struct nk_color nk_panel_get_border_color(const struct nk_style *style, enum nk_panel_type
       type);
5899 NK_LIB int nk_panel_is_sub(enum nk_panel_type type);
5900 NK_LIB int nk_panel_is_nonblock(enum nk_panel_type type);
5901 NK_LIB int nk_panel_begin(struct nk_context *ctx, const char *title, enum nk_panel_type panel_type);
5902 NK_LIB void nk_panel_end(struct nk_context *ctx);
5903
5904 /* layout */
```

```
5905 NK_LIB float nk_layout_row_calculate_usable_space(const struct nk_style *style, enum nk_panel_type
       type, float total_space, int columns);
5906 NK_LIB void nk_panel_layout(const struct nk_context *ctx, struct nk_window *win, float height, int
       cols);
5907 NK_LIB void nk_row_layout(struct nk_context *ctx, enum nk_layout_format fmt, float height, int cols,
       int width);
5908 NK_LIB void nk_panel_alloc_row(const struct nk_context *ctx, struct nk_window *win);
5909 NK_LIB void nk_layout_widget_space(struct nk_rect *bounds, const struct nk_context *ctx, struct
       nk_window *win, int modify);
5910 NK_LIB void nk_panel_alloc_space(struct nk_rect *bounds, const struct nk_context *ctx);
5911 NK_LIB void nk_layout_peek(struct nk_rect *bounds, struct nk_context *ctx);
5912
5913 /* popup */
5914 NK_LIB int nk_nonblock_begin(struct nk_context *ctx, nk_flags flags, struct nk_rect body, struct
       nk_rect header, enum nk_panel_type panel_type);
5915
5916 /* text */
5917 struct nk_text {
5918     struct nk_vec2 padding;
5919     struct nk_color background;
5920     struct nk_color text;
5921 };
5922 NK_LIB void nk_widget_text(struct nk_command_buffer *o, struct nk_rect b, const char *string, int len,
       const struct nk_text *t, nk_flags a, const struct nk_user_font *f);
5923 NK_LIB void nk_widget_text_wrap(struct nk_command_buffer *o, struct nk_rect b, const char *string, int
       len, const struct nk_text *t, const struct nk_user_font *f);
5924
5925 /* button */
5926 NK_LIB int nk_button_behavior(nk_flags *state, struct nk_rect r, const struct nk_input *i, enum
       nk_button_behavior behavior);
5927 NK_LIB const struct nk_style_item* nk_draw_button(struct nk_command_buffer *out, const struct nk_rect
       *bounds, nk_flags state, const struct nk_style_button *style);
5928 NK_LIB int nk_do_button(nk_flags *state, struct nk_command_buffer *out, struct nk_rect r, const struct
       nk_style_button *style, const struct nk_input *in, enum nk_button_behavior behavior, struct nk_rect
       *content);
5929 NK_LIB void nk_draw_button_text(struct nk_command_buffer *out, const struct nk_rect *bounds, const
       struct nk_rect *content, nk_flags state, const struct nk_style_button *style, const char *txt, int
       len, nk_flags text_alignment, const struct nk_user_font *font);
5930 NK_LIB int nk_do_button_text(nk_flags *state, struct nk_command_buffer *out, struct nk_rect bounds,
       const char *string, int len, nk_flags align, enum nk_button_behavior behavior, const struct
       nk_style_button *style, const struct nk_input *in, const struct nk_user_font *font);
5931 NK_LIB void nk_draw_button_symbol(struct nk_command_buffer *out, const struct nk_rect *bounds, const
       struct nk_rect *content, nk_flags state, const struct nk_style_button *style, enum nk_symbol_type
       type, const struct nk_user_font *font);
5932 NK_LIB int nk_do_button_symbol(nk_flags *state, struct nk_command_buffer *out, struct nk_rect bounds,
       enum nk_symbol_type symbol, enum nk_button_behavior behavior, const struct nk_style_button *style,
       const struct nk_input *in, const struct nk_user_font *font);
5933 NK_LIB void nk_draw_button_image(struct nk_command_buffer *out, const struct nk_rect *bounds, const
       struct nk_rect *content, nk_flags state, const struct nk_style_button *style, const struct nk_image
       *img);
5934 NK_LIB int nk_do_button_image(nk_flags *state, struct nk_command_buffer *out, struct nk_rect bounds,
       struct nk_image img, enum nk_button_behavior b, const struct nk_style_button *style, const struct
       nk_input *in);
5935 NK_LIB void nk_draw_button_text_symbol(struct nk_command_buffer *out, const struct nk_rect *bounds,
       const struct nk_rect *label, const struct nk_rect *symbol, nk_flags state, const struct
       nk_style_button *style, const char *str, int len, enum nk_symbol_type type, const struct nk_user_font
       *font);
5936 NK_LIB int nk_do_button_text_symbol(nk_flags *state, struct nk_command_buffer *out, struct nk_rect
       bounds, enum nk_symbol_type symbol, const char *str, int len, nk_flags align, enum nk_button_behavior
       behavior, const struct nk_style_button *style, const struct nk_user_font *font, const struct nk_input
       *in);
5937 NK_LIB void nk_draw_button_text_image(struct nk_command_buffer *out, const struct nk_rect *bounds,
       const struct nk_rect *label, const struct nk_rect *image, nk_flags state, const struct
       nk_style_button *style, const char *str, int len, const struct nk_user_font *font, const struct
       nk_image *img);
5938 NK_LIB int nk_do_button_text_image(nk_flags *state, struct nk_command_buffer *out, struct nk_rect
       bounds, struct nk_image img, const char* str, int len, nk_flags align, enum nk_button_behavior
       behavior, const struct nk_style_button *style, const struct nk_user_font *font, const struct nk_input
       *in);
5939
5940 /* toggle */
5941 enum nk_toggle_type {
5942     NK_TOGGLE_CHECK,
5943     NK_TOGGLE_OPTION
5944 };
5945 NK_LIB int nk_toggle_behavior(const struct nk_input *in, struct nk_rect select, nk_flags *state, int
       active);
5946 NK_LIB void nk_draw_checkbox(struct nk_command_buffer *out, nk_flags state, const struct
       nk_style_toggle *style, int active, const struct nk_rect *label, const struct nk_rect *selector,
       const struct nk_rect *cursors, const char *string, int len, const struct nk_user_font *font);
5947 NK_LIB void nk_draw_option(struct nk_command_buffer *out, nk_flags state, const struct nk_style_toggle
       *style, int active, const struct nk_rect *label, const struct nk_rect *selector, const struct nk_rect
       *cursors, const char *string, int len, const struct nk_user_font *font);
5948 NK_LIB int nk_do_toggle(nk_flags *state, struct nk_command_buffer *out, struct nk_rect r, int *active,
       const char *str, int len, enum nk_toggle_type type, const struct nk_style_toggle *style, const struct
       nk_input *in, const struct nk_user_font *font);
5949
```

```
5950 /* progress */
5951 NK_LIB nk_size nk_progress_behavior(nk_flags *state, struct nk_input *in, struct nk_rect r, struct
       nk_rect cursor, nk_size max, nk_size value, int modifiable);
5952 NK_LIB void nk_draw_progress(struct nk_command_buffer *out, nk_flags state, const struct
       nk_style_progress *style, const struct nk_rect *bounds, const struct nk_rect *scursor, nk_size value,
       nk_size max);
5953 NK_LIB nk_size nk_do_progress(nk_flags *state, struct nk_command_buffer *out, struct nk_rect bounds,
       nk_size value, nk_size max, int modifiable, const struct nk_style_progress *style, struct nk_input
       *in);
5954
5955 /* slider */
5956 NK_LIB float nk_slider_behavior(nk_flags *state, struct nk_rect *logical_cursor, struct nk_rect
       *visual_cursor, struct nk_input *in, struct nk_rect bounds, float slider_min, float slider_max, float
       slider_value, float slider_step, float slider_steps);
5957 NK_LIB void nk_draw_slider(struct nk_command_buffer *out, nk_flags state, const struct nk_style_slider
       *style, const struct nk_rect *bounds, const struct nk_rect *visual_cursor, float min, float value,
       float max);
5958 NK_LIB float nk_do_slider(nk_flags *state, struct nk_command_buffer *out, struct nk_rect bounds, float
       min, float val, float max, float step, const struct nk_style_slider *style, struct nk_input *in,
       const struct nk_user_font *font);
5959
5960 /* scrollbar */
5961 NK_LIB float nk_scrollbar_behavior(nk_flags *state, struct nk_input *in, int has_scrolling, const
       struct nk_rect *scroll, const struct nk_rect *cursor, const struct nk_rect *empty0, const struct
       nk_rect *empty1, float scroll_offset, float target, float scroll_step, enum nk_orientation o);
5962 NK_LIB void nk_draw_scrollbar(struct nk_command_buffer *out, nk_flags state, const struct
       nk_style_scrollbar *style, const struct nk_rect *bounds, const struct nk_rect *scroll);
5963 NK_LIB float nk_do_scrollbarv(nk_flags *state, struct nk_command_buffer *out, struct nk_rect scroll,
       int has_scrolling, float offset, float target, float step, float button_pixel_inc, const struct
       nk_style_scrollbar *style, struct nk_input *in, const struct nk_user_font *font);
5964 NK_LIB float nk_do_scrollbarh(nk_flags *state, struct nk_command_buffer *out, struct nk_rect scroll,
       int has_scrolling, float offset, float target, float step, float button_pixel_inc, const struct
       nk_style_scrollbar *style, struct nk_input *in, const struct nk_user_font *font);
5965
5966 /* selectable */
5967 NK_LIB void nk_draw_selectable(struct nk_command_buffer *out, nk_flags state, const struct
       nk_style_selectable *style, int active, const struct nk_rect *bounds, const struct nk_rect *icon,
       const struct nk_image *img, enum nk_symbol_type sym, const char *string, int len, nk_flags align,
       const struct nk_user_font *font);
5968 NK_LIB int nk_do_selectable(nk_flags *state, struct nk_command_buffer *out, struct nk_rect bounds,
       const char *str, int len, nk_flags align, int *value, const struct nk_style_selectable *style, const
       struct nk_input *in, const struct nk_user_font *font);
5969 NK_LIB int nk_do_selectable_image(nk_flags *state, struct nk_command_buffer *out, struct nk_rect
       bounds, const char *str, int len, nk_flags align, int *value, const struct nk_image *img, const
       struct nk_style_selectable *style, const struct nk_input *in, const struct nk_user_font *font);
5970
5971 /* edit */
5972 NK_LIB void nk_edit_draw_text(struct nk_command_buffer *out, const struct nk_style_edit *style, float
       pos_x, float pos_y, float x_offset, const char *text, int byte_len, float row_height, const struct
       nk_user_font *font, struct nk_color background, struct nk_color foreground, int is_selected);
5973 NK_LIB nk_flags nk_do_edit(nk_flags *state, struct nk_command_buffer *out, struct nk_rect bounds,
       nk_flags flags, nk_plugin_filter filter, struct nk_text_edit *edit, const struct nk_style_edit
       *style, struct nk_input *in, const struct nk_user_font *font);
5974
5975 /* color-picker */
5976 NK_LIB int nk_color_picker_behavior(nk_flags *state, const struct nk_rect *bounds, const struct nk_rect
       *matrix, const struct nk_rect *hue_bar, const struct nk_rect *alpha_bar, struct nk_colorf *color,
       const struct nk_input *in);
5977 NK_LIB void nk_draw_color_picker(struct nk_command_buffer *o, const struct nk_rect *matrix, const
       struct nk_rect *hue_bar, const struct nk_rect *alpha_bar, struct nk_colorf col);
5978 NK_LIB int nk_do_color_picker(nk_flags *state, struct nk_command_buffer *out, struct nk_colorf *col,
       enum nk_color_format fmt, struct nk_rect bounds, struct nk_vec2 padding, const struct nk_input *in,
       const struct nk_user_font *font);
5979
5980 /* property */
5981 enum nk_property_status {
5982     NK_PROPERTY_DEFAULT,
5983     NK_PROPERTY_EDIT,
5984     NK_PROPERTY_DRAG
5985 };
5986 enum nk_property_filter {
5987     NK_FILTER_INT,
5988     NK_FILTER_FLOAT
5989 };
5990 enum nk_property_kind {
5991     NK_PROPERTY_INT,
5992     NK_PROPERTY_FLOAT,
5993     NK_PROPERTY_DOUBLE
5994 };
5995 union nk_property {
5996     int i;
5997     float f;
5998     double d;
5999 };
6000 struct nk_property_variant {
6001     enum nk_property_kind kind;
6002     union nk_property value;
```

```
6003     union nk_property min_value;
6004     union nk_property max_value;
6005     union nk_property step;
6006 };
6007 NK_LIB struct nk_property_variant nk_property_variant_int(int value, int min_value, int max_value, int
       step);
6008 NK_LIB struct nk_property_variant nk_property_variant_float(float value, float min_value, float
       max_value, float step);
6009 NK_LIB struct nk_property_variant nk_property_variant_double(double value, double min_value, double
       max_value, double step);
6010
6011 NK_LIB void nk_drag_behavior(nk_flags *state, const struct nk_input *in, struct nk_rect drag, struct
       nk_property_variant *variant, float inc_per_pixel);
6012 NK_LIB void nk_property_behavior(nk_flags *ws, const struct nk_input *in, struct nk_rect property,
       struct nk_rect label, struct nk_rect edit, struct nk_rect empty, int *state, struct
       nk_property_variant *variant, float inc_per_pixel);
6013 NK_LIB void nk_draw_property(struct nk_command_buffer *out, const struct nk_style_property *style,
       const struct nk_rect *bounds, const struct nk_rect *label, nk_flags state, const char *name, int len,
       const struct nk_user_font *font);
6014 NK_LIB void nk_do_property(nk_flags *ws, struct nk_command_buffer *out, struct nk_rect property, const
       char *name, struct nk_property_variant *variant, float inc_per_pixel, char *buffer, int *len, int
       *state, int *cursor, int *select_begin, int *select_end, const struct nk_style_property *style, enum
       nk_property_filter filter, struct nk_input *in, const struct nk_user_font *font, struct nk_text_edit
       *text_edit, enum nk_button_behavior behavior);
6015 NK_LIB void nk_property(struct nk_context *ctx, const char *name, struct nk_property_variant *variant,
       float inc_per_pixel, const enum nk_property_filter filter);
6016
6017 #endif
6018
6019
6020
6021
6022
6023 /* ===============================================================
6024  *
6025  *                              MATH
6026  *
6027  * ===============================================================*/
6028 /*  Since nuklear is supposed to work on all systems providing floating point
6029     math without any dependencies I also had to implement my own math functions
6030     for sqrt, sin and cos. Since the actual highly accurate implementations for
6031     the standard library functions are quite complex and I do not need high
6032     precision for my use cases I use approximations.
6033
6034     Sqrt
6035     ----
6036     For square root nuklear uses the famous fast inverse square root:
6037     https://en.wikipedia.org/wiki/Fast_inverse_square_root with
6038     slightly tweaked magic constant. While on today's hardware it is
6039     probably not faster it is still fast and accurate enough for
6040     nuklear's use cases. IMPORTANT: this requires float format IEEE 754
6041
6042     Sine/Cosine
6043     -----------
6044     All constants inside both function are generated Remez's minimax
6045     approximations for value range 0...2*PI. The reason why I decided to
6046     approximate exactly that range is that nuklear only needs sine and
6047     cosine to generate circles which only requires that exact range.
6048     In addition I used Remez instead of Taylor for additional precision:
6049     www.lolengine.net/blog/2011/12/21/better-function-approximations.
6050
6051     The tool I used to generate constants for both sine and cosine
6052     (it can actually approximate a lot more functions) can be
6053     found here: www.lolengine.net/wiki/oss/lolremez
6054 */
6055 NK_LIB float
6056 nk_inv_sqrt(float n)
6057 {
6058     float x2;
6059     const float threehalfs = 1.5f;
6060     union {nk_uint i; float f;} conv = {0};
6061     conv.f = n;
6062     x2 = n * 0.5f;
6063     conv.i = 0x5f375A84 - (conv.i >> 1);
6064     conv.f = conv.f * (threehalfs - (x2 * conv.f * conv.f));
6065     return conv.f;
6066 }
6067 NK_LIB float
6068 nk_sqrt(float x)
6069 {
6070     return x * nk_inv_sqrt(x);
6071 }
6072 NK_LIB float
6073 nk_sin(float x)
6074 {
6075     NK_STORAGE const float a0 = +1.91059300966915117e-31f;
6076     NK_STORAGE const float a1 = +1.00086760103908896f;
```

```
6077     NK_STORAGE const float a2 = -1.21276126894734565e-2f;
6078     NK_STORAGE const float a3 = -1.38078780785773762e-1f;
6079     NK_STORAGE const float a4 = -2.67353392911981221e-2f;
6080     NK_STORAGE const float a5 = +2.08026600266304389e-2f;
6081     NK_STORAGE const float a6 = -3.03996055049204407e-3f;
6082     NK_STORAGE const float a7 = +1.38235642404333740e-4f;
6083     return a0 + x*(a1 + x*(a2 + x*(a3 + x*(a4 + x*(a5 + x*(a6 + x*a7))))));
6084 }
6085 NK_LIB float
6086 nk_cos(float x)
6087 {
6088     /* New implementation. Also generated using lolremez. */
6089     /* Old version significantly deviated from expected results. */
6090     NK_STORAGE const float a0 = 9.9995999154986614e-1f;
6091     NK_STORAGE const float a1 = 1.2548995793001028e-3f;
6092     NK_STORAGE const float a2 = -5.0648546280678015e-1f;
6093     NK_STORAGE const float a3 = 1.2942246466519995e-2f;
6094     NK_STORAGE const float a4 = 2.8668384702547972e-2f;
6095     NK_STORAGE const float a5 = 7.3726485210586547e-3f;
6096     NK_STORAGE const float a6 = -3.8510875386947414e-3f;
6097     NK_STORAGE const float a7 = 4.7196604604366623e-4f;
6098     NK_STORAGE const float a8 = -1.8776444013090451e-5f;
6099     return a0 + x*(a1 + x*(a2 + x*(a3 + x*(a4 + x*(a5 + x*(a6 + x*(a7 + x*a8)))))));
6100 }
6101 NK_LIB nk_uint
6102 nk_round_up_pow2(nk_uint v)
6103 {
6104     v--;
6105     v |= v >> 1;
6106     v |= v >> 2;
6107     v |= v >> 4;
6108     v |= v >> 8;
6109     v |= v >> 16;
6110     v++;
6111     return v;
6112 }
6113 NK_LIB double
6114 nk_pow(double x, int n)
6115 {
6116     /*  check the sign of n */
6117     double r = 1;
6118     int plus = n >= 0;
6119     n = (plus) ? n : -n;
6120     while (n > 0) {
6121         if ((n & 1) == 1)
6122             r *= x;
6123         n /= 2;
6124         x *= x;
6125     }
6126     return plus ? r : 1.0 / r;
6127 }
6128 NK_LIB int
6129 nk_ifloord(double x)
6130 {
6131     x = (double)((int)x - ((x < 0.0) ? 1 : 0));
6132     return (int)x;
6133 }
6134 NK_LIB int
6135 nk_ifloorf(float x)
6136 {
6137     x = (float)((int)x - ((x < 0.0f) ? 1 : 0));
6138     return (int)x;
6139 }
6140 NK_LIB int
6141 nk_iceilf(float x)
6142 {
6143     if (x >= 0) {
6144         int i = (int)x;
6145         return (x > i) ? i+1: i;
6146     } else {
6147         int t = (int)x;
6148         float r = x - (float)t;
6149         return (r > 0.0f) ? t+1: t;
6150     }
6151 }
6152 NK_LIB int
6153 nk_log10(double n)
6154 {
6155     int neg;
6156     int ret;
6157     int exp = 0;
6158
6159     neg = (n < 0) ? 1 : 0;
6160     ret = (neg) ? (int)-n : (int)n;
6161     while ((ret / 10) > 0) {
6162         ret /= 10;
6163         exp++;
```

```
6164     }
6165     if (neg) exp = -exp;
6166     return exp;
6167 }
6168 NK_API struct nk_rect
6169 nk_get_null_rect(void)
6170 {
6171     return nk_null_rect;
6172 }
6173 NK_API struct nk_rect
6174 nk_rect(float x, float y, float w, float h)
6175 {
6176     struct nk_rect r;
6177     r.x = x; r.y = y;
6178     r.w = w; r.h = h;
6179     return r;
6180 }
6181 NK_API struct nk_rect
6182 nk_recti(int x, int y, int w, int h)
6183 {
6184     struct nk_rect r;
6185     r.x = (float)x;
6186     r.y = (float)y;
6187     r.w = (float)w;
6188     r.h = (float)h;
6189     return r;
6190 }
6191 NK_API struct nk_rect
6192 nk_recta(struct nk_vec2 pos, struct nk_vec2 size)
6193 {
6194     return nk_rect(pos.x, pos.y, size.x, size.y);
6195 }
6196 NK_API struct nk_rect
6197 nk_rectv(const float *r)
6198 {
6199     return nk_rect(r[0], r[1], r[2], r[3]);
6200 }
6201 NK_API struct nk_rect
6202 nk_rectiv(const int *r)
6203 {
6204     return nk_recti(r[0], r[1], r[2], r[3]);
6205 }
6206 NK_API struct nk_vec2
6207 nk_rect_pos(struct nk_rect r)
6208 {
6209     struct nk_vec2 ret;
6210     ret.x = r.x; ret.y = r.y;
6211     return ret;
6212 }
6213 NK_API struct nk_vec2
6214 nk_rect_size(struct nk_rect r)
6215 {
6216     struct nk_vec2 ret;
6217     ret.x = r.w; ret.y = r.h;
6218     return ret;
6219 }
6220 NK_LIB struct nk_rect
6221 nk_shrink_rect(struct nk_rect r, float amount)
6222 {
6223     struct nk_rect res;
6224     r.w = NK_MAX(r.w, 2 * amount);
6225     r.h = NK_MAX(r.h, 2 * amount);
6226     res.x = r.x + amount;
6227     res.y = r.y + amount;
6228     res.w = r.w - 2 * amount;
6229     res.h = r.h - 2 * amount;
6230     return res;
6231 }
6232 NK_LIB struct nk_rect
6233 nk_pad_rect(struct nk_rect r, struct nk_vec2 pad)
6234 {
6235     r.w = NK_MAX(r.w, 2 * pad.x);
6236     r.h = NK_MAX(r.h, 2 * pad.y);
6237     r.x += pad.x; r.y += pad.y;
6238     r.w -= 2 * pad.x;
6239     r.h -= 2 * pad.y;
6240     return r;
6241 }
6242 NK_API struct nk_vec2
6243 nk_vec2(float x, float y)
6244 {
6245     struct nk_vec2 ret;
6246     ret.x = x; ret.y = y;
6247     return ret;
6248 }
6249 NK_API struct nk_vec2
6250 nk_vec2i(int x, int y)
```

```
6251 {
6252     struct nk_vec2 ret;
6253     ret.x = (float)x;
6254     ret.y = (float)y;
6255     return ret;
6256 }
6257 NK_API struct nk_vec2
6258 nk_vec2v(const float *v)
6259 {
6260     return nk_vec2(v[0], v[1]);
6261 }
6262 NK_API struct nk_vec2
6263 nk_vec2iv(const int *v)
6264 {
6265     return nk_vec2i(v[0], v[1]);
6266 }
6267 NK_LIB void
6268 nk_unify(struct nk_rect *clip, const struct nk_rect *a, float x0, float y0,
6269     float x1, float y1)
6270 {
6271     NK_ASSERT(a);
6272     NK_ASSERT(clip);
6273     clip->x = NK_MAX(a->x, x0);
6274     clip->y = NK_MAX(a->y, y0);
6275     clip->w = NK_MIN(a->x + a->w, x1) - clip->x;
6276     clip->h = NK_MIN(a->y + a->h, y1) - clip->y;
6277     clip->w = NK_MAX(0, clip->w);
6278     clip->h = NK_MAX(0, clip->h);
6279 }
6280
6281 NK_API void
6282 nk_triangle_from_direction(struct nk_vec2 *result, struct nk_rect r,
6283     float pad_x, float pad_y, enum nk_heading direction)
6284 {
6285     float w_half, h_half;
6286     NK_ASSERT(result);
6287
6288     r.w = NK_MAX(2 * pad_x, r.w);
6289     r.h = NK_MAX(2 * pad_y, r.h);
6290     r.w = r.w - 2 * pad_x;
6291     r.h = r.h - 2 * pad_y;
6292
6293     r.x = r.x + pad_x;
6294     r.y = r.y + pad_y;
6295
6296     w_half = r.w / 2.0f;
6297     h_half = r.h / 2.0f;
6298
6299     if (direction == NK_UP) {
6300         result[0] = nk_vec2(r.x + w_half, r.y);
6301         result[1] = nk_vec2(r.x + r.w, r.y + r.h);
6302         result[2] = nk_vec2(r.x, r.y + r.h);
6303     } else if (direction == NK_RIGHT) {
6304         result[0] = nk_vec2(r.x, r.y);
6305         result[1] = nk_vec2(r.x + r.w, r.y + h_half);
6306         result[2] = nk_vec2(r.x, r.y + r.h);
6307     } else if (direction == NK_DOWN) {
6308         result[0] = nk_vec2(r.x, r.y);
6309         result[1] = nk_vec2(r.x + r.w, r.y);
6310         result[2] = nk_vec2(r.x + w_half, r.y + r.h);
6311     } else {
6312         result[0] = nk_vec2(r.x, r.y + h_half);
6313         result[1] = nk_vec2(r.x + r.w, r.y);
6314         result[2] = nk_vec2(r.x + r.w, r.y + r.h);
6315     }
6316 }
6317
6318
6319
6320
6321
6322 /* ===============================================================
6323  *
6324  *                              UTIL
6325  *
6326  * ===============================================================*/
6327 NK_INTERN int nk_str_match_here(const char *regexp, const char *text);
6328 NK_INTERN int nk_str_match_star(int c, const char *regexp, const char *text);
6329 NK_LIB int nk_is_lower(int c) {return (c >= 'a' && c <= 'z') || (c >= 0xE0 && c <= 0xFF);}
6330 NK_LIB int nk_is_upper(int c){return (c >= 'A' && c <= 'Z') || (c >= 0xC0 && c <= 0xDF);}
6331 NK_LIB int nk_to_upper(int c) {return (c >= 'a' && c <= 'z') ? (c - ('a' - 'A')) : c;}
6332 NK_LIB int nk_to_lower(int c) {return (c >= 'A' && c <= 'Z') ? (c - ('a' + 'A')) : c;}
6333
6334 NK_LIB void*
6335 nk_memcopy(void *dst0, const void *src0, nk_size length)
6336 {
6337     nk_ptr t;
```

```
6338     char *dst = (char*)dst0;
6339     const char *src = (const char*)src0;
6340     if (length == 0 || dst == src)
6341         goto done;
6342 
6343     #define nk_word int
6344     #define nk_wsize sizeof(nk_word)
6345     #define nk_wmask (nk_wsize-1)
6346     #define NK_TLOOP(s) if (t) NK_TLOOP1(s)
6347     #define NK_TLOOP1(s) do { s; } while (--t)
6348 
6349     if (dst < src) {
6350         t = (nk_ptr)src; /* only need low bits */
6351         if ((t | (nk_ptr)dst) & nk_wmask) {
6352             if ((t ^ (nk_ptr)dst) & nk_wmask || length < nk_wsize)
6353                 t = length;
6354             else
6355                 t = nk_wsize - (t & nk_wmask);
6356             length -= t;
6357             NK_TLOOP1(*dst++ = *src++);
6358         }
6359         t = length / nk_wsize;
6360         NK_TLOOP(*(nk_word*)(void*)dst = *(const nk_word*)(const void*)src;
6361             src += nk_wsize; dst += nk_wsize);
6362         t = length & nk_wmask;
6363         NK_TLOOP(*dst++ = *src++);
6364     } else {
6365         src += length;
6366         dst += length;
6367         t = (nk_ptr)src;
6368         if ((t | (nk_ptr)dst) & nk_wmask) {
6369             if ((t ^ (nk_ptr)dst) & nk_wmask || length <= nk_wsize)
6370                 t = length;
6371             else
6372                 t &= nk_wmask;
6373             length -= t;
6374             NK_TLOOP1(*--dst = *--src);
6375         }
6376         t = length / nk_wsize;
6377         NK_TLOOP(src -= nk_wsize; dst -= nk_wsize;
6378             *(nk_word*)(void*)dst = *(const nk_word*)(const void*)src);
6379         t = length & nk_wmask;
6380         NK_TLOOP(*--dst = *--src);
6381     }
6382     #undef nk_word
6383     #undef nk_wsize
6384     #undef nk_wmask
6385     #undef NK_TLOOP
6386     #undef NK_TLOOP1
6387 done:
6388     return (dst0);
6389 }
6390 NK_LIB void
6391 nk_memset(void *ptr, int c0, nk_size size)
6392 {
6393     #define nk_word unsigned
6394     #define nk_wsize sizeof(nk_word)
6395     #define nk_wmask (nk_wsize - 1)
6396     nk_byte *dst = (nk_byte*)ptr;
6397     unsigned c = 0;
6398     nk_size t = 0;
6399 
6400     if ((c = (nk_byte)c0) != 0) {
6401         c = (c << 8) | c; /* at least 16-bits  */
6402         if (sizeof(unsigned int) > 2)
6403             c = (c << 16) | c; /* at least 32-bits*/
6404     }
6405 
6406     /* too small of a word count */
6407     dst = (nk_byte*)ptr;
6408     if (size < 3 * nk_wsize) {
6409         while (size--) *dst++ = (nk_byte)c0;
6410         return;
6411     }
6412 
6413     /* align destination */
6414     if ((t = NK_PTR_TO_UINT(dst) & nk_wmask) != 0) {
6415         t = nk_wsize -t;
6416         size -= t;
6417         do {
6418             *dst++ = (nk_byte)c0;
6419         } while (--t != 0);
6420     }
6421 
6422     /* fill word */
6423     t = size / nk_wsize;
6424     do {
```

```
6425         *(nk_word*)((void*)dst) = c;
6426         dst += nk_wsize;
6427     } while (--t != 0);
6428
6429     /* fill trailing bytes */
6430     t = (size & nk_wmask);
6431     if (t != 0) {
6432         do {
6433             *dst++ = (nk_byte)c0;
6434         } while (--t != 0);
6435     }
6436
6437     #undef nk_word
6438     #undef nk_wsize
6439     #undef nk_wmask
6440 }
6441 NK_LIB void
6442 nk_zero(void *ptr, nk_size size)
6443 {
6444     NK_ASSERT(ptr);
6445     NK_MEMSET(ptr, 0, size);
6446 }
6447 NK_API int
6448 nk_strlen(const char *str)
6449 {
6450     int siz = 0;
6451     NK_ASSERT(str);
6452     while (str && *str++ != '\0') siz++;
6453     return siz;
6454 }
6455 NK_API int
6456 nk_strtoi(const char *str, const char **endptr)
6457 {
6458     int neg = 1;
6459     const char *p = str;
6460     int value = 0;
6461
6462     NK_ASSERT(str);
6463     if (!str) return 0;
6464
6465     /* skip whitespace */
6466     while (*p == ' ') p++;
6467     if (*p == '-') {
6468         neg = -1;
6469         p++;
6470     }
6471     while (*p && *p >= '0' && *p <= '9') {
6472         value = value * 10 + (int) (*p - '0');
6473         p++;
6474     }
6475     if (endptr)
6476         *endptr = p;
6477     return neg*value;
6478 }
6479 NK_API double
6480 nk_strtod(const char *str, const char **endptr)
6481 {
6482     double m;
6483     double neg = 1.0;
6484     const char *p = str;
6485     double value = 0;
6486     double number = 0;
6487
6488     NK_ASSERT(str);
6489     if (!str) return 0;
6490
6491     /* skip whitespace */
6492     while (*p == ' ') p++;
6493     if (*p == '-') {
6494         neg = -1.0;
6495         p++;
6496     }
6497
6498     while (*p && *p != '.' && *p != 'e') {
6499         value = value * 10.0 + (double) (*p - '0');
6500         p++;
6501     }
6502
6503     if (*p == '.') {
6504         p++;
6505         for(m = 0.1; *p && *p != 'e'; p++ ) {
6506             value = value + (double) (*p - '0') * m;
6507             m *= 0.1;
6508         }
6509     }
6510     if (*p == 'e') {
6511         int i, pow, div;
```

```
6512         p++;
6513         if (*p == '-') {
6514             div = nk_true;
6515             p++;
6516         } else if (*p == '+') {
6517             div = nk_false;
6518             p++;
6519         } else div = nk_false;
6520
6521         for (pow = 0; *p; p++)
6522             pow = pow * 10 + (int) (*p - '0');
6523
6524         for (m = 1.0, i = 0; i < pow; i++)
6525             m *= 10.0;
6526
6527         if (div)
6528             value /= m;
6529         else value *= m;
6530     }
6531     number = value * neg;
6532     if (endptr)
6533         *endptr = p;
6534     return number;
6535 }
6536 NK_API float
6537 nk_strtof(const char *str, const char **endptr)
6538 {
6539     float float_value;
6540     double double_value;
6541     double_value = NK_STRTOD(str, endptr);
6542     float_value = (float)double_value;
6543     return float_value;
6544 }
6545 NK_API int
6546 nk_stricmp(const char *s1, const char *s2)
6547 {
6548     nk_int c1,c2,d;
6549     do {
6550         c1 = *s1++;
6551         c2 = *s2++;
6552         d = c1 - c2;
6553         while (d) {
6554             if (c1 <= 'Z' && c1 >= 'A') {
6555                 d += ('a' - 'A');
6556                 if (!d) break;
6557             }
6558             if (c2 <= 'Z' && c2 >= 'A') {
6559                 d -= ('a' - 'A');
6560                 if (!d) break;
6561             }
6562             return ((d >= 0) « 1) - 1;
6563         }
6564     } while (c1);
6565     return 0;
6566 }
6567 NK_API int
6568 nk_stricmpn(const char *s1, const char *s2, int n)
6569 {
6570     int c1,c2,d;
6571     NK_ASSERT(n >= 0);
6572     do {
6573         c1 = *s1++;
6574         c2 = *s2++;
6575         if (!n--) return 0;
6576
6577         d = c1 - c2;
6578         while (d) {
6579             if (c1 <= 'Z' && c1 >= 'A') {
6580                 d += ('a' - 'A');
6581                 if (!d) break;
6582             }
6583             if (c2 <= 'Z' && c2 >= 'A') {
6584                 d -= ('a' - 'A');
6585                 if (!d) break;
6586             }
6587             return ((d >= 0) « 1) - 1;
6588         }
6589     } while (c1);
6590     return 0;
6591 }
6592 NK_INTERN int
6593 nk_str_match_here(const char *regexp, const char *text)
6594 {
6595     if (regexp[0] == '\0')
6596         return 1;
6597     if (regexp[1] == '*')
6598         return nk_str_match_star(regexp[0], regexp+2, text);
```

```
6599     if (regexp[0] == '$' && regexp[1] == '\0')
6600         return *text == '\0';
6601     if (*text!='\0' && (regexp[0]=='.' || regexp[0]==*text))
6602         return nk_str_match_here(regexp+1, text+1);
6603     return 0;
6604 }
6605 NK_INTERN int
6606 nk_str_match_star(int c, const char *regexp, const char *text)
6607 {
6608     do {/* a '* matches zero or more instances */
6609         if (nk_str_match_here(regexp, text))
6610             return 1;
6611     } while (*text != '\0' && (*text++ == c || c == '.'));
6612     return 0;
6613 }
6614 NK_API int
6615 nk_strfilter(const char *text, const char *regexp)
6616 {
6617     /*
6618     c    matches any literal character c
6619     .    matches any single character
6620     ^    matches the beginning of the input string
6621     $    matches the end of the input string
6622     *    matches zero or more occurrences of the previous character*/
6623     if (regexp[0] == '^')
6624         return nk_str_match_here(regexp+1, text);
6625     do {    /* must look even if string is empty */
6626         if (nk_str_match_here(regexp, text))
6627             return 1;
6628     } while (*text++ != '\0');
6629     return 0;
6630 }
6631 NK_API int
6632 nk_strmatch_fuzzy_text(const char *str, int str_len,
6633     const char *pattern, int *out_score)
6634 {
6635     /* Returns true if each character in pattern is found sequentially within str
6636      * if found then out_score is also set. Score value has no intrinsic meaning.
6637      * Range varies with pattern. Can only compare scores with same search pattern. */
6638
6639     /* bonus for adjacent matches */
6640     #define NK_ADJACENCY_BONUS 5
6641     /* bonus if match occurs after a separator */
6642     #define NK_SEPARATOR_BONUS 10
6643     /* bonus if match is uppercase and prev is lower */
6644     #define NK_CAMEL_BONUS 10
6645     /* penalty applied for every letter in str before the first match */
6646     #define NK_LEADING_LETTER_PENALTY (-3)
6647     /* maximum penalty for leading letters */
6648     #define NK_MAX_LEADING_LETTER_PENALTY (-9)
6649     /* penalty for every letter that doesn't matter */
6650     #define NK_UNMATCHED_LETTER_PENALTY (-1)
6651
6652     /* loop variables */
6653     int score = 0;
6654     char const * pattern_iter = pattern;
6655     int str_iter = 0;
6656     int prev_matched = nk_false;
6657     int prev_lower = nk_false;
6658     /* true so if first letter match gets separator bonus*/
6659     int prev_separator = nk_true;
6660
6661     /* use "best" matched letter if multiple string letters match the pattern */
6662     char const * best_letter = 0;
6663     int best_letter_score = 0;
6664
6665     /* loop over strings */
6666     NK_ASSERT(str);
6667     NK_ASSERT(pattern);
6668     if (!str || !str_len || !pattern) return 0;
6669     while (str_iter < str_len)
6670     {
6671         const char pattern_letter = *pattern_iter;
6672         const char str_letter = str[str_iter];
6673
6674         int next_match = *pattern_iter != '\0' &&
6675             nk_to_lower(pattern_letter) == nk_to_lower(str_letter);
6676         int rematch = best_letter && nk_to_upper(*best_letter) == nk_to_upper(str_letter);
6677
6678         int advanced = next_match && best_letter;
6679         int pattern_repeat = best_letter && *pattern_iter != '\0';
6680         pattern_repeat = pattern_repeat &&
6681             nk_to_lower(*best_letter) == nk_to_lower(pattern_letter);
6682
6683         if (advanced || pattern_repeat) {
6684             score += best_letter_score;
6685             best_letter = 0;
```

```
6686            best_letter_score = 0;
6687        }
6688
6689        if (next_match || rematch)
6690        {
6691            int new_score = 0;
6692            /* Apply penalty for each letter before the first pattern match */
6693            if (pattern_iter == pattern) {
6694                int count = (int)(&str[str_iter] - str);
6695                int penalty = NK_LEADING_LETTER_PENALTY * count;
6696                if (penalty < NK_MAX_LEADING_LETTER_PENALTY)
6697                    penalty = NK_MAX_LEADING_LETTER_PENALTY;
6698
6699                score += penalty;
6700            }
6701
6702            /* apply bonus for consecutive bonuses */
6703            if (prev_matched)
6704                new_score += NK_ADJACENCY_BONUS;
6705
6706            /* apply bonus for matches after a separator */
6707            if (prev_separator)
6708                new_score += NK_SEPARATOR_BONUS;
6709
6710            /* apply bonus across camel case boundaries */
6711            if (prev_lower && nk_is_upper(str_letter))
6712                new_score += NK_CAMEL_BONUS;
6713
6714            /* update pattern iter IFF the next pattern letter was matched */
6715            if (next_match)
6716                ++pattern_iter;
6717
6718            /* update best letter in str which may be for a "next" letter or a rematch */
6719            if (new_score >= best_letter_score) {
6720                /* apply penalty for now skipped letter */
6721                if (best_letter != 0)
6722                    score += NK_UNMATCHED_LETTER_PENALTY;
6723
6724                best_letter = &str[str_iter];
6725                best_letter_score = new_score;
6726            }
6727            prev_matched = nk_true;
6728        } else {
6729            score += NK_UNMATCHED_LETTER_PENALTY;
6730            prev_matched = nk_false;
6731        }
6732
6733        /* separators should be more easily defined */
6734        prev_lower = nk_is_lower(str_letter) != 0;
6735        prev_separator = str_letter == '_' || str_letter == ' ';
6736
6737        ++str_iter;
6738    }
6739
6740    /* apply score for last match */
6741    if (best_letter)
6742        score += best_letter_score;
6743
6744    /* did not match full pattern */
6745    if (*pattern_iter != '\0')
6746        return nk_false;
6747
6748    if (out_score)
6749        *out_score = score;
6750    return nk_true;
6751 }
6752 NK_API int
6753 nk_strmatch_fuzzy_string(char const *str, char const *pattern, int *out_score)
6754 {
6755     return nk_strmatch_fuzzy_text(str, nk_strlen(str), pattern, out_score);
6756 }
6757 NK_LIB int
6758 nk_string_float_limit(char *string, int prec)
6759 {
6760     int dot = 0;
6761     char *c = string;
6762     while (*c) {
6763         if (*c == '.') {
6764             dot = 1;
6765             c++;
6766             continue;
6767         }
6768         if (dot == (prec+1)) {
6769             *c = 0;
6770             break;
6771         }
6772         if (dot > 0) dot++;
```

```
6773         c++;
6774     }
6775     return (int)(c - string);
6776 }
6777 NK_INTERN void
6778 nk_strrev_ascii(char *s)
6779 {
6780     int len = nk_strlen(s);
6781     int end = len / 2;
6782     int i = 0;
6783     char t;
6784     for (; i < end; ++i) {
6785         t = s[i];
6786         s[i] = s[len - 1 - i];
6787         s[len -1 - i] = t;
6788     }
6789 }
6790 NK_LIB char*
6791 nk_itoa(char *s, long n)
6792 {
6793     long i = 0;
6794     if (n == 0) {
6795         s[i++] = '0';
6796         s[i] = 0;
6797         return s;
6798     }
6799     if (n < 0) {
6800         s[i++] = '-';
6801         n = -n;
6802     }
6803     while (n > 0) {
6804         s[i++] = (char)('0' + (n % 10));
6805         n /= 10;
6806     }
6807     s[i] = 0;
6808     if (s[0] == '-')
6809         ++s;
6810
6811     nk_strrev_ascii(s);
6812     return s;
6813 }
6814 NK_LIB char*
6815 nk_dtoa(char *s, double n)
6816 {
6817     int useExp = 0;
6818     int digit = 0, m = 0, m1 = 0;
6819     char *c = s;
6820     int neg = 0;
6821
6822     NK_ASSERT(s);
6823     if (!s) return 0;
6824
6825     if (n == 0.0) {
6826         s[0] = '0'; s[1] = '\0';
6827         return s;
6828     }
6829
6830     neg = (n < 0);
6831     if (neg) n = -n;
6832
6833     /* calculate magnitude */
6834     m = nk_log10(n);
6835     useExp = (m >= 14 || (neg && m >= 9) || m <= -9);
6836     if (neg) *(c++) = '-';
6837
6838     /* set up for scientific notation */
6839     if (useExp) {
6840         if (m < 0)
6841            m -= 1;
6842         n = n / (double)nk_pow(10.0, m);
6843         m1 = m;
6844         m = 0;
6845     }
6846     if (m < 1.0) {
6847         m = 0;
6848     }
6849
6850     /* convert the number */
6851     while (n > NK_FLOAT_PRECISION || m >= 0) {
6852         double weight = nk_pow(10.0, m);
6853         if (weight > 0) {
6854             double t = (double)n / weight;
6855             digit = nk_ifloord(t);
6856             n -= ((double)digit * weight);
6857             *(c++) = (char)('0' + (char)digit);
6858         }
6859         if (m == 0 && n > 0)
```

```
6860            *(c++) = '.';
6861        m--;
6862    }
6863
6864    if (useExp) {
6865        /* convert the exponent */
6866        int i, j;
6867        *(c++) = 'e';
6868        if (m1 > 0) {
6869            *(c++) = '+';
6870        } else {
6871            *(c++) = '-';
6872            m1 = -m1;
6873        }
6874        m = 0;
6875        while (m1 > 0) {
6876            *(c++) = (char)('0' + (char)(m1 % 10));
6877            m1 /= 10;
6878            m++;
6879        }
6880        c -= m;
6881        for (i = 0, j = m-1; i<j; i++, j--) {
6882            /* swap without temporary */
6883            c[i] ^= c[j];
6884            c[j] ^= c[i];
6885            c[i] ^= c[j];
6886        }
6887        c += m;
6888    }
6889    *(c) = '\0';
6890    return s;
6891 }
6892 #ifdef NK_INCLUDE_STANDARD_VARARGS
6893 #ifndef NK_INCLUDE_STANDARD_IO
6894 NK_INTERN int
6895 nk_vsnprintf(char *buf, int buf_size, const char *fmt, va_list args)
6896 {
6897     enum nk_arg_type {
6898         NK_ARG_TYPE_CHAR,
6899         NK_ARG_TYPE_SHORT,
6900         NK_ARG_TYPE_DEFAULT,
6901         NK_ARG_TYPE_LONG
6902     };
6903     enum nk_arg_flags {
6904         NK_ARG_FLAG_LEFT = 0x01,
6905         NK_ARG_FLAG_PLUS = 0x02,
6906         NK_ARG_FLAG_SPACE = 0x04,
6907         NK_ARG_FLAG_NUM = 0x10,
6908         NK_ARG_FLAG_ZERO = 0x20
6909     };
6910
6911     char number_buffer[NK_MAX_NUMBER_BUFFER];
6912     enum nk_arg_type arg_type = NK_ARG_TYPE_DEFAULT;
6913     int precision = NK_DEFAULT;
6914     int width = NK_DEFAULT;
6915     nk_flags flag = 0;
6916
6917     int len = 0;
6918     int result = -1;
6919     const char *iter = fmt;
6920
6921     NK_ASSERT(buf);
6922     NK_ASSERT(buf_size);
6923     if (!buf || !buf_size || !fmt) return 0;
6924     for (iter = fmt; *iter && len < buf_size; iter++) {
6925         /* copy all non-format characters */
6926         while (*iter && (*iter != '%') && (len < buf_size))
6927             buf[len++] = *iter++;
6928         if (!(*iter) || len >= buf_size) break;
6929         iter++;
6930
6931         /* flag arguments */
6932         while (*iter) {
6933             if (*iter == '-') flag |= NK_ARG_FLAG_LEFT;
6934             else if (*iter == '+') flag |= NK_ARG_FLAG_PLUS;
6935             else if (*iter == ' ') flag |= NK_ARG_FLAG_SPACE;
6936             else if (*iter == '#') flag |= NK_ARG_FLAG_NUM;
6937             else if (*iter == '0') flag |= NK_ARG_FLAG_ZERO;
6938             else break;
6939             iter++;
6940         }
6941
6942         /* width argument */
6943         width = NK_DEFAULT;
6944         if (*iter >= '1' && *iter <= '9') {
6945             const char *end;
6946             width = nk_strtoi(iter, &end);
```

```
6947            if (end == iter)
6948                width = -1;
6949            else iter = end;
6950        } else if (*iter == '*') {
6951            width = va_arg(args, int);
6952            iter++;
6953        }
6954
6955        /* precision argument */
6956        precision = NK_DEFAULT;
6957        if (*iter == '.') {
6958            iter++;
6959            if (*iter == '*') {
6960                precision = va_arg(args, int);
6961                iter++;
6962            } else {
6963                const char *end;
6964                precision = nk_strtoi(iter, &end);
6965                if (end == iter)
6966                    precision = -1;
6967                else iter = end;
6968            }
6969        }
6970
6971        /* length modifier */
6972        if (*iter == 'h') {
6973            if (*(iter+1) == 'h') {
6974                arg_type = NK_ARG_TYPE_CHAR;
6975                iter++;
6976            } else arg_type = NK_ARG_TYPE_SHORT;
6977            iter++;
6978        } else if (*iter == 'l') {
6979            arg_type = NK_ARG_TYPE_LONG;
6980            iter++;
6981        } else arg_type = NK_ARG_TYPE_DEFAULT;
6982
6983        /* specifier */
6984        if (*iter == '%') {
6985            NK_ASSERT(arg_type == NK_ARG_TYPE_DEFAULT);
6986            NK_ASSERT(precision == NK_DEFAULT);
6987            NK_ASSERT(width == NK_DEFAULT);
6988            if (len < buf_size)
6989                buf[len++] = '%';
6990        } else if (*iter == 's') {
6991            /* string  */
6992            const char *str = va_arg(args, const char*);
6993            NK_ASSERT(str != buf && "buffer and argument are not allowed to overlap!");
6994            NK_ASSERT(arg_type == NK_ARG_TYPE_DEFAULT);
6995            NK_ASSERT(precision == NK_DEFAULT);
6996            NK_ASSERT(width == NK_DEFAULT);
6997            if (str == buf) return -1;
6998            while (str && *str && len < buf_size)
6999                buf[len++] = *str++;
7000        } else if (*iter == 'n') {
7001            /* current length callback */
7002            signed int *n = va_arg(args, int*);
7003            NK_ASSERT(arg_type == NK_ARG_TYPE_DEFAULT);
7004            NK_ASSERT(precision == NK_DEFAULT);
7005            NK_ASSERT(width == NK_DEFAULT);
7006            if (n) *n = len;
7007        } else if (*iter == 'c' || *iter == 'i' || *iter == 'd') {
7008            /* signed integer */
7009            long value = 0;
7010            const char *num_iter;
7011            int num_len, num_print, padding;
7012            int cur_precision = NK_MAX(precision, 1);
7013            int cur_width = NK_MAX(width, 0);
7014
7015            /* retrieve correct value type */
7016            if (arg_type == NK_ARG_TYPE_CHAR)
7017                value = (signed char)va_arg(args, int);
7018            else if (arg_type == NK_ARG_TYPE_SHORT)
7019                value = (signed short)va_arg(args, int);
7020            else if (arg_type == NK_ARG_TYPE_LONG)
7021                value = va_arg(args, signed long);
7022            else if (*iter == 'c')
7023                value = (unsigned char)va_arg(args, int);
7024            else value = va_arg(args, signed int);
7025
7026            /* convert number to string */
7027            nk_itoa(number_buffer, value);
7028            num_len = nk_strlen(number_buffer);
7029            padding = NK_MAX(cur_width - NK_MAX(cur_precision, num_len), 0);
7030            if ((flag & NK_ARG_FLAG_PLUS) || (flag & NK_ARG_FLAG_SPACE))
7031                padding = NK_MAX(padding-1, 0);
7032
7033            /* fill left padding up to a total of `width` characters */
```

```
7034            if (!(flag & NK_ARG_FLAG_LEFT)) {
7035                while (padding-- > 0 && (len < buf_size)) {
7036                    if ((flag & NK_ARG_FLAG_ZERO) && (precision == NK_DEFAULT))
7037                        buf[len++] = '0';
7038                    else buf[len++] = ' ';
7039                }
7040            }
7041
7042            /* copy string value representation into buffer */
7043            if ((flag & NK_ARG_FLAG_PLUS) && value >= 0 && len < buf_size)
7044                buf[len++] = '+';
7045            else if ((flag & NK_ARG_FLAG_SPACE) && value >= 0 && len < buf_size)
7046                buf[len++] = ' ';
7047
7048            /* fill up to precision number of digits with '0' */
7049            num_print = NK_MAX(cur_precision, num_len);
7050            while (precision && (num_print > num_len) && (len < buf_size)) {
7051                buf[len++] = '0';
7052                num_print--;
7053            }
7054
7055            /* copy string value representation into buffer */
7056            num_iter = number_buffer;
7057            while (precision && *num_iter && len < buf_size)
7058                buf[len++] = *num_iter++;
7059
7060            /* fill right padding up to width characters */
7061            if (flag & NK_ARG_FLAG_LEFT) {
7062                while ((padding-- > 0) && (len < buf_size))
7063                    buf[len++] = ' ';
7064            }
7065        } else if (*iter == 'o' || *iter == 'x' || *iter == 'X' || *iter == 'u') {
7066            /* unsigned integer */
7067            unsigned long value = 0;
7068            int num_len = 0, num_print, padding = 0;
7069            int cur_precision = NK_MAX(precision, 1);
7070            int cur_width = NK_MAX(width, 0);
7071            unsigned int base = (*iter == 'o') ? 8: (*iter == 'u')? 10: 16;
7072
7073            /* print oct/hex/dec value */
7074            const char *upper_output_format = "0123456789ABCDEF";
7075            const char *lower_output_format = "0123456789abcdef";
7076            const char *output_format = (*iter == 'x') ?
7077                lower_output_format: upper_output_format;
7078
7079            /* retrieve correct value type */
7080            if (arg_type == NK_ARG_TYPE_CHAR)
7081                value = (unsigned char)va_arg(args, int);
7082            else if (arg_type == NK_ARG_TYPE_SHORT)
7083                value = (unsigned short)va_arg(args, int);
7084            else if (arg_type == NK_ARG_TYPE_LONG)
7085                value = va_arg(args, unsigned long);
7086            else value = va_arg(args, unsigned int);
7087
7088            do {
7089                /* convert decimal number into hex/oct number */
7090                int digit = output_format[value % base];
7091                if (num_len < NK_MAX_NUMBER_BUFFER)
7092                    number_buffer[num_len++] = (char)digit;
7093                value /= base;
7094            } while (value > 0);
7095
7096            num_print = NK_MAX(cur_precision, num_len);
7097            padding = NK_MAX(cur_width - NK_MAX(cur_precision, num_len), 0);
7098            if (flag & NK_ARG_FLAG_NUM)
7099                padding = NK_MAX(padding-1, 0);
7100
7101            /* fill left padding up to a total of `width` characters */
7102            if (!(flag & NK_ARG_FLAG_LEFT)) {
7103                while ((padding-- > 0) && (len < buf_size)) {
7104                    if ((flag & NK_ARG_FLAG_ZERO) && (precision == NK_DEFAULT))
7105                        buf[len++] = '0';
7106                    else buf[len++] = ' ';
7107                }
7108            }
7109
7110            /* fill up to precision number of digits */
7111            if (num_print && (flag & NK_ARG_FLAG_NUM)) {
7112                if ((*iter == 'o') && (len < buf_size)) {
7113                    buf[len++] = '0';
7114                } else if ((*iter == 'x') && ((len+1) < buf_size)) {
7115                    buf[len++] = '0';
7116                    buf[len++] = 'x';
7117                } else if ((*iter == 'X') && ((len+1) < buf_size)) {
7118                    buf[len++] = '0';
7119                    buf[len++] = 'X';
7120                }
```

```
7121             }
7122             while (precision && (num_print > num_len) && (len < buf_size)) {
7123                 buf[len++] = '0';
7124                 num_print--;
7125             }
7126
7127             /* reverse number direction */
7128             while (num_len > 0) {
7129                 if (precision && (len < buf_size))
7130                     buf[len++] = number_buffer[num_len-1];
7131                 num_len--;
7132             }
7133
7134             /* fill right padding up to width characters */
7135             if (flag & NK_ARG_FLAG_LEFT) {
7136                 while ((padding-- > 0) && (len < buf_size))
7137                     buf[len++] = ' ';
7138             }
7139         } else if (*iter == 'f') {
7140             /* floating point */
7141             const char *num_iter;
7142             int cur_precision = (precision < 0) ? 6: precision;
7143             int prefix, cur_width = NK_MAX(width, 0);
7144             double value = va_arg(args, double);
7145             int num_len = 0, frac_len = 0, dot = 0;
7146             int padding = 0;
7147
7148             NK_ASSERT(arg_type == NK_ARG_TYPE_DEFAULT);
7149             NK_DTOA(number_buffer, value);
7150             num_len = nk_strlen(number_buffer);
7151
7152             /* calculate padding */
7153             num_iter = number_buffer;
7154             while (*num_iter && *num_iter != '.')
7155                 num_iter++;
7156
7157             prefix = (*num_iter == '.')?(int)(num_iter - number_buffer)+1:0;
7158             padding = NK_MAX(cur_width - (prefix + NK_MIN(cur_precision, num_len - prefix)) , 0);
7159             if ((flag & NK_ARG_FLAG_PLUS) || (flag & NK_ARG_FLAG_SPACE))
7160                 padding = NK_MAX(padding-1, 0);
7161
7162             /* fill left padding up to a total of `width` characters */
7163             if (!(flag & NK_ARG_FLAG_LEFT)) {
7164                 while (padding-- > 0 && (len < buf_size)) {
7165                     if (flag & NK_ARG_FLAG_ZERO)
7166                         buf[len++] = '0';
7167                     else buf[len++] = ' ';
7168                 }
7169             }
7170
7171             /* copy string value representation into buffer */
7172             num_iter = number_buffer;
7173             if ((flag & NK_ARG_FLAG_PLUS) && (value >= 0) && (len < buf_size))
7174                 buf[len++] = '+';
7175             else if ((flag & NK_ARG_FLAG_SPACE) && (value >= 0) && (len < buf_size))
7176                 buf[len++] = ' ';
7177             while (*num_iter) {
7178                 if (dot) frac_len++;
7179                 if (len < buf_size)
7180                     buf[len++] = *num_iter;
7181                 if (*num_iter == '.') dot = 1;
7182                 if (frac_len >= cur_precision) break;
7183                 num_iter++;
7184             }
7185
7186             /* fill number up to precision */
7187             while (frac_len < cur_precision) {
7188                 if (!dot && len < buf_size) {
7189                     buf[len++] = '.';
7190                     dot = 1;
7191                 }
7192                 if (len < buf_size)
7193                     buf[len++] = '0';
7194                 frac_len++;
7195             }
7196
7197             /* fill right padding up to width characters */
7198             if (flag & NK_ARG_FLAG_LEFT) {
7199                 while ((padding-- > 0) && (len < buf_size))
7200                     buf[len++] = ' ';
7201             }
7202         } else {
7203             /* Specifier not supported: g,G,e,E,p,z */
7204             NK_ASSERT(0 && "specifier is not supported!");
7205             return result;
7206         }
7207     }
```

```
7208     buf[(len >= buf_size)?(buf_size-1):len] = 0;
7209     result = (len >= buf_size)?-1:len;
7210     return result;
7211 }
7212 #endif
7213 NK_LIB int
7214 nk_strfmt(char *buf, int buf_size, const char *fmt, va_list args)
7215 {
7216     int result = -1;
7217     NK_ASSERT(buf);
7218     NK_ASSERT(buf_size);
7219     if (!buf || !buf_size || !fmt) return 0;
7220 #ifdef NK_INCLUDE_STANDARD_IO
7221     result = NK_VSNPRINTF(buf, (nk_size)buf_size, fmt, args);
7222     result = (result >= buf_size) ? -1: result;
7223     buf[buf_size-1] = 0;
7224 #else
7225     result = nk_vsnprintf(buf, buf_size, fmt, args);
7226 #endif
7227     return result;
7228 }
7229 #endif
7230 NK_API nk_hash
7231 nk_murmur_hash(const void * key, int len, nk_hash seed)
7232 {
7233     /* 32-Bit MurmurHash3: https://code.google.com/p/smhasher/wiki/MurmurHash3*/
7234     #define NK_ROTL(x,r) ((x) « (r) | ((x) » (32 - r)))
7235
7236     nk_uint h1 = seed;
7237     nk_uint k1;
7238     const nk_byte *data = (const nk_byte*)key;
7239     const nk_byte *keyptr = data;
7240     nk_byte *k1ptr;
7241     const int bsize = sizeof(k1);
7242     const int nblocks = len/4;
7243
7244     const nk_uint c1 = 0xcc9e2d51;
7245     const nk_uint c2 = 0x1b873593;
7246     const nk_byte *tail;
7247     int i;
7248
7249     /* body */
7250     if (!key) return 0;
7251     for (i = 0; i < nblocks; ++i, keyptr += bsize) {
7252         k1ptr = (nk_byte*)&k1;
7253         k1ptr[0] = keyptr[0];
7254         k1ptr[1] = keyptr[1];
7255         k1ptr[2] = keyptr[2];
7256         k1ptr[3] = keyptr[3];
7257
7258         k1 *= c1;
7259         k1 = NK_ROTL(k1,15);
7260         k1 *= c2;
7261
7262         h1 ^= k1;
7263         h1 = NK_ROTL(h1,13);
7264         h1 = h1*5+0xe6546b64;
7265     }
7266
7267     /* tail */
7268     tail = (const nk_byte*)(data + nblocks*4);
7269     k1 = 0;
7270     switch (len & 3) {
7271         case 3: k1 ^= (nk_uint)(tail[2] « 16); /* fallthrough */
7272         case 2: k1 ^= (nk_uint)(tail[1] « 8u); /* fallthrough */
7273         case 1: k1 ^= tail[0];
7274             k1 *= c1;
7275             k1 = NK_ROTL(k1,15);
7276             k1 *= c2;
7277             h1 ^= k1;
7278             break;
7279         default: break;
7280     }
7281
7282     /* finalization */
7283     h1 ^= (nk_uint)len;
7284     /* fmix32 */
7285     h1 ^= h1 » 16;
7286     h1 *= 0x85ebca6b;
7287     h1 ^= h1 » 13;
7288     h1 *= 0xc2b2ae35;
7289     h1 ^= h1 » 16;
7290
7291     #undef NK_ROTL
7292     return h1;
7293 }
7294 #ifdef NK_INCLUDE_STANDARD_IO
```

```
7295 NK_LIB char*
7296 nk_file_load(const char* path, nk_size* siz, struct nk_allocator *alloc)
7297 {
7298     char *buf;
7299     FILE *fd;
7300     long ret;
7301
7302     NK_ASSERT(path);
7303     NK_ASSERT(siz);
7304     NK_ASSERT(alloc);
7305     if (!path || !siz || !alloc)
7306         return 0;
7307
7308     fd = fopen(path, "rb");
7309     if (!fd) return 0;
7310     fseek(fd, 0, SEEK_END);
7311     ret = ftell(fd);
7312     if (ret < 0) {
7313         fclose(fd);
7314         return 0;
7315     }
7316     *siz = (nk_size)ret;
7317     fseek(fd, 0, SEEK_SET);
7318     buf = (char*)alloc->alloc(alloc->userdata,0, *siz);
7319     NK_ASSERT(buf);
7320     if (!buf) {
7321         fclose(fd);
7322         return 0;
7323     }
7324     *siz = (nk_size)fread(buf, 1,*siz, fd);
7325     fclose(fd);
7326     return buf;
7327 }
7328 #endif
7329 NK_LIB int
7330 nk_text_clamp(const struct nk_user_font *font, const char *text,
7331     int text_len, float space, int *glyphs, float *text_width,
7332     nk_rune *sep_list, int sep_count)
7333 {
7334     int i = 0;
7335     int glyph_len = 0;
7336     float last_width = 0;
7337     nk_rune unicode = 0;
7338     float width = 0;
7339     int len = 0;
7340     int g = 0;
7341     float s;
7342
7343     int sep_len = 0;
7344     int sep_g = 0;
7345     float sep_width = 0;
7346     sep_count = NK_MAX(sep_count,0);
7347
7348     glyph_len = nk_utf_decode(text, &unicode, text_len);
7349     while (glyph_len && (width < space) && (len < text_len)) {
7350         len += glyph_len;
7351         s = font->width(font->userdata, font->height, text, len);
7352         for (i = 0; i < sep_count; ++i) {
7353             if (unicode != sep_list[i]) continue;
7354             sep_width = last_width = width;
7355             sep_g = g+1;
7356             sep_len = len;
7357             break;
7358         }
7359         if (i == sep_count){
7360             last_width = sep_width = width;
7361             sep_g = g+1;
7362         }
7363         width = s;
7364         glyph_len = nk_utf_decode(&text[len], &unicode, text_len - len);
7365         g++;
7366     }
7367     if (len >= text_len) {
7368         *glyphs = g;
7369         *text_width = last_width;
7370         return len;
7371     } else {
7372         *glyphs = sep_g;
7373         *text_width = sep_width;
7374         return (!sep_len) ? len: sep_len;
7375     }
7376 }
7377 NK_LIB struct nk_vec2
7378 nk_text_calculate_text_bounds(const struct nk_user_font *font,
7379     const char *begin, int byte_len, float row_height, const char **remaining,
7380     struct nk_vec2 *out_offset, int *glyphs, int op)
7381 {
```

```
7382     float line_height = row_height;
7383     struct nk_vec2 text_size = nk_vec2(0,0);
7384     float line_width = 0.0f;
7385
7386     float glyph_width;
7387     int glyph_len = 0;
7388     nk_rune unicode = 0;
7389     int text_len = 0;
7390     if (!begin || byte_len <= 0 || !font)
7391         return nk_vec2(0,row_height);
7392
7393     glyph_len = nk_utf_decode(begin, &unicode, byte_len);
7394     if (!glyph_len) return text_size;
7395     glyph_width = font->width(font->userdata, font->height, begin, glyph_len);
7396
7397     *glyphs = 0;
7398     while ((text_len < byte_len) && glyph_len) {
7399         if (unicode == '\n') {
7400             text_size.x = NK_MAX(text_size.x, line_width);
7401             text_size.y += line_height;
7402             line_width = 0;
7403             *glyphs+=1;
7404             if (op == NK_STOP_ON_NEW_LINE)
7405                 break;
7406
7407             text_len++;
7408             glyph_len = nk_utf_decode(begin + text_len, &unicode, byte_len-text_len);
7409             continue;
7410         }
7411
7412         if (unicode == '\r') {
7413             text_len++;
7414             *glyphs+=1;
7415             glyph_len = nk_utf_decode(begin + text_len, &unicode, byte_len-text_len);
7416             continue;
7417         }
7418
7419         *glyphs = *glyphs + 1;
7420         text_len += glyph_len;
7421         line_width += (float)glyph_width;
7422         glyph_len = nk_utf_decode(begin + text_len, &unicode, byte_len-text_len);
7423         glyph_width = font->width(font->userdata, font->height, begin+text_len, glyph_len);
7424         continue;
7425     }
7426
7427     if (text_size.x < line_width)
7428         text_size.x = line_width;
7429     if (out_offset)
7430         *out_offset = nk_vec2(line_width, text_size.y + line_height);
7431     if (line_width > 0 || text_size.y == 0.0f)
7432         text_size.y += line_height;
7433     if (remaining)
7434         *remaining = begin+text_len;
7435     return text_size;
7436 }
7437
7438
7439
7440
7441
7442 /* ===============================================================
7443  *
7444  *                          COLOR
7445  *
7446  * ===============================================================*/
7447 NK_INTERN int
7448 nk_parse_hex(const char *p, int length)
7449 {
7450     int i = 0;
7451     int len = 0;
7452     while (len < length) {
7453         i <<= 4;
7454         if (p[len] >= 'a' && p[len] <= 'f')
7455             i += ((p[len] - 'a') + 10);
7456         else if (p[len] >= 'A' && p[len] <= 'F')
7457             i += ((p[len] - 'A') + 10);
7458         else i += (p[len] - '0');
7459         len++;
7460     }
7461     return i;
7462 }
7463 NK_API struct nk_color
7464 nk_rgba(int r, int g, int b, int a)
7465 {
7466     struct nk_color ret;
7467     ret.r = (nk_byte)NK_CLAMP(0, r, 255);
7468     ret.g = (nk_byte)NK_CLAMP(0, g, 255);
```

```
7469     ret.b = (nk_byte)NK_CLAMP(0, b, 255);
7470     ret.a = (nk_byte)NK_CLAMP(0, a, 255);
7471     return ret;
7472 }
7473 NK_API struct nk_color
7474 nk_rgb_hex(const char *rgb)
7475 {
7476     struct nk_color col;
7477     const char *c = rgb;
7478     if (*c == '#') c++;
7479     col.r = (nk_byte)nk_parse_hex(c, 2);
7480     col.g = (nk_byte)nk_parse_hex(c+2, 2);
7481     col.b = (nk_byte)nk_parse_hex(c+4, 2);
7482     col.a = 255;
7483     return col;
7484 }
7485 NK_API struct nk_color
7486 nk_rgba_hex(const char *rgb)
7487 {
7488     struct nk_color col;
7489     const char *c = rgb;
7490     if (*c == '#') c++;
7491     col.r = (nk_byte)nk_parse_hex(c, 2);
7492     col.g = (nk_byte)nk_parse_hex(c+2, 2);
7493     col.b = (nk_byte)nk_parse_hex(c+4, 2);
7494     col.a = (nk_byte)nk_parse_hex(c+6, 2);
7495     return col;
7496 }
7497 NK_API void
7498 nk_color_hex_rgba(char *output, struct nk_color col)
7499 {
7500     #define NK_TO_HEX(i) ((i) <= 9 ? '0' + (i): 'A' - 10 + (i))
7501     output[0] = (char)NK_TO_HEX((col.r & 0xF0) » 4);
7502     output[1] = (char)NK_TO_HEX((col.r & 0x0F));
7503     output[2] = (char)NK_TO_HEX((col.g & 0xF0) » 4);
7504     output[3] = (char)NK_TO_HEX((col.g & 0x0F));
7505     output[4] = (char)NK_TO_HEX((col.b & 0xF0) » 4);
7506     output[5] = (char)NK_TO_HEX((col.b & 0x0F));
7507     output[6] = (char)NK_TO_HEX((col.a & 0xF0) » 4);
7508     output[7] = (char)NK_TO_HEX((col.a & 0x0F));
7509     output[8] = '\0';
7510     #undef NK_TO_HEX
7511 }
7512 NK_API void
7513 nk_color_hex_rgb(char *output, struct nk_color col)
7514 {
7515     #define NK_TO_HEX(i) ((i) <= 9 ? '0' + (i): 'A' - 10 + (i))
7516     output[0] = (char)NK_TO_HEX((col.r & 0xF0) » 4);
7517     output[1] = (char)NK_TO_HEX((col.r & 0x0F));
7518     output[2] = (char)NK_TO_HEX((col.g & 0xF0) » 4);
7519     output[3] = (char)NK_TO_HEX((col.g & 0x0F));
7520     output[4] = (char)NK_TO_HEX((col.b & 0xF0) » 4);
7521     output[5] = (char)NK_TO_HEX((col.b & 0x0F));
7522     output[6] = '\0';
7523     #undef NK_TO_HEX
7524 }
7525 NK_API struct nk_color
7526 nk_rgba_iv(const int *c)
7527 {
7528     return nk_rgba(c[0], c[1], c[2], c[3]);
7529 }
7530 NK_API struct nk_color
7531 nk_rgba_bv(const nk_byte *c)
7532 {
7533     return nk_rgba(c[0], c[1], c[2], c[3]);
7534 }
7535 NK_API struct nk_color
7536 nk_rgb(int r, int g, int b)
7537 {
7538     struct nk_color ret;
7539     ret.r = (nk_byte)NK_CLAMP(0, r, 255);
7540     ret.g = (nk_byte)NK_CLAMP(0, g, 255);
7541     ret.b = (nk_byte)NK_CLAMP(0, b, 255);
7542     ret.a = (nk_byte)255;
7543     return ret;
7544 }
7545 NK_API struct nk_color
7546 nk_rgb_iv(const int *c)
7547 {
7548     return nk_rgb(c[0], c[1], c[2]);
7549 }
7550 NK_API struct nk_color
7551 nk_rgb_bv(const nk_byte* c)
7552 {
7553     return nk_rgb(c[0], c[1], c[2]);
7554 }
7555 NK_API struct nk_color
```

```
7556 nk_rgba_u32(nk_uint in)
7557 {
7558     struct nk_color ret;
7559     ret.r = (in & 0xFF);
7560     ret.g = ((in » 8) & 0xFF);
7561     ret.b = ((in » 16) & 0xFF);
7562     ret.a = (nk_byte)((in » 24) & 0xFF);
7563     return ret;
7564 }
7565 NK_API struct nk_color
7566 nk_rgba_f(float r, float g, float b, float a)
7567 {
7568     struct nk_color ret;
7569     ret.r = (nk_byte)(NK_SATURATE(r) * 255.0f);
7570     ret.g = (nk_byte)(NK_SATURATE(g) * 255.0f);
7571     ret.b = (nk_byte)(NK_SATURATE(b) * 255.0f);
7572     ret.a = (nk_byte)(NK_SATURATE(a) * 255.0f);
7573     return ret;
7574 }
7575 NK_API struct nk_color
7576 nk_rgba_fv(const float *c)
7577 {
7578     return nk_rgba_f(c[0], c[1], c[2], c[3]);
7579 }
7580 NK_API struct nk_color
7581 nk_rgba_cf(struct nk_colorf c)
7582 {
7583     return nk_rgba_f(c.r, c.g, c.b, c.a);
7584 }
7585 NK_API struct nk_color
7586 nk_rgb_f(float r, float g, float b)
7587 {
7588     struct nk_color ret;
7589     ret.r = (nk_byte)(NK_SATURATE(r) * 255.0f);
7590     ret.g = (nk_byte)(NK_SATURATE(g) * 255.0f);
7591     ret.b = (nk_byte)(NK_SATURATE(b) * 255.0f);
7592     ret.a = 255;
7593     return ret;
7594 }
7595 NK_API struct nk_color
7596 nk_rgb_fv(const float *c)
7597 {
7598     return nk_rgb_f(c[0], c[1], c[2]);
7599 }
7600 NK_API struct nk_color
7601 nk_rgb_cf(struct nk_colorf c)
7602 {
7603     return nk_rgb_f(c.r, c.g, c.b);
7604 }
7605 NK_API struct nk_color
7606 nk_hsv(int h, int s, int v)
7607 {
7608     return nk_hsva(h, s, v, 255);
7609 }
7610 NK_API struct nk_color
7611 nk_hsv_iv(const int *c)
7612 {
7613     return nk_hsv(c[0], c[1], c[2]);
7614 }
7615 NK_API struct nk_color
7616 nk_hsv_bv(const nk_byte *c)
7617 {
7618     return nk_hsv(c[0], c[1], c[2]);
7619 }
7620 NK_API struct nk_color
7621 nk_hsv_f(float h, float s, float v)
7622 {
7623     return nk_hsva_f(h, s, v, 1.0f);
7624 }
7625 NK_API struct nk_color
7626 nk_hsv_fv(const float *c)
7627 {
7628     return nk_hsv_f(c[0], c[1], c[2]);
7629 }
7630 NK_API struct nk_color
7631 nk_hsva(int h, int s, int v, int a)
7632 {
7633     float hf = ((float)NK_CLAMP(0, h, 255)) / 255.0f;
7634     float sf = ((float)NK_CLAMP(0, s, 255)) / 255.0f;
7635     float vf = ((float)NK_CLAMP(0, v, 255)) / 255.0f;
7636     float af = ((float)NK_CLAMP(0, a, 255)) / 255.0f;
7637     return nk_hsva_f(hf, sf, vf, af);
7638 }
7639 NK_API struct nk_color
7640 nk_hsva_iv(const int *c)
7641 {
7642     return nk_hsva(c[0], c[1], c[2], c[3]);
```

```
7643 }
7644 NK_API struct nk_color
7645 nk_hsva_bv(const nk_byte *c)
7646 {
7647     return nk_hsva(c[0], c[1], c[2], c[3]);
7648 }
7649 NK_API struct nk_colorf
7650 nk_hsva_colorf(float h, float s, float v, float a)
7651 {
7652     int i;
7653     float p, q, t, f;
7654     struct nk_colorf out = {0,0,0,0};
7655     if (s <= 0.0f) {
7656         out.r = v; out.g = v; out.b = v; out.a = a;
7657         return out;
7658     }
7659     h = h / (60.0f/360.0f);
7660     i = (int)h;
7661     f = h - (float)i;
7662     p = v * (1.0f - s);
7663     q = v * (1.0f - (s * f));
7664     t = v * (1.0f - s * (1.0f - f));
7665
7666     switch (i) {
7667     case 0: default: out.r = v; out.g = t; out.b = p; break;
7668     case 1: out.r = q; out.g = v; out.b = p; break;
7669     case 2: out.r = p; out.g = v; out.b = t; break;
7670     case 3: out.r = p; out.g = q; out.b = v; break;
7671     case 4: out.r = t; out.g = p; out.b = v; break;
7672     case 5: out.r = v; out.g = p; out.b = q; break;}
7673     out.a = a;
7674     return out;
7675 }
7676 NK_API struct nk_colorf
7677 nk_hsva_colorfv(float *c)
7678 {
7679     return nk_hsva_colorf(c[0], c[1], c[2], c[3]);
7680 }
7681 NK_API struct nk_color
7682 nk_hsva_f(float h, float s, float v, float a)
7683 {
7684     struct nk_colorf c = nk_hsva_colorf(h, s, v, a);
7685     return nk_rgba_f(c.r, c.g, c.b, c.a);
7686 }
7687 NK_API struct nk_color
7688 nk_hsva_fv(const float *c)
7689 {
7690     return nk_hsva_f(c[0], c[1], c[2], c[3]);
7691 }
7692 NK_API nk_uint
7693 nk_color_u32(struct nk_color in)
7694 {
7695     nk_uint out = (nk_uint)in.r;
7696     out |= ((nk_uint)in.g « 8);
7697     out |= ((nk_uint)in.b « 16);
7698     out |= ((nk_uint)in.a « 24);
7699     return out;
7700 }
7701 NK_API void
7702 nk_color_f(float *r, float *g, float *b, float *a, struct nk_color in)
7703 {
7704     NK_STORAGE const float s = 1.0f/255.0f;
7705     *r = (float)in.r * s;
7706     *g = (float)in.g * s;
7707     *b = (float)in.b * s;
7708     *a = (float)in.a * s;
7709 }
7710 NK_API void
7711 nk_color_fv(float *c, struct nk_color in)
7712 {
7713     nk_color_f(&c[0], &c[1], &c[2], &c[3], in);
7714 }
7715 NK_API struct nk_colorf
7716 nk_color_cf(struct nk_color in)
7717 {
7718     struct nk_colorf o;
7719     nk_color_f(&o.r, &o.g, &o.b, &o.a, in);
7720     return o;
7721 }
7722 NK_API void
7723 nk_color_d(double *r, double *g, double *b, double *a, struct nk_color in)
7724 {
7725     NK_STORAGE const double s = 1.0/255.0;
7726     *r = (double)in.r * s;
7727     *g = (double)in.g * s;
7728     *b = (double)in.b * s;
7729     *a = (double)in.a * s;
```

```
7730 }
7731 NK_API void
7732 nk_color_dv(double *c, struct nk_color in)
7733 {
7734     nk_color_d(&c[0], &c[1], &c[2], &c[3], in);
7735 }
7736 NK_API void
7737 nk_color_hsv_f(float *out_h, float *out_s, float *out_v, struct nk_color in)
7738 {
7739     float a;
7740     nk_color_hsva_f(out_h, out_s, out_v, &a, in);
7741 }
7742 NK_API void
7743 nk_color_hsv_fv(float *out, struct nk_color in)
7744 {
7745     float a;
7746     nk_color_hsva_f(&out[0], &out[1], &out[2], &a, in);
7747 }
7748 NK_API void
7749 nk_colorf_hsva_f(float *out_h, float *out_s,
7750     float *out_v, float *out_a, struct nk_colorf in)
7751 {
7752     float chroma;
7753     float K = 0.0f;
7754     if (in.g < in.b) {
7755         const float t = in.g; in.g = in.b; in.b = t;
7756         K = -1.f;
7757     }
7758     if (in.r < in.g) {
7759         const float t = in.r; in.r = in.g; in.g = t;
7760         K = -2.f/6.0f - K;
7761     }
7762     chroma = in.r - ((in.g < in.b) ? in.g: in.b);
7763     *out_h = NK_ABS(K + (in.g - in.b)/(6.0f * chroma + 1e-20f));
7764     *out_s = chroma / (in.r + 1e-20f);
7765     *out_v = in.r;
7766     *out_a = in.a;
7767
7768 }
7769 NK_API void
7770 nk_colorf_hsva_fv(float *hsva, struct nk_colorf in)
7771 {
7772     nk_colorf_hsva_f(&hsva[0], &hsva[1], &hsva[2], &hsva[3], in);
7773 }
7774 NK_API void
7775 nk_color_hsva_f(float *out_h, float *out_s,
7776     float *out_v, float *out_a, struct nk_color in)
7777 {
7778     struct nk_colorf col;
7779     nk_color_f(&col.r,&col.g,&col.b,&col.a, in);
7780     nk_colorf_hsva_f(out_h, out_s, out_v, out_a, col);
7781 }
7782 NK_API void
7783 nk_color_hsva_fv(float *out, struct nk_color in)
7784 {
7785     nk_color_hsva_f(&out[0], &out[1], &out[2], &out[3], in);
7786 }
7787 NK_API void
7788 nk_color_hsva_i(int *out_h, int *out_s, int *out_v,
7789                 int *out_a, struct nk_color in)
7790 {
7791     float h,s,v,a;
7792     nk_color_hsva_f(&h, &s, &v, &a, in);
7793     *out_h = (nk_byte)(h * 255.0f);
7794     *out_s = (nk_byte)(s * 255.0f);
7795     *out_v = (nk_byte)(v * 255.0f);
7796     *out_a = (nk_byte)(a * 255.0f);
7797 }
7798 NK_API void
7799 nk_color_hsva_iv(int *out, struct nk_color in)
7800 {
7801     nk_color_hsva_i(&out[0], &out[1], &out[2], &out[3], in);
7802 }
7803 NK_API void
7804 nk_color_hsva_bv(nk_byte *out, struct nk_color in)
7805 {
7806     int tmp[4];
7807     nk_color_hsva_i(&tmp[0], &tmp[1], &tmp[2], &tmp[3], in);
7808     out[0] = (nk_byte)tmp[0];
7809     out[1] = (nk_byte)tmp[1];
7810     out[2] = (nk_byte)tmp[2];
7811     out[3] = (nk_byte)tmp[3];
7812 }
7813 NK_API void
7814 nk_color_hsva_b(nk_byte *h, nk_byte *s, nk_byte *v, nk_byte *a, struct nk_color in)
7815 {
7816     int tmp[4];
```

```
7817     nk_color_hsva_i(&tmp[0], &tmp[1], &tmp[2], &tmp[3], in);
7818     *h = (nk_byte)tmp[0];
7819     *s = (nk_byte)tmp[1];
7820     *v = (nk_byte)tmp[2];
7821     *a = (nk_byte)tmp[3];
7822 }
7823 NK_API void
7824 nk_color_hsv_i(int *out_h, int *out_s, int *out_v, struct nk_color in)
7825 {
7826     int a;
7827     nk_color_hsva_i(out_h, out_s, out_v, &a, in);
7828 }
7829 NK_API void
7830 nk_color_hsv_b(nk_byte *out_h, nk_byte *out_s, nk_byte *out_v, struct nk_color in)
7831 {
7832     int tmp[4];
7833     nk_color_hsva_i(&tmp[0], &tmp[1], &tmp[2], &tmp[3], in);
7834     *out_h = (nk_byte)tmp[0];
7835     *out_s = (nk_byte)tmp[1];
7836     *out_v = (nk_byte)tmp[2];
7837 }
7838 NK_API void
7839 nk_color_hsv_iv(int *out, struct nk_color in)
7840 {
7841     nk_color_hsv_i(&out[0], &out[1], &out[2], in);
7842 }
7843 NK_API void
7844 nk_color_hsv_bv(nk_byte *out, struct nk_color in)
7845 {
7846     int tmp[4];
7847     nk_color_hsv_i(&tmp[0], &tmp[1], &tmp[2], in);
7848     out[0] = (nk_byte)tmp[0];
7849     out[1] = (nk_byte)tmp[1];
7850     out[2] = (nk_byte)tmp[2];
7851 }
7852
7853
7854
7855
7856
7857 /* ===============================================================
7858  *
7859  *                              UTF-8
7860  *
7861  * ===============================================================*/
7862 NK_GLOBAL const nk_byte nk_utfbyte[NK_UTF_SIZE+1] = {0x80, 0, 0xC0, 0xE0, 0xF0};
7863 NK_GLOBAL const nk_byte nk_utfmask[NK_UTF_SIZE+1] = {0xC0, 0x80, 0xE0, 0xF0, 0xF8};
7864 NK_GLOBAL const nk_uint nk_utfmin[NK_UTF_SIZE+1] = {0, 0, 0x80, 0x800, 0x10000};
7865 NK_GLOBAL const nk_uint nk_utfmax[NK_UTF_SIZE+1] = {0x10FFFF, 0x7F, 0x7FF, 0xFFFF, 0x10FFFF};
7866
7867 NK_INTERN int
7868 nk_utf_validate(nk_rune *u, int i)
7869 {
7870     NK_ASSERT(u);
7871     if (!u) return 0;
7872     if (!NK_BETWEEN(*u, nk_utfmin[i], nk_utfmax[i]) ||
7873          NK_BETWEEN(*u, 0xD800, 0xDFFF))
7874             *u = NK_UTF_INVALID;
7875     for (i = 1; *u > nk_utfmax[i]; ++i);
7876     return i;
7877 }
7878 NK_INTERN nk_rune
7879 nk_utf_decode_byte(char c, int *i)
7880 {
7881     NK_ASSERT(i);
7882     if (!i) return 0;
7883     for(*i = 0; *i < (int)NK_LEN(nk_utfmask); ++(*i)) {
7884         if (((nk_byte)c & nk_utfmask[*i]) == nk_utfbyte[*i])
7885             return (nk_byte)(c & ~nk_utfmask[*i]);
7886     }
7887     return 0;
7888 }
7889 NK_API int
7890 nk_utf_decode(const char *c, nk_rune *u, int clen)
7891 {
7892     int i, j, len, type=0;
7893     nk_rune udecoded;
7894
7895     NK_ASSERT(c);
7896     NK_ASSERT(u);
7897
7898     if (!c || !u) return 0;
7899     if (!clen) return 0;
7900     *u = NK_UTF_INVALID;
7901
7902     udecoded = nk_utf_decode_byte(c[0], &len);
7903     if (!NK_BETWEEN(len, 1, NK_UTF_SIZE))
```

```
7904         return 1;
7905
7906     for (i = 1, j = 1; i < clen && j < len; ++i, ++j) {
7907         udecoded = (udecoded « 6) | nk_utf_decode_byte(c[i], &type);
7908         if (type != 0)
7909             return j;
7910     }
7911     if (j < len)
7912         return 0;
7913     *u = udecoded;
7914     nk_utf_validate(u, len);
7915     return len;
7916 }
7917 NK_INTERN char
7918 nk_utf_encode_byte(nk_rune u, int i)
7919 {
7920     return (char)((nk_utfbyte[i]) | ((nk_byte)u & ~nk_utfmask[i]));
7921 }
7922 NK_API int
7923 nk_utf_encode(nk_rune u, char *c, int clen)
7924 {
7925     int len, i;
7926     len = nk_utf_validate(&u, 0);
7927     if (clen < len || !len || len > NK_UTF_SIZE)
7928         return 0;
7929
7930     for (i = len - 1; i != 0; --i) {
7931         c[i] = nk_utf_encode_byte(u, 0);
7932         u »= 6;
7933     }
7934     c[0] = nk_utf_encode_byte(u, len);
7935     return len;
7936 }
7937 NK_API int
7938 nk_utf_len(const char *str, int len)
7939 {
7940     const char *text;
7941     int glyphs = 0;
7942     int text_len;
7943     int glyph_len;
7944     int src_len = 0;
7945     nk_rune unicode;
7946
7947     NK_ASSERT(str);
7948     if (!str || !len) return 0;
7949
7950     text = str;
7951     text_len = len;
7952     glyph_len = nk_utf_decode(text, &unicode, text_len);
7953     while (glyph_len && src_len < len) {
7954         glyphs++;
7955         src_len = src_len + glyph_len;
7956         glyph_len = nk_utf_decode(text + src_len, &unicode, text_len - src_len);
7957     }
7958     return glyphs;
7959 }
7960 NK_API const char*
7961 nk_utf_at(const char *buffer, int length, int index,
7962     nk_rune *unicode, int *len)
7963 {
7964     int i = 0;
7965     int src_len = 0;
7966     int glyph_len = 0;
7967     const char *text;
7968     int text_len;
7969
7970     NK_ASSERT(buffer);
7971     NK_ASSERT(unicode);
7972     NK_ASSERT(len);
7973
7974     if (!buffer || !unicode || !len) return 0;
7975     if (index < 0) {
7976         *unicode = NK_UTF_INVALID;
7977         *len = 0;
7978         return 0;
7979     }
7980
7981     text = buffer;
7982     text_len = length;
7983     glyph_len = nk_utf_decode(text, unicode, text_len);
7984     while (glyph_len) {
7985         if (i == index) {
7986             *len = glyph_len;
7987             break;
7988         }
7989
7990         i++;
```

```
7991         src_len = src_len + glyph_len;
7992         glyph_len = nk_utf_decode(text + src_len, unicode, text_len - src_len);
7993     }
7994     if (i != index) return 0;
7995     return buffer + src_len;
7996 }
7997
7998
7999
8000
8001
8002 /* ===============================================================
8003  *
8004  *                          BUFFER
8005  *
8006  * ===============================================================*/
8007 #ifdef NK_INCLUDE_DEFAULT_ALLOCATOR
8008 NK_LIB void*
8009 nk_malloc(nk_handle unused, void *old,nk_size size)
8010 {
8011     NK_UNUSED(unused);
8012     NK_UNUSED(old);
8013     return malloc(size);
8014 }
8015 NK_LIB void
8016 nk_mfree(nk_handle unused, void *ptr)
8017 {
8018     NK_UNUSED(unused);
8019     free(ptr);
8020 }
8021 NK_API void
8022 nk_buffer_init_default(struct nk_buffer *buffer)
8023 {
8024     struct nk_allocator alloc;
8025     alloc.userdata.ptr = 0;
8026     alloc.alloc = nk_malloc;
8027     alloc.free = nk_mfree;
8028     nk_buffer_init(buffer, &alloc, NK_BUFFER_DEFAULT_INITIAL_SIZE);
8029 }
8030 #endif
8031
8032 NK_API void
8033 nk_buffer_init(struct nk_buffer *b, const struct nk_allocator *a,
8034     nk_size initial_size)
8035 {
8036     NK_ASSERT(b);
8037     NK_ASSERT(a);
8038     NK_ASSERT(initial_size);
8039     if (!b || !a || !initial_size) return;
8040
8041     nk_zero(b, sizeof(*b));
8042     b->type = NK_BUFFER_DYNAMIC;
8043     b->memory.ptr = a->alloc(a->userdata,0, initial_size);
8044     b->memory.size = initial_size;
8045     b->size = initial_size;
8046     b->grow_factor = 2.0f;
8047     b->pool = *a;
8048 }
8049 NK_API void
8050 nk_buffer_init_fixed(struct nk_buffer *b, void *m, nk_size size)
8051 {
8052     NK_ASSERT(b);
8053     NK_ASSERT(m);
8054     NK_ASSERT(size);
8055     if (!b || !m || !size) return;
8056
8057     nk_zero(b, sizeof(*b));
8058     b->type = NK_BUFFER_FIXED;
8059     b->memory.ptr = m;
8060     b->memory.size = size;
8061     b->size = size;
8062 }
8063 NK_LIB void*
8064 nk_buffer_align(void *unaligned,
8065     nk_size align, nk_size *alignment,
8066     enum nk_buffer_allocation_type type)
8067 {
8068     void *memory = 0;
8069     switch (type) {
8070     default:
8071     case NK_BUFFER_MAX:
8072     case NK_BUFFER_FRONT:
8073         if (align) {
8074             memory = NK_ALIGN_PTR(unaligned, align);
8075             *alignment = (nk_size)((nk_byte*)memory - (nk_byte*)unaligned);
8076         } else {
8077             memory = unaligned;
```

```
8078             *alignment = 0;
8079         }
8080         break;
8081     case NK_BUFFER_BACK:
8082         if (align) {
8083             memory = NK_ALIGN_PTR_BACK(unaligned, align);
8084             *alignment = (nk_size)((nk_byte*)unaligned - (nk_byte*)memory);
8085         } else {
8086             memory = unaligned;
8087             *alignment = 0;
8088         }
8089         break;
8090     }
8091     return memory;
8092 }
8093 NK_LIB void*
8094 nk_buffer_realloc(struct nk_buffer *b, nk_size capacity, nk_size *size)
8095 {
8096     void *temp;
8097     nk_size buffer_size;
8098
8099     NK_ASSERT(b);
8100     NK_ASSERT(size);
8101     if (!b || !size || !b->pool.alloc || !b->pool.free)
8102         return 0;
8103
8104     buffer_size = b->memory.size;
8105     temp = b->pool.alloc(b->pool.userdata, b->memory.ptr, capacity);
8106     NK_ASSERT(temp);
8107     if (!temp) return 0;
8108
8109     *size = capacity;
8110     if (temp != b->memory.ptr) {
8111         NK_MEMCPY(temp, b->memory.ptr, buffer_size);
8112         b->pool.free(b->pool.userdata, b->memory.ptr);
8113     }
8114
8115     if (b->size == buffer_size) {
8116         /* no back buffer so just set correct size */
8117         b->size = capacity;
8118         return temp;
8119     } else {
8120         /* copy back buffer to the end of the new buffer */
8121         void *dst, *src;
8122         nk_size back_size;
8123         back_size = buffer_size - b->size;
8124         dst = nk_ptr_add(void, temp, capacity - back_size);
8125         src = nk_ptr_add(void, temp, b->size);
8126         NK_MEMCPY(dst, src, back_size);
8127         b->size = capacity - back_size;
8128     }
8129     return temp;
8130 }
8131 NK_LIB void*
8132 nk_buffer_alloc(struct nk_buffer *b, enum nk_buffer_allocation_type type,
8133     nk_size size, nk_size align)
8134 {
8135     int full;
8136     nk_size alignment;
8137     void *unaligned;
8138     void *memory;
8139
8140     NK_ASSERT(b);
8141     NK_ASSERT(size);
8142     if (!b || !size) return 0;
8143     b->needed += size;
8144
8145     /* calculate total size with needed alignment + size */
8146     if (type == NK_BUFFER_FRONT)
8147         unaligned = nk_ptr_add(void, b->memory.ptr, b->allocated);
8148     else unaligned = nk_ptr_add(void, b->memory.ptr, b->size - size);
8149     memory = nk_buffer_align(unaligned, align, &alignment, type);
8150
8151     /* check if buffer has enough memory*/
8152     if (type == NK_BUFFER_FRONT)
8153         full = ((b->allocated + size + alignment) > b->size);
8154     else full = ((b->size - NK_MIN(b->size,(size + alignment))) <= b->allocated);
8155
8156     if (full) {
8157         nk_size capacity;
8158         if (b->type != NK_BUFFER_DYNAMIC)
8159             return 0;
8160         NK_ASSERT(b->pool.alloc && b->pool.free);
8161         if (b->type != NK_BUFFER_DYNAMIC || !b->pool.alloc || !b->pool.free)
8162             return 0;
8163
8164         /* buffer is full so allocate bigger buffer if dynamic */
```

```
8165         capacity = (nk_size)((float)b->memory.size * b->grow_factor);
8166         capacity = NK_MAX(capacity, nk_round_up_pow2((nk_uint)(b->allocated + size)));
8167         b->memory.ptr = nk_buffer_realloc(b, capacity, &b->memory.size);
8168         if (!b->memory.ptr) return 0;
8169
8170         /* align newly allocated pointer */
8171         if (type == NK_BUFFER_FRONT)
8172             unaligned = nk_ptr_add(void, b->memory.ptr, b->allocated);
8173         else unaligned = nk_ptr_add(void, b->memory.ptr, b->size - size);
8174         memory = nk_buffer_align(unaligned, align, &alignment, type);
8175     }
8176     if (type == NK_BUFFER_FRONT)
8177         b->allocated += size + alignment;
8178     else b->size -= (size + alignment);
8179     b->needed += alignment;
8180     b->calls++;
8181     return memory;
8182 }
8183 NK_API void
8184 nk_buffer_push(struct nk_buffer *b, enum nk_buffer_allocation_type type,
8185     const void *memory, nk_size size, nk_size align)
8186 {
8187     void *mem = nk_buffer_alloc(b, type, size, align);
8188     if (!mem) return;
8189     NK_MEMCPY(mem, memory, size);
8190 }
8191 NK_API void
8192 nk_buffer_mark(struct nk_buffer *buffer, enum nk_buffer_allocation_type type)
8193 {
8194     NK_ASSERT(buffer);
8195     if (!buffer) return;
8196     buffer->marker[type].active = nk_true;
8197     if (type == NK_BUFFER_BACK)
8198         buffer->marker[type].offset = buffer->size;
8199     else buffer->marker[type].offset = buffer->allocated;
8200 }
8201 NK_API void
8202 nk_buffer_reset(struct nk_buffer *buffer, enum nk_buffer_allocation_type type)
8203 {
8204     NK_ASSERT(buffer);
8205     if (!buffer) return;
8206     if (type == NK_BUFFER_BACK) {
8207         /* reset back buffer either back to marker or empty */
8208         buffer->needed -= (buffer->memory.size - buffer->marker[type].offset);
8209         if (buffer->marker[type].active)
8210             buffer->size = buffer->marker[type].offset;
8211         else buffer->size = buffer->memory.size;
8212         buffer->marker[type].active = nk_false;
8213     } else {
8214         /* reset front buffer either back to back marker or empty */
8215         buffer->needed -= (buffer->allocated - buffer->marker[type].offset);
8216         if (buffer->marker[type].active)
8217             buffer->allocated = buffer->marker[type].offset;
8218         else buffer->allocated = 0;
8219         buffer->marker[type].active = nk_false;
8220     }
8221 }
8222 NK_API void
8223 nk_buffer_clear(struct nk_buffer *b)
8224 {
8225     NK_ASSERT(b);
8226     if (!b) return;
8227     b->allocated = 0;
8228     b->size = b->memory.size;
8229     b->calls = 0;
8230     b->needed = 0;
8231 }
8232 NK_API void
8233 nk_buffer_free(struct nk_buffer *b)
8234 {
8235     NK_ASSERT(b);
8236     if (!b || !b->memory.ptr) return;
8237     if (b->type == NK_BUFFER_FIXED) return;
8238     if (!b->pool.free) return;
8239     NK_ASSERT(b->pool.free);
8240     b->pool.free(b->pool.userdata, b->memory.ptr);
8241 }
8242 NK_API void
8243 nk_buffer_info(struct nk_memory_status *s, struct nk_buffer *b)
8244 {
8245     NK_ASSERT(b);
8246     NK_ASSERT(s);
8247     if (!s || !b) return;
8248     s->allocated = b->allocated;
8249     s->size =  b->memory.size;
8250     s->needed = b->needed;
8251     s->memory = b->memory.ptr;
```

```
8252     s->calls = b->calls;
8253 }
8254 NK_API void*
8255 nk_buffer_memory(struct nk_buffer *buffer)
8256 {
8257     NK_ASSERT(buffer);
8258     if (!buffer) return 0;
8259     return buffer->memory.ptr;
8260 }
8261 NK_API const void*
8262 nk_buffer_memory_const(const struct nk_buffer *buffer)
8263 {
8264     NK_ASSERT(buffer);
8265     if (!buffer) return 0;
8266     return buffer->memory.ptr;
8267 }
8268 NK_API nk_size
8269 nk_buffer_total(struct nk_buffer *buffer)
8270 {
8271     NK_ASSERT(buffer);
8272     if (!buffer) return 0;
8273     return buffer->memory.size;
8274 }
8275
8276
8277
8278
8279
8280 /* ===============================================================
8281  *
8282  *                              STRING
8283  *
8284  * ===============================================================*/
8285 #ifdef NK_INCLUDE_DEFAULT_ALLOCATOR
8286 NK_API void
8287 nk_str_init_default(struct nk_str *str)
8288 {
8289     struct nk_allocator alloc;
8290     alloc.userdata.ptr = 0;
8291     alloc.alloc = nk_malloc;
8292     alloc.free = nk_mfree;
8293     nk_buffer_init(&str->buffer, &alloc, 32);
8294     str->len = 0;
8295 }
8296 #endif
8297
8298 NK_API void
8299 nk_str_init(struct nk_str *str, const struct nk_allocator *alloc, nk_size size)
8300 {
8301     nk_buffer_init(&str->buffer, alloc, size);
8302     str->len = 0;
8303 }
8304 NK_API void
8305 nk_str_init_fixed(struct nk_str *str, void *memory, nk_size size)
8306 {
8307     nk_buffer_init_fixed(&str->buffer, memory, size);
8308     str->len = 0;
8309 }
8310 NK_API int
8311 nk_str_append_text_char(struct nk_str *s, const char *str, int len)
8312 {
8313     char *mem;
8314     NK_ASSERT(s);
8315     NK_ASSERT(str);
8316     if (!s || !str || !len) return 0;
8317     mem = (char*)nk_buffer_alloc(&s->buffer, NK_BUFFER_FRONT, (nk_size)len * sizeof(char), 0);
8318     if (!mem) return 0;
8319     NK_MEMCPY(mem, str, (nk_size)len * sizeof(char));
8320     s->len += nk_utf_len(str, len);
8321     return len;
8322 }
8323 NK_API int
8324 nk_str_append_str_char(struct nk_str *s, const char *str)
8325 {
8326     return nk_str_append_text_char(s, str, nk_strlen(str));
8327 }
8328 NK_API int
8329 nk_str_append_text_utf8(struct nk_str *str, const char *text, int len)
8330 {
8331     int i = 0;
8332     int byte_len = 0;
8333     nk_rune unicode;
8334     if (!str || !text || !len) return 0;
8335     for (i = 0; i < len; ++i)
8336         byte_len += nk_utf_decode(text+byte_len, &unicode, 4);
8337     nk_str_append_text_char(str, text, byte_len);
8338     return len;
```

```
8339 }
8340 NK_API int
8341 nk_str_append_str_utf8(struct nk_str *str, const char *text)
8342 {
8343     int runes = 0;
8344     int byte_len = 0;
8345     int num_runes = 0;
8346     int glyph_len = 0;
8347     nk_rune unicode;
8348     if (!str || !text) return 0;
8349
8350     glyph_len = byte_len = nk_utf_decode(text+byte_len, &unicode, 4);
8351     while (unicode != '\0' && glyph_len) {
8352         glyph_len = nk_utf_decode(text+byte_len, &unicode, 4);
8353         byte_len += glyph_len;
8354         num_runes++;
8355     }
8356     nk_str_append_text_char(str, text, byte_len);
8357     return runes;
8358 }
8359 NK_API int
8360 nk_str_append_text_runes(struct nk_str *str, const nk_rune *text, int len)
8361 {
8362     int i = 0;
8363     int byte_len = 0;
8364     nk_glyph glyph;
8365
8366     NK_ASSERT(str);
8367     if (!str || !text || !len) return 0;
8368     for (i = 0; i < len; ++i) {
8369         byte_len = nk_utf_encode(text[i], glyph, NK_UTF_SIZE);
8370         if (!byte_len) break;
8371         nk_str_append_text_char(str, glyph, byte_len);
8372     }
8373     return len;
8374 }
8375 NK_API int
8376 nk_str_append_str_runes(struct nk_str *str, const nk_rune *runes)
8377 {
8378     int i = 0;
8379     nk_glyph glyph;
8380     int byte_len;
8381     NK_ASSERT(str);
8382     if (!str || !runes) return 0;
8383     while (runes[i] != '\0') {
8384         byte_len = nk_utf_encode(runes[i], glyph, NK_UTF_SIZE);
8385         nk_str_append_text_char(str, glyph, byte_len);
8386         i++;
8387     }
8388     return i;
8389 }
8390 NK_API int
8391 nk_str_insert_at_char(struct nk_str *s, int pos, const char *str, int len)
8392 {
8393     int i;
8394     void *mem;
8395     char *src;
8396     char *dst;
8397
8398     int copylen;
8399     NK_ASSERT(s);
8400     NK_ASSERT(str);
8401     NK_ASSERT(len >= 0);
8402     if (!s || !str || !len || (nk_size)pos > s->buffer.allocated) return 0;
8403     if ((s->buffer.allocated + (nk_size)len >= s->buffer.memory.size) &&
8404         (s->buffer.type == NK_BUFFER_FIXED)) return 0;
8405
8406     copylen = (int)s->buffer.allocated - pos;
8407     if (!copylen) {
8408         nk_str_append_text_char(s, str, len);
8409         return 1;
8410     }
8411     mem = nk_buffer_alloc(&s->buffer, NK_BUFFER_FRONT, (nk_size)len * sizeof(char), 0);
8412     if (!mem) return 0;
8413
8414     /* memmove */
8415     NK_ASSERT(((int)pos + (int)len + ((int)copylen - 1)) >= 0);
8416     NK_ASSERT(((int)pos + ((int)copylen - 1)) >= 0);
8417     dst = nk_ptr_add(char, s->buffer.memory.ptr, pos + len + (copylen - 1));
8418     src = nk_ptr_add(char, s->buffer.memory.ptr, pos + (copylen-1));
8419     for (i = 0; i < copylen; ++i) *dst-- = *src--;
8420     mem = nk_ptr_add(void, s->buffer.memory.ptr, pos);
8421     NK_MEMCPY(mem, str, (nk_size)len * sizeof(char));
8422     s->len = nk_utf_len((char *)s->buffer.memory.ptr, (int)s->buffer.allocated);
8423     return 1;
8424 }
8425 NK_API int
```

```
8426 nk_str_insert_at_rune(struct nk_str *str, int pos, const char *cstr, int len)
8427 {
8428     int glyph_len;
8429     nk_rune unicode;
8430     const char *begin;
8431     const char *buffer;
8432
8433     NK_ASSERT(str);
8434     NK_ASSERT(cstr);
8435     NK_ASSERT(len);
8436     if (!str || !cstr || !len) return 0;
8437     begin = nk_str_at_rune(str, pos, &unicode, &glyph_len);
8438     if (!str->len)
8439         return nk_str_append_text_char(str, cstr, len);
8440     buffer = nk_str_get_const(str);
8441     if (!begin) return 0;
8442     return nk_str_insert_at_char(str, (int)(begin - buffer), cstr, len);
8443 }
8444 NK_API int
8445 nk_str_insert_text_char(struct nk_str *str, int pos, const char *text, int len)
8446 {
8447     return nk_str_insert_text_utf8(str, pos, text, len);
8448 }
8449 NK_API int
8450 nk_str_insert_str_char(struct nk_str *str, int pos, const char *text)
8451 {
8452     return nk_str_insert_text_utf8(str, pos, text, nk_strlen(text));
8453 }
8454 NK_API int
8455 nk_str_insert_text_utf8(struct nk_str *str, int pos, const char *text, int len)
8456 {
8457     int i = 0;
8458     int byte_len = 0;
8459     nk_rune unicode;
8460
8461     NK_ASSERT(str);
8462     NK_ASSERT(text);
8463     if (!str || !text || !len) return 0;
8464     for (i = 0; i < len; ++i)
8465         byte_len += nk_utf_decode(text+byte_len, &unicode, 4);
8466     nk_str_insert_at_rune(str, pos, text, byte_len);
8467     return len;
8468 }
8469 NK_API int
8470 nk_str_insert_str_utf8(struct nk_str *str, int pos, const char *text)
8471 {
8472     int runes = 0;
8473     int byte_len = 0;
8474     int num_runes = 0;
8475     int glyph_len = 0;
8476     nk_rune unicode;
8477     if (!str || !text) return 0;
8478
8479     glyph_len = byte_len = nk_utf_decode(text+byte_len, &unicode, 4);
8480     while (unicode != '\0' && glyph_len) {
8481         glyph_len = nk_utf_decode(text+byte_len, &unicode, 4);
8482         byte_len += glyph_len;
8483         num_runes++;
8484     }
8485     nk_str_insert_at_rune(str, pos, text, byte_len);
8486     return runes;
8487 }
8488 NK_API int
8489 nk_str_insert_text_runes(struct nk_str *str, int pos, const nk_rune *runes, int len)
8490 {
8491     int i = 0;
8492     int byte_len = 0;
8493     nk_glyph glyph;
8494
8495     NK_ASSERT(str);
8496     if (!str || !runes || !len) return 0;
8497     for (i = 0; i < len; ++i) {
8498         byte_len = nk_utf_encode(runes[i], glyph, NK_UTF_SIZE);
8499         if (!byte_len) break;
8500         nk_str_insert_at_rune(str, pos+i, glyph, byte_len);
8501     }
8502     return len;
8503 }
8504 NK_API int
8505 nk_str_insert_str_runes(struct nk_str *str, int pos, const nk_rune *runes)
8506 {
8507     int i = 0;
8508     nk_glyph glyph;
8509     int byte_len;
8510     NK_ASSERT(str);
8511     if (!str || !runes) return 0;
8512     while (runes[i] != '\0') {
```

```
8513         byte_len = nk_utf_encode(runes[i], glyph, NK_UTF_SIZE);
8514         nk_str_insert_at_rune(str, pos+i, glyph, byte_len);
8515         i++;
8516     }
8517     return i;
8518 }
8519 NK_API void
8520 nk_str_remove_chars(struct nk_str *s, int len)
8521 {
8522     NK_ASSERT(s);
8523     NK_ASSERT(len >= 0);
8524     if (!s || len < 0 || (nk_size)len > s->buffer.allocated) return;
8525     NK_ASSERT(((int)s->buffer.allocated - (int)len) >= 0);
8526     s->buffer.allocated -= (nk_size)len;
8527     s->len = nk_utf_len((char *)s->buffer.memory.ptr, (int)s->buffer.allocated);
8528 }
8529 NK_API void
8530 nk_str_remove_runes(struct nk_str *str, int len)
8531 {
8532     int index;
8533     const char *begin;
8534     const char *end;
8535     nk_rune unicode;
8536
8537     NK_ASSERT(str);
8538     NK_ASSERT(len >= 0);
8539     if (!str || len < 0) return;
8540     if (len >= str->len) {
8541         str->len = 0;
8542         return;
8543     }
8544
8545     index = str->len - len;
8546     begin = nk_str_at_rune(str, index, &unicode, &len);
8547     end = (const char*)str->buffer.memory.ptr + str->buffer.allocated;
8548     nk_str_remove_chars(str, (int)(end-begin)+1);
8549 }
8550 NK_API void
8551 nk_str_delete_chars(struct nk_str *s, int pos, int len)
8552 {
8553     NK_ASSERT(s);
8554     if (!s || !len || (nk_size)pos > s->buffer.allocated ||
8555         (nk_size)(pos + len) > s->buffer.allocated) return;
8556
8557     if ((nk_size)(pos + len) < s->buffer.allocated) {
8558         /* memmove */
8559         char *dst = nk_ptr_add(char, s->buffer.memory.ptr, pos);
8560         char *src = nk_ptr_add(char, s->buffer.memory.ptr, pos + len);
8561         NK_MEMCPY(dst, src, s->buffer.allocated - (nk_size)(pos + len));
8562         NK_ASSERT(((int)s->buffer.allocated - (int)len) >= 0);
8563         s->buffer.allocated -= (nk_size)len;
8564     } else nk_str_remove_chars(s, len);
8565     s->len = nk_utf_len((char *)s->buffer.memory.ptr, (int)s->buffer.allocated);
8566 }
8567 NK_API void
8568 nk_str_delete_runes(struct nk_str *s, int pos, int len)
8569 {
8570     char *temp;
8571     nk_rune unicode;
8572     char *begin;
8573     char *end;
8574     int unused;
8575
8576     NK_ASSERT(s);
8577     NK_ASSERT(s->len >= pos + len);
8578     if (s->len < pos + len)
8579         len = NK_CLAMP(0, (s->len - pos), s->len);
8580     if (!len) return;
8581
8582     temp = (char *)s->buffer.memory.ptr;
8583     begin = nk_str_at_rune(s, pos, &unicode, &unused);
8584     if (!begin) return;
8585     s->buffer.memory.ptr = begin;
8586     end = nk_str_at_rune(s, len, &unicode, &unused);
8587     s->buffer.memory.ptr = temp;
8588     if (!end) return;
8589     nk_str_delete_chars(s, (int)(begin - temp), (int)(end - begin));
8590 }
8591 NK_API char*
8592 nk_str_at_char(struct nk_str *s, int pos)
8593 {
8594     NK_ASSERT(s);
8595     if (!s || pos > (int)s->buffer.allocated) return 0;
8596     return nk_ptr_add(char, s->buffer.memory.ptr, pos);
8597 }
8598 NK_API char*
8599 nk_str_at_rune(struct nk_str *str, int pos, nk_rune *unicode, int *len)
```

```
8600 {
8601     int i = 0;
8602     int src_len = 0;
8603     int glyph_len = 0;
8604     char *text;
8605     int text_len;
8606
8607     NK_ASSERT(str);
8608     NK_ASSERT(unicode);
8609     NK_ASSERT(len);
8610
8611     if (!str || !unicode || !len) return 0;
8612     if (pos < 0) {
8613         *unicode = 0;
8614         *len = 0;
8615         return 0;
8616     }
8617
8618     text = (char*)str->buffer.memory.ptr;
8619     text_len = (int)str->buffer.allocated;
8620     glyph_len = nk_utf_decode(text, unicode, text_len);
8621     while (glyph_len) {
8622         if (i == pos) {
8623             *len = glyph_len;
8624             break;
8625         }
8626
8627         i++;
8628         src_len = src_len + glyph_len;
8629         glyph_len = nk_utf_decode(text + src_len, unicode, text_len - src_len);
8630     }
8631     if (i != pos) return 0;
8632     return text + src_len;
8633 }
8634 NK_API const char*
8635 nk_str_at_char_const(const struct nk_str *s, int pos)
8636 {
8637     NK_ASSERT(s);
8638     if (!s || pos > (int)s->buffer.allocated) return 0;
8639     return nk_ptr_add(char, s->buffer.memory.ptr, pos);
8640 }
8641 NK_API const char*
8642 nk_str_at_const(const struct nk_str *str, int pos, nk_rune *unicode, int *len)
8643 {
8644     int i = 0;
8645     int src_len = 0;
8646     int glyph_len = 0;
8647     char *text;
8648     int text_len;
8649
8650     NK_ASSERT(str);
8651     NK_ASSERT(unicode);
8652     NK_ASSERT(len);
8653
8654     if (!str || !unicode || !len) return 0;
8655     if (pos < 0) {
8656         *unicode = 0;
8657         *len = 0;
8658         return 0;
8659     }
8660
8661     text = (char*)str->buffer.memory.ptr;
8662     text_len = (int)str->buffer.allocated;
8663     glyph_len = nk_utf_decode(text, unicode, text_len);
8664     while (glyph_len) {
8665         if (i == pos) {
8666             *len = glyph_len;
8667             break;
8668         }
8669
8670         i++;
8671         src_len = src_len + glyph_len;
8672         glyph_len = nk_utf_decode(text + src_len, unicode, text_len - src_len);
8673     }
8674     if (i != pos) return 0;
8675     return text + src_len;
8676 }
8677 NK_API nk_rune
8678 nk_str_rune_at(const struct nk_str *str, int pos)
8679 {
8680     int len;
8681     nk_rune unicode = 0;
8682     nk_str_at_const(str, pos, &unicode, &len);
8683     return unicode;
8684 }
8685 NK_API char*
8686 nk_str_get(struct nk_str *s)
```

```
8687 {
8688     NK_ASSERT(s);
8689     if (!s || !s->len || !s->buffer.allocated) return 0;
8690     return (char*)s->buffer.memory.ptr;
8691 }
8692 NK_API const char*
8693 nk_str_get_const(const struct nk_str *s)
8694 {
8695     NK_ASSERT(s);
8696     if (!s || !s->len || !s->buffer.allocated) return 0;
8697     return (const char*)s->buffer.memory.ptr;
8698 }
8699 NK_API int
8700 nk_str_len(struct nk_str *s)
8701 {
8702     NK_ASSERT(s);
8703     if (!s || !s->len || !s->buffer.allocated) return 0;
8704     return s->len;
8705 }
8706 NK_API int
8707 nk_str_len_char(struct nk_str *s)
8708 {
8709     NK_ASSERT(s);
8710     if (!s || !s->len || !s->buffer.allocated) return 0;
8711     return (int)s->buffer.allocated;
8712 }
8713 NK_API void
8714 nk_str_clear(struct nk_str *str)
8715 {
8716     NK_ASSERT(str);
8717     nk_buffer_clear(&str->buffer);
8718     str->len = 0;
8719 }
8720 NK_API void
8721 nk_str_free(struct nk_str *str)
8722 {
8723     NK_ASSERT(str);
8724     nk_buffer_free(&str->buffer);
8725     str->len = 0;
8726 }
8727
8728
8729
8730
8731
8732 /* ==============================================================
8733  *
8734  *                          DRAW
8735  *
8736  * ===============================================================*/
8737 NK_LIB void
8738 nk_command_buffer_init(struct nk_command_buffer *cb,
8739     struct nk_buffer *b, enum nk_command_clipping clip)
8740 {
8741     NK_ASSERT(cb);
8742     NK_ASSERT(b);
8743     if (!cb || !b) return;
8744     cb->base = b;
8745     cb->use_clipping = (int)clip;
8746     cb->begin = b->allocated;
8747     cb->end = b->allocated;
8748     cb->last = b->allocated;
8749 }
8750 NK_LIB void
8751 nk_command_buffer_reset(struct nk_command_buffer *b)
8752 {
8753     NK_ASSERT(b);
8754     if (!b) return;
8755     b->begin = 0;
8756     b->end = 0;
8757     b->last = 0;
8758     b->clip = nk_null_rect;
8759 #ifdef NK_INCLUDE_COMMAND_USERDATA
8760     b->userdata.ptr = 0;
8761 #endif
8762 }
8763 NK_LIB void*
8764 nk_command_buffer_push(struct nk_command_buffer* b,
8765     enum nk_command_type t, nk_size size)
8766 {
8767     NK_STORAGE const nk_size align = NK_ALIGNOF(struct nk_command);
8768     struct nk_command *cmd;
8769     nk_size alignment;
8770     void *unaligned;
8771     void *memory;
8772
8773     NK_ASSERT(b);
```

```
8774     NK_ASSERT(b->base);
8775     if (!b) return 0;
8776     cmd = (struct nk_command*)nk_buffer_alloc(b->base,NK_BUFFER_FRONT,size,align);
8777     if (!cmd) return 0;
8778
8779     /* make sure the offset to the next command is aligned */
8780     b->last = (nk_size)((nk_byte*)cmd - (nk_byte*)b->base->memory.ptr);
8781     unaligned = (nk_byte*)cmd + size;
8782     memory = NK_ALIGN_PTR(unaligned, align);
8783     alignment = (nk_size)((nk_byte*)memory - (nk_byte*)unaligned);
8784 #ifdef NK_ZERO_COMMAND_MEMORY
8785     NK_MEMSET(cmd, 0, size + alignment);
8786 #endif
8787
8788     cmd->type = t;
8789     cmd->next = b->base->allocated + alignment;
8790 #ifdef NK_INCLUDE_COMMAND_USERDATA
8791     cmd->userdata = b->userdata;
8792 #endif
8793     b->end = cmd->next;
8794     return cmd;
8795 }
8796 NK_API void
8797 nk_push_scissor(struct nk_command_buffer *b, struct nk_rect r)
8798 {
8799     struct nk_command_scissor *cmd;
8800     NK_ASSERT(b);
8801     if (!b) return;
8802
8803     b->clip.x = r.x;
8804     b->clip.y = r.y;
8805     b->clip.w = r.w;
8806     b->clip.h = r.h;
8807     cmd = (struct nk_command_scissor*)
8808         nk_command_buffer_push(b, NK_COMMAND_SCISSOR, sizeof(*cmd));
8809
8810     if (!cmd) return;
8811     cmd->x = (short)r.x;
8812     cmd->y = (short)r.y;
8813     cmd->w = (unsigned short)NK_MAX(0, r.w);
8814     cmd->h = (unsigned short)NK_MAX(0, r.h);
8815 }
8816 NK_API void
8817 nk_stroke_line(struct nk_command_buffer *b, float x0, float y0,
8818     float x1, float y1, float line_thickness, struct nk_color c)
8819 {
8820     struct nk_command_line *cmd;
8821     NK_ASSERT(b);
8822     if (!b || line_thickness <= 0) return;
8823     cmd = (struct nk_command_line*)
8824         nk_command_buffer_push(b, NK_COMMAND_LINE, sizeof(*cmd));
8825     if (!cmd) return;
8826     cmd->line_thickness = (unsigned short)line_thickness;
8827     cmd->begin.x = (short)x0;
8828     cmd->begin.y = (short)y0;
8829     cmd->end.x = (short)x1;
8830     cmd->end.y = (short)y1;
8831     cmd->color = c;
8832 }
8833 NK_API void
8834 nk_stroke_curve(struct nk_command_buffer *b, float ax, float ay,
8835     float ctrl0x, float ctrl0y, float ctrl1x, float ctrl1y,
8836     float bx, float by, float line_thickness, struct nk_color col)
8837 {
8838     struct nk_command_curve *cmd;
8839     NK_ASSERT(b);
8840     if (!b || col.a == 0 || line_thickness <= 0) return;
8841
8842     cmd = (struct nk_command_curve*)
8843         nk_command_buffer_push(b, NK_COMMAND_CURVE, sizeof(*cmd));
8844     if (!cmd) return;
8845     cmd->line_thickness = (unsigned short)line_thickness;
8846     cmd->begin.x = (short)ax;
8847     cmd->begin.y = (short)ay;
8848     cmd->ctrl[0].x = (short)ctrl0x;
8849     cmd->ctrl[0].y = (short)ctrl0y;
8850     cmd->ctrl[1].x = (short)ctrl1x;
8851     cmd->ctrl[1].y = (short)ctrl1y;
8852     cmd->end.x = (short)bx;
8853     cmd->end.y = (short)by;
8854     cmd->color = col;
8855 }
8856 NK_API void
8857 nk_stroke_rect(struct nk_command_buffer *b, struct nk_rect rect,
8858     float rounding, float line_thickness, struct nk_color c)
8859 {
8860     struct nk_command_rect *cmd;
```

```
8861     NK_ASSERT(b);
8862     if (!b || c.a == 0 || rect.w == 0 || rect.h == 0 || line_thickness <= 0) return;
8863     if (b->use_clipping) {
8864         const struct nk_rect *clip = &b->clip;
8865         if (!NK_INTERSECT(rect.x, rect.y, rect.w, rect.h,
8866             clip->x, clip->y, clip->w, clip->h)) return;
8867     }
8868     cmd = (struct nk_command_rect*)
8869         nk_command_buffer_push(b, NK_COMMAND_RECT, sizeof(*cmd));
8870     if (!cmd) return;
8871     cmd->rounding = (unsigned short)rounding;
8872     cmd->line_thickness = (unsigned short)line_thickness;
8873     cmd->x = (short)rect.x;
8874     cmd->y = (short)rect.y;
8875     cmd->w = (unsigned short)NK_MAX(0, rect.w);
8876     cmd->h = (unsigned short)NK_MAX(0, rect.h);
8877     cmd->color = c;
8878 }
8879 NK_API void
8880 nk_fill_rect(struct nk_command_buffer *b, struct nk_rect rect,
8881     float rounding, struct nk_color c)
8882 {
8883     struct nk_command_rect_filled *cmd;
8884     NK_ASSERT(b);
8885     if (!b || c.a == 0 || rect.w == 0 || rect.h == 0) return;
8886     if (b->use_clipping) {
8887         const struct nk_rect *clip = &b->clip;
8888         if (!NK_INTERSECT(rect.x, rect.y, rect.w, rect.h,
8889             clip->x, clip->y, clip->w, clip->h)) return;
8890     }
8891
8892     cmd = (struct nk_command_rect_filled*)
8893         nk_command_buffer_push(b, NK_COMMAND_RECT_FILLED, sizeof(*cmd));
8894     if (!cmd) return;
8895     cmd->rounding = (unsigned short)rounding;
8896     cmd->x = (short)rect.x;
8897     cmd->y = (short)rect.y;
8898     cmd->w = (unsigned short)NK_MAX(0, rect.w);
8899     cmd->h = (unsigned short)NK_MAX(0, rect.h);
8900     cmd->color = c;
8901 }
8902 NK_API void
8903 nk_fill_rect_multi_color(struct nk_command_buffer *b, struct nk_rect rect,
8904     struct nk_color left, struct nk_color top, struct nk_color right,
8905     struct nk_color bottom)
8906 {
8907     struct nk_command_rect_multi_color *cmd;
8908     NK_ASSERT(b);
8909     if (!b || rect.w == 0 || rect.h == 0) return;
8910     if (b->use_clipping) {
8911         const struct nk_rect *clip = &b->clip;
8912         if (!NK_INTERSECT(rect.x, rect.y, rect.w, rect.h,
8913             clip->x, clip->y, clip->w, clip->h)) return;
8914     }
8915
8916     cmd = (struct nk_command_rect_multi_color*)
8917         nk_command_buffer_push(b, NK_COMMAND_RECT_MULTI_COLOR, sizeof(*cmd));
8918     if (!cmd) return;
8919     cmd->x = (short)rect.x;
8920     cmd->y = (short)rect.y;
8921     cmd->w = (unsigned short)NK_MAX(0, rect.w);
8922     cmd->h = (unsigned short)NK_MAX(0, rect.h);
8923     cmd->left = left;
8924     cmd->top = top;
8925     cmd->right = right;
8926     cmd->bottom = bottom;
8927 }
8928 NK_API void
8929 nk_stroke_circle(struct nk_command_buffer *b, struct nk_rect r,
8930     float line_thickness, struct nk_color c)
8931 {
8932     struct nk_command_circle *cmd;
8933     if (!b || r.w == 0 || r.h == 0 || line_thickness <= 0) return;
8934     if (b->use_clipping) {
8935         const struct nk_rect *clip = &b->clip;
8936         if (!NK_INTERSECT(r.x, r.y, r.w, r.h, clip->x, clip->y, clip->w, clip->h))
8937             return;
8938     }
8939
8940     cmd = (struct nk_command_circle*)
8941         nk_command_buffer_push(b, NK_COMMAND_CIRCLE, sizeof(*cmd));
8942     if (!cmd) return;
8943     cmd->line_thickness = (unsigned short)line_thickness;
8944     cmd->x = (short)r.x;
8945     cmd->y = (short)r.y;
8946     cmd->w = (unsigned short)NK_MAX(r.w, 0);
8947     cmd->h = (unsigned short)NK_MAX(r.h, 0);
```

```
8948     cmd->color = c;
8949 }
8950 NK_API void
8951 nk_fill_circle(struct nk_command_buffer *b, struct nk_rect r, struct nk_color c)
8952 {
8953     struct nk_command_circle_filled *cmd;
8954     NK_ASSERT(b);
8955     if (!b || c.a == 0 || r.w == 0 || r.h == 0) return;
8956     if (b->use_clipping) {
8957         const struct nk_rect *clip = &b->clip;
8958         if (!NK_INTERSECT(r.x, r.y, r.w, r.h, clip->x, clip->y, clip->w, clip->h))
8959             return;
8960     }
8961
8962     cmd = (struct nk_command_circle_filled*)
8963         nk_command_buffer_push(b, NK_COMMAND_CIRCLE_FILLED, sizeof(*cmd));
8964     if (!cmd) return;
8965     cmd->x = (short)r.x;
8966     cmd->y = (short)r.y;
8967     cmd->w = (unsigned short)NK_MAX(r.w, 0);
8968     cmd->h = (unsigned short)NK_MAX(r.h, 0);
8969     cmd->color = c;
8970 }
8971 NK_API void
8972 nk_stroke_arc(struct nk_command_buffer *b, float cx, float cy, float radius,
8973     float a_min, float a_max, float line_thickness, struct nk_color c)
8974 {
8975     struct nk_command_arc *cmd;
8976     if (!b || c.a == 0 || line_thickness <= 0) return;
8977     cmd = (struct nk_command_arc*)
8978         nk_command_buffer_push(b, NK_COMMAND_ARC, sizeof(*cmd));
8979     if (!cmd) return;
8980     cmd->line_thickness = (unsigned short)line_thickness;
8981     cmd->cx = (short)cx;
8982     cmd->cy = (short)cy;
8983     cmd->r = (unsigned short)radius;
8984     cmd->a[0] = a_min;
8985     cmd->a[1] = a_max;
8986     cmd->color = c;
8987 }
8988 NK_API void
8989 nk_fill_arc(struct nk_command_buffer *b, float cx, float cy, float radius,
8990     float a_min, float a_max, struct nk_color c)
8991 {
8992     struct nk_command_arc_filled *cmd;
8993     NK_ASSERT(b);
8994     if (!b || c.a == 0) return;
8995     cmd = (struct nk_command_arc_filled*)
8996         nk_command_buffer_push(b, NK_COMMAND_ARC_FILLED, sizeof(*cmd));
8997     if (!cmd) return;
8998     cmd->cx = (short)cx;
8999     cmd->cy = (short)cy;
9000     cmd->r = (unsigned short)radius;
9001     cmd->a[0] = a_min;
9002     cmd->a[1] = a_max;
9003     cmd->color = c;
9004 }
9005 NK_API void
9006 nk_stroke_triangle(struct nk_command_buffer *b, float x0, float y0, float x1,
9007     float y1, float x2, float y2, float line_thickness, struct nk_color c)
9008 {
9009     struct nk_command_triangle *cmd;
9010     NK_ASSERT(b);
9011     if (!b || c.a == 0 || line_thickness <= 0) return;
9012     if (b->use_clipping) {
9013         const struct nk_rect *clip = &b->clip;
9014         if (!NK_INBOX(x0, y0, clip->x, clip->y, clip->w, clip->h) &&
9015             !NK_INBOX(x1, y1, clip->x, clip->y, clip->w, clip->h) &&
9016             !NK_INBOX(x2, y2, clip->x, clip->y, clip->w, clip->h))
9017             return;
9018     }
9019
9020     cmd = (struct nk_command_triangle*)
9021         nk_command_buffer_push(b, NK_COMMAND_TRIANGLE, sizeof(*cmd));
9022     if (!cmd) return;
9023     cmd->line_thickness = (unsigned short)line_thickness;
9024     cmd->a.x = (short)x0;
9025     cmd->a.y = (short)y0;
9026     cmd->b.x = (short)x1;
9027     cmd->b.y = (short)y1;
9028     cmd->c.x = (short)x2;
9029     cmd->c.y = (short)y2;
9030     cmd->color = c;
9031 }
9032 NK_API void
9033 nk_fill_triangle(struct nk_command_buffer *b, float x0, float y0, float x1,
9034     float y1, float x2, float y2, struct nk_color c)
```

```
9035 {
9036     struct nk_command_triangle_filled *cmd;
9037     NK_ASSERT(b);
9038     if (!b || c.a == 0) return;
9039     if (!b) return;
9040     if (b->use_clipping) {
9041         const struct nk_rect *clip = &b->clip;
9042         if (!NK_INBOX(x0, y0, clip->x, clip->y, clip->w, clip->h) &&
9043             !NK_INBOX(x1, y1, clip->x, clip->y, clip->w, clip->h) &&
9044             !NK_INBOX(x2, y2, clip->x, clip->y, clip->w, clip->h))
9045             return;
9046     }
9047
9048     cmd = (struct nk_command_triangle_filled*)
9049         nk_command_buffer_push(b, NK_COMMAND_TRIANGLE_FILLED, sizeof(*cmd));
9050     if (!cmd) return;
9051     cmd->a.x = (short)x0;
9052     cmd->a.y = (short)y0;
9053     cmd->b.x = (short)x1;
9054     cmd->b.y = (short)y1;
9055     cmd->c.x = (short)x2;
9056     cmd->c.y = (short)y2;
9057     cmd->color = c;
9058 }
9059 NK_API void
9060 nk_stroke_polygon(struct nk_command_buffer *b,  float *points, int point_count,
9061     float line_thickness, struct nk_color col)
9062 {
9063     int i;
9064     nk_size size = 0;
9065     struct nk_command_polygon *cmd;
9066
9067     NK_ASSERT(b);
9068     if (!b || col.a == 0 || line_thickness <= 0) return;
9069     size = sizeof(*cmd) + sizeof(short) * 2 * (nk_size)point_count;
9070     cmd = (struct nk_command_polygon*) nk_command_buffer_push(b, NK_COMMAND_POLYGON, size);
9071     if (!cmd) return;
9072     cmd->color = col;
9073     cmd->line_thickness = (unsigned short)line_thickness;
9074     cmd->point_count = (unsigned short)point_count;
9075     for (i = 0; i < point_count; ++i) {
9076         cmd->points[i].x = (short)points[i*2];
9077         cmd->points[i].y = (short)points[i*2+1];
9078     }
9079 }
9080 NK_API void
9081 nk_fill_polygon(struct nk_command_buffer *b, float *points, int point_count,
9082     struct nk_color col)
9083 {
9084     int i;
9085     nk_size size = 0;
9086     struct nk_command_polygon_filled *cmd;
9087
9088     NK_ASSERT(b);
9089     if (!b || col.a == 0) return;
9090     size = sizeof(*cmd) + sizeof(short) * 2 * (nk_size)point_count;
9091     cmd = (struct nk_command_polygon_filled*)
9092         nk_command_buffer_push(b, NK_COMMAND_POLYGON_FILLED, size);
9093     if (!cmd) return;
9094     cmd->color = col;
9095     cmd->point_count = (unsigned short)point_count;
9096     for (i = 0; i < point_count; ++i) {
9097         cmd->points[i].x = (short)points[i*2+0];
9098         cmd->points[i].y = (short)points[i*2+1];
9099     }
9100 }
9101 NK_API void
9102 nk_stroke_polyline(struct nk_command_buffer *b, float *points, int point_count,
9103     float line_thickness, struct nk_color col)
9104 {
9105     int i;
9106     nk_size size = 0;
9107     struct nk_command_polyline *cmd;
9108
9109     NK_ASSERT(b);
9110     if (!b || col.a == 0 || line_thickness <= 0) return;
9111     size = sizeof(*cmd) + sizeof(short) * 2 * (nk_size)point_count;
9112     cmd = (struct nk_command_polyline*) nk_command_buffer_push(b, NK_COMMAND_POLYLINE, size);
9113     if (!cmd) return;
9114     cmd->color = col;
9115     cmd->point_count = (unsigned short)point_count;
9116     cmd->line_thickness = (unsigned short)line_thickness;
9117     for (i = 0; i < point_count; ++i) {
9118         cmd->points[i].x = (short)points[i*2];
9119         cmd->points[i].y = (short)points[i*2+1];
9120     }
9121 }
```

```
9122 NK_API void
9123 nk_draw_image(struct nk_command_buffer *b, struct nk_rect r,
9124     const struct nk_image *img, struct nk_color col)
9125 {
9126     struct nk_command_image *cmd;
9127     NK_ASSERT(b);
9128     if (!b) return;
9129     if (b->use_clipping) {
9130         const struct nk_rect *c = &b->clip;
9131         if (c->w == 0 || c->h == 0 || !NK_INTERSECT(r.x, r.y, r.w, r.h, c->x, c->y, c->w, c->h))
9132             return;
9133     }
9134
9135     cmd = (struct nk_command_image*)
9136         nk_command_buffer_push(b, NK_COMMAND_IMAGE, sizeof(*cmd));
9137     if (!cmd) return;
9138     cmd->x = (short)r.x;
9139     cmd->y = (short)r.y;
9140     cmd->w = (unsigned short)NK_MAX(0, r.w);
9141     cmd->h = (unsigned short)NK_MAX(0, r.h);
9142     cmd->img = *img;
9143     cmd->col = col;
9144 }
9145 NK_API void
9146 nk_push_custom(struct nk_command_buffer *b, struct nk_rect r,
9147     nk_command_custom_callback cb, nk_handle usr)
9148 {
9149     struct nk_command_custom *cmd;
9150     NK_ASSERT(b);
9151     if (!b) return;
9152     if (b->use_clipping) {
9153         const struct nk_rect *c = &b->clip;
9154         if (c->w == 0 || c->h == 0 || !NK_INTERSECT(r.x, r.y, r.w, r.h, c->x, c->y, c->w, c->h))
9155             return;
9156     }
9157
9158     cmd = (struct nk_command_custom*)
9159         nk_command_buffer_push(b, NK_COMMAND_CUSTOM, sizeof(*cmd));
9160     if (!cmd) return;
9161     cmd->x = (short)r.x;
9162     cmd->y = (short)r.y;
9163     cmd->w = (unsigned short)NK_MAX(0, r.w);
9164     cmd->h = (unsigned short)NK_MAX(0, r.h);
9165     cmd->callback_data = usr;
9166     cmd->callback = cb;
9167 }
9168 NK_API void
9169 nk_draw_text(struct nk_command_buffer *b, struct nk_rect r,
9170     const char *string, int length, const struct nk_user_font *font,
9171     struct nk_color bg, struct nk_color fg)
9172 {
9173     float text_width = 0;
9174     struct nk_command_text *cmd;
9175
9176     NK_ASSERT(b);
9177     NK_ASSERT(font);
9178     if (!b || !string || !length || (bg.a == 0 && fg.a == 0)) return;
9179     if (b->use_clipping) {
9180         const struct nk_rect *c = &b->clip;
9181         if (c->w == 0 || c->h == 0 || !NK_INTERSECT(r.x, r.y, r.w, r.h, c->x, c->y, c->w, c->h))
9182             return;
9183     }
9184
9185     /* make sure text fits inside bounds */
9186     text_width = font->width(font->userdata, font->height, string, length);
9187     if (text_width > r.w){
9188         int glyphs = 0;
9189         float txt_width = (float)text_width;
9190         length = nk_text_clamp(font, string, length, r.w, &glyphs, &txt_width, 0,0);
9191     }
9192
9193     if (!length) return;
9194     cmd = (struct nk_command_text*)
9195         nk_command_buffer_push(b, NK_COMMAND_TEXT, sizeof(*cmd) + (nk_size)(length + 1));
9196     if (!cmd) return;
9197     cmd->x = (short)r.x;
9198     cmd->y = (short)r.y;
9199     cmd->w = (unsigned short)r.w;
9200     cmd->h = (unsigned short)r.h;
9201     cmd->background = bg;
9202     cmd->foreground = fg;
9203     cmd->font = font;
9204     cmd->length = length;
9205     cmd->height = font->height;
9206     NK_MEMCPY(cmd->string, string, (nk_size)length);
9207     cmd->string[length] = '\0';
9208 }
```

```
9209
9210
9211
9212
9213
9214 /* ===============================================================
9215  *
9216  *                              VERTEX
9217  *
9218  * ===============================================================*/
9219 #ifdef NK_INCLUDE_VERTEX_BUFFER_OUTPUT
9220 NK_API void
9221 nk_draw_list_init(struct nk_draw_list *list)
9222 {
9223     nk_size i = 0;
9224     NK_ASSERT(list);
9225     if (!list) return;
9226     nk_zero(list, sizeof(*list));
9227     for (i = 0; i < NK_LEN(list->circle_vtx); ++i) {
9228         const float a = ((float)i / (float)NK_LEN(list->circle_vtx)) * 2 * NK_PI;
9229         list->circle_vtx[i].x = (float)NK_COS(a);
9230         list->circle_vtx[i].y = (float)NK_SIN(a);
9231     }
9232 }
9233 NK_API void
9234 nk_draw_list_setup(struct nk_draw_list *canvas, const struct nk_convert_config *config,
9235     struct nk_buffer *cmds, struct nk_buffer *vertices, struct nk_buffer *elements,
9236     enum nk_anti_aliasing line_aa, enum nk_anti_aliasing shape_aa)
9237 {
9238     NK_ASSERT(canvas);
9239     NK_ASSERT(config);
9240     NK_ASSERT(cmds);
9241     NK_ASSERT(vertices);
9242     NK_ASSERT(elements);
9243     if (!canvas || !config || !cmds || !vertices || !elements)
9244         return;
9245
9246     canvas->buffer = cmds;
9247     canvas->config = *config;
9248     canvas->elements = elements;
9249     canvas->vertices = vertices;
9250     canvas->line_AA = line_aa;
9251     canvas->shape_AA = shape_aa;
9252     canvas->clip_rect = nk_null_rect;
9253
9254     canvas->cmd_offset = 0;
9255     canvas->element_count = 0;
9256     canvas->vertex_count = 0;
9257     canvas->cmd_offset = 0;
9258     canvas->cmd_count = 0;
9259     canvas->path_count = 0;
9260 }
9261 NK_API const struct nk_draw_command*
9262 nk__draw_list_begin(const struct nk_draw_list *canvas, const struct nk_buffer *buffer)
9263 {
9264     nk_byte *memory;
9265     nk_size offset;
9266     const struct nk_draw_command *cmd;
9267
9268     NK_ASSERT(buffer);
9269     if (!buffer || !buffer->size || !canvas->cmd_count)
9270         return 0;
9271
9272     memory = (nk_byte*)buffer->memory.ptr;
9273     offset = buffer->memory.size - canvas->cmd_offset;
9274     cmd = nk_ptr_add(const struct nk_draw_command, memory, offset);
9275     return cmd;
9276 }
9277 NK_API const struct nk_draw_command*
9278 nk__draw_list_end(const struct nk_draw_list *canvas, const struct nk_buffer *buffer)
9279 {
9280     nk_size size;
9281     nk_size offset;
9282     nk_byte *memory;
9283     const struct nk_draw_command *end;
9284
9285     NK_ASSERT(buffer);
9286     NK_ASSERT(canvas);
9287     if (!buffer || !canvas)
9288         return 0;
9289
9290     memory = (nk_byte*)buffer->memory.ptr;
9291     size = buffer->memory.size;
9292     offset = size - canvas->cmd_offset;
9293     end = nk_ptr_add(const struct nk_draw_command, memory, offset);
9294     end -= (canvas->cmd_count-1);
9295     return end;
```

```
9296 }
9297 NK_API const struct nk_draw_command*
9298 nk__draw_list_next(const struct nk_draw_command *cmd,
9299     const struct nk_buffer *buffer, const struct nk_draw_list *canvas)
9300 {
9301     const struct nk_draw_command *end;
9302     NK_ASSERT(buffer);
9303     NK_ASSERT(canvas);
9304     if (!cmd || !buffer || !canvas)
9305         return 0;
9306
9307     end = nk__draw_list_end(canvas, buffer);
9308     if (cmd <= end) return 0;
9309     return (cmd-1);
9310 }
9311 NK_INTERN struct nk_vec2*
9312 nk_draw_list_alloc_path(struct nk_draw_list *list, int count)
9313 {
9314     struct nk_vec2 *points;
9315     NK_STORAGE const nk_size point_align = NK_ALIGNOF(struct nk_vec2);
9316     NK_STORAGE const nk_size point_size = sizeof(struct nk_vec2);
9317     points = (struct nk_vec2*)
9318         nk_buffer_alloc(list->buffer, NK_BUFFER_FRONT,
9319                         point_size * (nk_size)count, point_align);
9320
9321     if (!points) return 0;
9322     if (!list->path_offset) {
9323         void *memory = nk_buffer_memory(list->buffer);
9324         list->path_offset = (unsigned int)((nk_byte*)points - (nk_byte*)memory);
9325     }
9326     list->path_count += (unsigned int)count;
9327     return points;
9328 }
9329 NK_INTERN struct nk_vec2
9330 nk_draw_list_path_last(struct nk_draw_list *list)
9331 {
9332     void *memory;
9333     struct nk_vec2 *point;
9334     NK_ASSERT(list->path_count);
9335     memory = nk_buffer_memory(list->buffer);
9336     point = nk_ptr_add(struct nk_vec2, memory, list->path_offset);
9337     point += (list->path_count-1);
9338     return *point;
9339 }
9340 NK_INTERN struct nk_draw_command*
9341 nk_draw_list_push_command(struct nk_draw_list *list, struct nk_rect clip,
9342     nk_handle texture)
9343 {
9344     NK_STORAGE const nk_size cmd_align = NK_ALIGNOF(struct nk_draw_command);
9345     NK_STORAGE const nk_size cmd_size = sizeof(struct nk_draw_command);
9346     struct nk_draw_command *cmd;
9347
9348     NK_ASSERT(list);
9349     cmd = (struct nk_draw_command*)
9350         nk_buffer_alloc(list->buffer, NK_BUFFER_BACK, cmd_size, cmd_align);
9351
9352     if (!cmd) return 0;
9353     if (!list->cmd_count) {
9354         nk_byte *memory = (nk_byte*)nk_buffer_memory(list->buffer);
9355         nk_size total = nk_buffer_total(list->buffer);
9356         memory = nk_ptr_add(nk_byte, memory, total);
9357         list->cmd_offset = (nk_size)(memory - (nk_byte*)cmd);
9358     }
9359
9360     cmd->elem_count = 0;
9361     cmd->clip_rect = clip;
9362     cmd->texture = texture;
9363 #ifdef NK_INCLUDE_COMMAND_USERDATA
9364     cmd->userdata = list->userdata;
9365 #endif
9366
9367     list->cmd_count++;
9368     list->clip_rect = clip;
9369     return cmd;
9370 }
9371 NK_INTERN struct nk_draw_command*
9372 nk_draw_list_command_last(struct nk_draw_list *list)
9373 {
9374     void *memory;
9375     nk_size size;
9376     struct nk_draw_command *cmd;
9377     NK_ASSERT(list->cmd_count);
9378
9379     memory = nk_buffer_memory(list->buffer);
9380     size = nk_buffer_total(list->buffer);
9381     cmd = nk_ptr_add(struct nk_draw_command, memory, size - list->cmd_offset);
9382     return (cmd - (list->cmd_count-1));
```

```
9383 }
9384 NK_INTERN void
9385 nk_draw_list_add_clip(struct nk_draw_list *list, struct nk_rect rect)
9386 {
9387     NK_ASSERT(list);
9388     if (!list) return;
9389     if (!list->cmd_count) {
9390         nk_draw_list_push_command(list, rect, list->config.null.texture);
9391     } else {
9392         struct nk_draw_command *prev = nk_draw_list_command_last(list);
9393         if (prev->elem_count == 0)
9394             prev->clip_rect = rect;
9395         nk_draw_list_push_command(list, rect, prev->texture);
9396     }
9397 }
9398 NK_INTERN void
9399 nk_draw_list_push_image(struct nk_draw_list *list, nk_handle texture)
9400 {
9401     NK_ASSERT(list);
9402     if (!list) return;
9403     if (!list->cmd_count) {
9404         nk_draw_list_push_command(list, nk_null_rect, texture);
9405     } else {
9406         struct nk_draw_command *prev = nk_draw_list_command_last(list);
9407         if (prev->elem_count == 0) {
9408             prev->texture = texture;
9409         #ifdef NK_INCLUDE_COMMAND_USERDATA
9410             prev->userdata = list->userdata;
9411         #endif
9412     } else if (prev->texture.id != texture.id
9413         #ifdef NK_INCLUDE_COMMAND_USERDATA
9414             || prev->userdata.id != list->userdata.id
9415         #endif
9416         ) nk_draw_list_push_command(list, prev->clip_rect, texture);
9417     }
9418 }
9419 #ifdef NK_INCLUDE_COMMAND_USERDATA
9420 NK_API void
9421 nk_draw_list_push_userdata(struct nk_draw_list *list, nk_handle userdata)
9422 {
9423     list->userdata = userdata;
9424 }
9425 #endif
9426 NK_INTERN void*
9427 nk_draw_list_alloc_vertices(struct nk_draw_list *list, nk_size count)
9428 {
9429     void *vtx;
9430     NK_ASSERT(list);
9431     if (!list) return 0;
9432     vtx = nk_buffer_alloc(list->vertices, NK_BUFFER_FRONT,
9433         list->config.vertex_size*count, list->config.vertex_alignment);
9434     if (!vtx) return 0;
9435     list->vertex_count += (unsigned int)count;
9436
9437     /* This assert triggers because your are drawing a lot of stuff and nuklear
9438      * defined `nk_draw_index` as `nk_ushort` to safe space be default.
9439      *
9440      * So you reached the maximum number of indicies or rather vertexes.
9441      * To solve this issue please change typdef `nk_draw_index` to `nk_uint`
9442      * and don't forget to specify the new element size in your drawing
9443      * backend (OpenGL, DirectX, ...). For example in OpenGL for `glDrawElements`
9444      * instead of specifing `GL_UNSIGNED_SHORT` you have to define `GL_UNSIGNED_INT`.
9445      * Sorry for the inconvenience. */
9446     if(sizeof(nk_draw_index)==2) NK_ASSERT((list->vertex_count < NK_USHORT_MAX &&
9447         "To many verticies for 16-bit vertex indicies. Please read comment above on how to solve this
      problem"));
9448     return vtx;
9449 }
9450 NK_INTERN nk_draw_index*
9451 nk_draw_list_alloc_elements(struct nk_draw_list *list, nk_size count)
9452 {
9453     nk_draw_index *ids;
9454     struct nk_draw_command *cmd;
9455     NK_STORAGE const nk_size elem_align = NK_ALIGNOF(nk_draw_index);
9456     NK_STORAGE const nk_size elem_size = sizeof(nk_draw_index);
9457     NK_ASSERT(list);
9458     if (!list) return 0;
9459
9460     ids = (nk_draw_index*)
9461         nk_buffer_alloc(list->elements, NK_BUFFER_FRONT, elem_size*count, elem_align);
9462     if (!ids) return 0;
9463     cmd = nk_draw_list_command_last(list);
9464     list->element_count += (unsigned int)count;
9465     cmd->elem_count += (unsigned int)count;
9466     return ids;
9467 }
9468 NK_INTERN int
```

```
9469 nk_draw_vertex_layout_element_is_end_of_layout(
9470     const struct nk_draw_vertex_layout_element *element)
9471 {
9472     return (element->attribute == NK_VERTEX_ATTRIBUTE_COUNT ||
9473             element->format == NK_FORMAT_COUNT);
9474 }
9475 NK_INTERN void
9476 nk_draw_vertex_color(void *attr, const float *vals,
9477     enum nk_draw_vertex_layout_format format)
9478 {
9479     /* if this triggers you tried to provide a value format for a color */
9480     float val[4];
9481     NK_ASSERT(format >= NK_FORMAT_COLOR_BEGIN);
9482     NK_ASSERT(format <= NK_FORMAT_COLOR_END);
9483     if (format < NK_FORMAT_COLOR_BEGIN || format > NK_FORMAT_COLOR_END) return;
9484
9485     val[0] = NK_SATURATE(vals[0]);
9486     val[1] = NK_SATURATE(vals[1]);
9487     val[2] = NK_SATURATE(vals[2]);
9488     val[3] = NK_SATURATE(vals[3]);
9489
9490     switch (format) {
9491     default: NK_ASSERT(0 && "Invalid vertex layout color format"); break;
9492     case NK_FORMAT_R8G8B8A8:
9493     case NK_FORMAT_R8G8B8: {
9494         struct nk_color col = nk_rgba_fv(val);
9495         NK_MEMCPY(attr, &col.r, sizeof(col));
9496     } break;
9497     case NK_FORMAT_B8G8R8A8: {
9498         struct nk_color col = nk_rgba_fv(val);
9499         struct nk_color bgra = nk_rgba(col.b, col.g, col.r, col.a);
9500         NK_MEMCPY(attr, &bgra, sizeof(bgra));
9501     } break;
9502     case NK_FORMAT_R16G15B16: {
9503         nk_ushort col[3];
9504         col[0] = (nk_ushort)(val[0]*(float)NK_USHORT_MAX);
9505         col[1] = (nk_ushort)(val[1]*(float)NK_USHORT_MAX);
9506         col[2] = (nk_ushort)(val[2]*(float)NK_USHORT_MAX);
9507         NK_MEMCPY(attr, col, sizeof(col));
9508     } break;
9509     case NK_FORMAT_R16G15B16A16: {
9510         nk_ushort col[4];
9511         col[0] = (nk_ushort)(val[0]*(float)NK_USHORT_MAX);
9512         col[1] = (nk_ushort)(val[1]*(float)NK_USHORT_MAX);
9513         col[2] = (nk_ushort)(val[2]*(float)NK_USHORT_MAX);
9514         col[3] = (nk_ushort)(val[3]*(float)NK_USHORT_MAX);
9515         NK_MEMCPY(attr, col, sizeof(col));
9516     } break;
9517     case NK_FORMAT_R32G32B32: {
9518         nk_uint col[3];
9519         col[0] = (nk_uint)(val[0]*(float)NK_UINT_MAX);
9520         col[1] = (nk_uint)(val[1]*(float)NK_UINT_MAX);
9521         col[2] = (nk_uint)(val[2]*(float)NK_UINT_MAX);
9522         NK_MEMCPY(attr, col, sizeof(col));
9523     } break;
9524     case NK_FORMAT_R32G32B32A32: {
9525         nk_uint col[4];
9526         col[0] = (nk_uint)(val[0]*(float)NK_UINT_MAX);
9527         col[1] = (nk_uint)(val[1]*(float)NK_UINT_MAX);
9528         col[2] = (nk_uint)(val[2]*(float)NK_UINT_MAX);
9529         col[3] = (nk_uint)(val[3]*(float)NK_UINT_MAX);
9530         NK_MEMCPY(attr, col, sizeof(col));
9531     } break;
9532     case NK_FORMAT_R32G32B32A32_FLOAT:
9533         NK_MEMCPY(attr, val, sizeof(float)*4);
9534         break;
9535     case NK_FORMAT_R32G32B32A32_DOUBLE: {
9536         double col[4];
9537         col[0] = (double)val[0];
9538         col[1] = (double)val[1];
9539         col[2] = (double)val[2];
9540         col[3] = (double)val[3];
9541         NK_MEMCPY(attr, col, sizeof(col));
9542     } break;
9543     case NK_FORMAT_RGB32:
9544     case NK_FORMAT_RGBA32: {
9545         struct nk_color col = nk_rgba_fv(val);
9546         nk_uint color = nk_color_u32(col);
9547         NK_MEMCPY(attr, &color, sizeof(color));
9548     } break; }
9549 }
9550 NK_INTERN void
9551 nk_draw_vertex_element(void *dst, const float *values, int value_count,
9552     enum nk_draw_vertex_layout_format format)
9553 {
9554     int value_index;
9555     void *attribute = dst;
```

```
9556     /* if this triggers you tried to provide a color format for a value */
9557     NK_ASSERT(format < NK_FORMAT_COLOR_BEGIN);
9558     if (format >= NK_FORMAT_COLOR_BEGIN && format <= NK_FORMAT_COLOR_END) return;
9559     for (value_index = 0; value_index < value_count; ++value_index) {
9560         switch (format) {
9561         default: NK_ASSERT(0 && "invalid vertex layout format"); break;
9562         case NK_FORMAT_SCHAR: {
9563             char value = (char)NK_CLAMP((float)NK_SCHAR_MIN, values[value_index], (float)NK_SCHAR_MAX);
9564             NK_MEMCPY(attribute, &value, sizeof(value));
9565             attribute = (void*)((char*)attribute + sizeof(char));
9566         } break;
9567         case NK_FORMAT_SSHORT: {
9568             nk_short value = (nk_short)NK_CLAMP((float)NK_SSHORT_MIN, values[value_index],
     (float)NK_SSHORT_MAX);
9569             NK_MEMCPY(attribute, &value, sizeof(value));
9570             attribute = (void*)((char*)attribute + sizeof(value));
9571         } break;
9572         case NK_FORMAT_SINT: {
9573             nk_int value = (nk_int)NK_CLAMP((float)NK_SINT_MIN, values[value_index],
     (float)NK_SINT_MAX);
9574             NK_MEMCPY(attribute, &value, sizeof(value));
9575             attribute = (void*)((char*)attribute + sizeof(nk_int));
9576         } break;
9577         case NK_FORMAT_UCHAR: {
9578             unsigned char value = (unsigned char)NK_CLAMP((float)NK_UCHAR_MIN, values[value_index],
     (float)NK_UCHAR_MAX);
9579             NK_MEMCPY(attribute, &value, sizeof(value));
9580             attribute = (void*)((char*)attribute + sizeof(unsigned char));
9581         } break;
9582         case NK_FORMAT_USHORT: {
9583             nk_ushort value = (nk_ushort)NK_CLAMP((float)NK_USHORT_MIN, values[value_index],
     (float)NK_USHORT_MAX);
9584             NK_MEMCPY(attribute, &value, sizeof(value));
9585             attribute = (void*)((char*)attribute + sizeof(value));
9586             } break;
9587         case NK_FORMAT_UINT: {
9588             nk_uint value = (nk_uint)NK_CLAMP((float)NK_UINT_MIN, values[value_index],
     (float)NK_UINT_MAX);
9589             NK_MEMCPY(attribute, &value, sizeof(value));
9590             attribute = (void*)((char*)attribute + sizeof(nk_uint));
9591         } break;
9592         case NK_FORMAT_FLOAT:
9593             NK_MEMCPY(attribute, &values[value_index], sizeof(values[value_index]));
9594             attribute = (void*)((char*)attribute + sizeof(float));
9595             break;
9596         case NK_FORMAT_DOUBLE: {
9597             double value = (double)values[value_index];
9598             NK_MEMCPY(attribute, &value, sizeof(value));
9599             attribute = (void*)((char*)attribute + sizeof(double));
9600             } break;
9601         }
9602     }
9603 }
9604 NK_INTERN void*
9605 nk_draw_vertex(void *dst, const struct nk_convert_config *config,
9606     struct nk_vec2 pos, struct nk_vec2 uv, struct nk_colorf color)
9607 {
9608     void *result = (void*)((char*)dst + config->vertex_size);
9609     const struct nk_draw_vertex_layout_element *elem_iter = config->vertex_layout;
9610     while (!nk_draw_vertex_layout_element_is_end_of_layout(elem_iter)) {
9611         void *address = (void*)((char*)dst + elem_iter->offset);
9612         switch (elem_iter->attribute) {
9613         case NK_VERTEX_ATTRIBUTE_COUNT:
9614         default: NK_ASSERT(0 && "wrong element attribute"); break;
9615         case NK_VERTEX_POSITION: nk_draw_vertex_element(address, &pos.x, 2, elem_iter->format); break;
9616         case NK_VERTEX_TEXCOORD: nk_draw_vertex_element(address, &uv.x, 2, elem_iter->format); break;
9617         case NK_VERTEX_COLOR: nk_draw_vertex_color(address, &color.r, elem_iter->format); break;
9618         }
9619         elem_iter++;
9620     }
9621     return result;
9622 }
9623 NK_API void
9624 nk_draw_list_stroke_poly_line(struct nk_draw_list *list, const struct nk_vec2 *points,
9625     const unsigned int points_count, struct nk_color color, enum nk_draw_list_stroke closed,
9626     float thickness, enum nk_anti_aliasing aliasing)
9627 {
9628     nk_size count;
9629     int thick_line;
9630     struct nk_colorf col;
9631     struct nk_colorf col_trans;
9632     NK_ASSERT(list);
9633     if (!list || points_count < 2) return;
9634
9635     color.a = (nk_byte)((float)color.a * list->config.global_alpha);
9636     count = points_count;
9637     if (!closed) count = points_count-1;
```

```
9638     thick_line = thickness > 1.0f;
9639
9640 #ifdef NK_INCLUDE_COMMAND_USERDATA
9641     nk_draw_list_push_userdata(list, list->userdata);
9642 #endif
9643
9644     color.a = (nk_byte)((float)color.a * list->config.global_alpha);
9645     nk_color_fv(&col.r, color);
9646     col_trans = col;
9647     col_trans.a = 0;
9648
9649     if (aliasing == NK_ANTI_ALIASING_ON) {
9650         /* ANTI-ALIASED STROKE */
9651         const float AA_SIZE = 1.0f;
9652         NK_STORAGE const nk_size pnt_align = NK_ALIGNOF(struct nk_vec2);
9653         NK_STORAGE const nk_size pnt_size = sizeof(struct nk_vec2);
9654
9655         /* allocate vertices and elements  */
9656         nk_size i1 = 0;
9657         nk_size vertex_offset;
9658         nk_size index = list->vertex_count;
9659
9660         const nk_size idx_count = (thick_line) ?  (count * 18) : (count * 12);
9661         const nk_size vtx_count = (thick_line) ? (points_count * 4): (points_count *3);
9662
9663         void *vtx = nk_draw_list_alloc_vertices(list, vtx_count);
9664         nk_draw_index *ids = nk_draw_list_alloc_elements(list, idx_count);
9665
9666         nk_size size;
9667         struct nk_vec2 *normals, *temp;
9668         if (!vtx || !ids) return;
9669
9670         /* temporary allocate normals + points */
9671         vertex_offset = (nk_size)((nk_byte*)vtx - (nk_byte*)list->vertices->memory.ptr);
9672         nk_buffer_mark(list->vertices, NK_BUFFER_FRONT);
9673         size = pnt_size * ((thick_line) ? 5 : 3) * points_count;
9674         normals = (struct nk_vec2*) nk_buffer_alloc(list->vertices, NK_BUFFER_FRONT, size, pnt_align);
9675         if (!normals) return;
9676         temp = normals + points_count;
9677
9678         /* make sure vertex pointer is still correct */
9679         vtx = (void*)((nk_byte*)list->vertices->memory.ptr + vertex_offset);
9680
9681         /* calculate normals */
9682         for (i1 = 0; i1 < count; ++i1) {
9683             const nk_size i2 = ((i1 + 1) == points_count) ? 0 : (i1 + 1);
9684             struct nk_vec2 diff = nk_vec2_sub(points[i2], points[i1]);
9685             float len;
9686
9687             /* vec2 inverted length  */
9688             len = nk_vec2_len_sqr(diff);
9689             if (len != 0.0f)
9690                 len = nk_inv_sqrt(len);
9691             else len = 1.0f;
9692
9693             diff = nk_vec2_muls(diff, len);
9694             normals[i1].x = diff.y;
9695             normals[i1].y = -diff.x;
9696         }
9697
9698         if (!closed)
9699             normals[points_count-1] = normals[points_count-2];
9700
9701         if (!thick_line) {
9702             nk_size idx1, i;
9703             if (!closed) {
9704                 struct nk_vec2 d;
9705                 temp[0] = nk_vec2_add(points[0], nk_vec2_muls(normals[0], AA_SIZE));
9706                 temp[1] = nk_vec2_sub(points[0], nk_vec2_muls(normals[0], AA_SIZE));
9707                 d = nk_vec2_muls(normals[points_count-1], AA_SIZE);
9708                 temp[(points_count-1) * 2 + 0] = nk_vec2_add(points[points_count-1], d);
9709                 temp[(points_count-1) * 2 + 1] = nk_vec2_sub(points[points_count-1], d);
9710             }
9711
9712             /* fill elements */
9713             idx1 = index;
9714             for (i1 = 0; i1 < count; i1++) {
9715                 struct nk_vec2 dm;
9716                 float dmr2;
9717                 nk_size i2 = ((i1 + 1) == points_count) ? 0 : (i1 + 1);
9718                 nk_size idx2 = ((i1+1) == points_count) ? index: (idx1 + 3);
9719
9720                 /* average normals */
9721                 dm = nk_vec2_muls(nk_vec2_add(normals[i1], normals[i2]), 0.5f);
9722                 dmr2 = dm.x * dm.x + dm.y* dm.y;
9723                 if (dmr2 > 0.000001f) {
9724                     float scale = 1.0f/dmr2;
```

```
9725                    scale = NK_MIN(100.0f, scale);
9726                    dm = nk_vec2_muls(dm, scale);
9727                }
9728
9729                dm = nk_vec2_muls(dm, AA_SIZE);
9730                temp[i2*2+0] = nk_vec2_add(points[i2], dm);
9731                temp[i2*2+1] = nk_vec2_sub(points[i2], dm);
9732
9733                ids[0] = (nk_draw_index)(idx2 + 0); ids[1] = (nk_draw_index)(idx1+0);
9734                ids[2] = (nk_draw_index)(idx1 + 2); ids[3] = (nk_draw_index)(idx1+2);
9735                ids[4] = (nk_draw_index)(idx2 + 2); ids[5] = (nk_draw_index)(idx2+0);
9736                ids[6] = (nk_draw_index)(idx2 + 1); ids[7] = (nk_draw_index)(idx1+1);
9737                ids[8] = (nk_draw_index)(idx1 + 0); ids[9] = (nk_draw_index)(idx1+0);
9738                ids[10]= (nk_draw_index)(idx2 + 0); ids[11]= (nk_draw_index)(idx2+1);
9739                ids += 12;
9740                idx1 = idx2;
9741            }
9742
9743            /* fill vertices */
9744            for (i = 0; i < points_count; ++i) {
9745                const struct nk_vec2 uv = list->config.null.uv;
9746                vtx = nk_draw_vertex(vtx, &list->config, points[i], uv, col);
9747                vtx = nk_draw_vertex(vtx, &list->config, temp[i*2+0], uv, col_trans);
9748                vtx = nk_draw_vertex(vtx, &list->config, temp[i*2+1], uv, col_trans);
9749            }
9750        } else {
9751            nk_size idx1, i;
9752            const float half_inner_thickness = (thickness - AA_SIZE) * 0.5f;
9753            if (!closed) {
9754                struct nk_vec2 d1 = nk_vec2_muls(normals[0], half_inner_thickness + AA_SIZE);
9755                struct nk_vec2 d2 = nk_vec2_muls(normals[0], half_inner_thickness);
9756
9757                temp[0] = nk_vec2_add(points[0], d1);
9758                temp[1] = nk_vec2_add(points[0], d2);
9759                temp[2] = nk_vec2_sub(points[0], d2);
9760                temp[3] = nk_vec2_sub(points[0], d1);
9761
9762                d1 = nk_vec2_muls(normals[points_count-1], half_inner_thickness + AA_SIZE);
9763                d2 = nk_vec2_muls(normals[points_count-1], half_inner_thickness);
9764
9765                temp[(points_count-1)*4+0] = nk_vec2_add(points[points_count-1], d1);
9766                temp[(points_count-1)*4+1] = nk_vec2_add(points[points_count-1], d2);
9767                temp[(points_count-1)*4+2] = nk_vec2_sub(points[points_count-1], d2);
9768                temp[(points_count-1)*4+3] = nk_vec2_sub(points[points_count-1], d1);
9769            }
9770
9771            /* add all elements */
9772            idx1 = index;
9773            for (i1 = 0; i1 < count; ++i1) {
9774                struct nk_vec2 dm_out, dm_in;
9775                const nk_size i2 = ((i1+1) == points_count) ? 0: (i1 + 1);
9776                nk_size idx2 = ((i1+1) == points_count) ? index: (idx1 + 4);
9777
9778                /* average normals */
9779                struct nk_vec2 dm = nk_vec2_muls(nk_vec2_add(normals[i1], normals[i2]), 0.5f);
9780                float dmr2 = dm.x * dm.x + dm.y* dm.y;
9781                if (dmr2 > 0.000001f) {
9782                    float scale = 1.0f/dmr2;
9783                    scale = NK_MIN(100.0f, scale);
9784                    dm = nk_vec2_muls(dm, scale);
9785                }
9786
9787                dm_out = nk_vec2_muls(dm, ((half_inner_thickness) + AA_SIZE));
9788                dm_in = nk_vec2_muls(dm, half_inner_thickness);
9789                temp[i2*4+0] = nk_vec2_add(points[i2], dm_out);
9790                temp[i2*4+1] = nk_vec2_add(points[i2], dm_in);
9791                temp[i2*4+2] = nk_vec2_sub(points[i2], dm_in);
9792                temp[i2*4+3] = nk_vec2_sub(points[i2], dm_out);
9793
9794                /* add indexes */
9795                ids[0] = (nk_draw_index)(idx2 + 1); ids[1] = (nk_draw_index)(idx1+1);
9796                ids[2] = (nk_draw_index)(idx1 + 2); ids[3] = (nk_draw_index)(idx1+2);
9797                ids[4] = (nk_draw_index)(idx2 + 2); ids[5] = (nk_draw_index)(idx2+1);
9798                ids[6] = (nk_draw_index)(idx2 + 1); ids[7] = (nk_draw_index)(idx1+1);
9799                ids[8] = (nk_draw_index)(idx1 + 0); ids[9] = (nk_draw_index)(idx1+0);
9800                ids[10]= (nk_draw_index)(idx2 + 0); ids[11] = (nk_draw_index)(idx2+1);
9801                ids[12]= (nk_draw_index)(idx2 + 2); ids[13] = (nk_draw_index)(idx1+2);
9802                ids[14]= (nk_draw_index)(idx1 + 3); ids[15] = (nk_draw_index)(idx1+3);
9803                ids[16]= (nk_draw_index)(idx2 + 3); ids[17] = (nk_draw_index)(idx2+2);
9804                ids += 18;
9805                idx1 = idx2;
9806            }
9807
9808            /* add vertices */
9809            for (i = 0; i < points_count; ++i) {
9810                const struct nk_vec2 uv = list->config.null.uv;
9811                vtx = nk_draw_vertex(vtx, &list->config, temp[i*4+0], uv, col_trans);
```

```
9812                 vtx = nk_draw_vertex(vtx, &list->config, temp[i*4+1], uv, col);
9813                 vtx = nk_draw_vertex(vtx, &list->config, temp[i*4+2], uv, col);
9814                 vtx = nk_draw_vertex(vtx, &list->config, temp[i*4+3], uv, col_trans);
9815             }
9816         }
9817         /* free temporary normals + points */
9818         nk_buffer_reset(list->vertices, NK_BUFFER_FRONT);
9819     } else {
9820         /* NON ANTI-ALIASED STROKE */
9821         nk_size i1 = 0;
9822         nk_size idx = list->vertex_count;
9823         const nk_size idx_count = count * 6;
9824         const nk_size vtx_count = count * 4;
9825         void *vtx = nk_draw_list_alloc_vertices(list, vtx_count);
9826         nk_draw_index *ids = nk_draw_list_alloc_elements(list, idx_count);
9827         if (!vtx || !ids) return;
9828
9829         for (i1 = 0; i1 < count; ++i1) {
9830             float dx, dy;
9831             const struct nk_vec2 uv = list->config.null.uv;
9832             const nk_size i2 = ((i1+1) == points_count) ? 0 : i1 + 1;
9833             const struct nk_vec2 p1 = points[i1];
9834             const struct nk_vec2 p2 = points[i2];
9835             struct nk_vec2 diff = nk_vec2_sub(p2, p1);
9836             float len;
9837
9838             /* vec2 inverted length  */
9839             len = nk_vec2_len_sqr(diff);
9840             if (len != 0.0f)
9841                 len = nk_inv_sqrt(len);
9842             else len = 1.0f;
9843             diff = nk_vec2_muls(diff, len);
9844
9845             /* add vertices */
9846             dx = diff.x * (thickness * 0.5f);
9847             dy = diff.y * (thickness * 0.5f);
9848
9849             vtx = nk_draw_vertex(vtx, &list->config, nk_vec2(p1.x + dy, p1.y - dx), uv, col);
9850             vtx = nk_draw_vertex(vtx, &list->config, nk_vec2(p2.x + dy, p2.y - dx), uv, col);
9851             vtx = nk_draw_vertex(vtx, &list->config, nk_vec2(p2.x - dy, p2.y + dx), uv, col);
9852             vtx = nk_draw_vertex(vtx, &list->config, nk_vec2(p1.x - dy, p1.y + dx), uv, col);
9853
9854             ids[0] = (nk_draw_index)(idx+0); ids[1] = (nk_draw_index)(idx+1);
9855             ids[2] = (nk_draw_index)(idx+2); ids[3] = (nk_draw_index)(idx+0);
9856             ids[4] = (nk_draw_index)(idx+2); ids[5] = (nk_draw_index)(idx+3);
9857
9858             ids += 6;
9859             idx += 4;
9860         }
9861     }
9862 }
9863 NK_API void
9864 nk_draw_list_fill_poly_convex(struct nk_draw_list *list,
9865     const struct nk_vec2 *points, const unsigned int points_count,
9866     struct nk_color color, enum nk_anti_aliasing aliasing)
9867 {
9868     struct nk_colorf col;
9869     struct nk_colorf col_trans;
9870
9871     NK_STORAGE const nk_size pnt_align = NK_ALIGNOF(struct nk_vec2);
9872     NK_STORAGE const nk_size pnt_size = sizeof(struct nk_vec2);
9873     NK_ASSERT(list);
9874     if (!list || points_count < 3) return;
9875
9876 #ifdef NK_INCLUDE_COMMAND_USERDATA
9877     nk_draw_list_push_userdata(list, list->userdata);
9878 #endif
9879
9880     color.a = (nk_byte)((float)color.a * list->config.global_alpha);
9881     nk_color_fv(&col.r, color);
9882     col_trans = col;
9883     col_trans.a = 0;
9884
9885     if (aliasing == NK_ANTI_ALIASING_ON) {
9886         nk_size i = 0;
9887         nk_size i0 = 0;
9888         nk_size i1 = 0;
9889
9890         const float AA_SIZE = 1.0f;
9891         nk_size vertex_offset = 0;
9892         nk_size index = list->vertex_count;
9893
9894         const nk_size idx_count = (points_count-2)*3 + points_count*6;
9895         const nk_size vtx_count = (points_count*2);
9896
9897         void *vtx = nk_draw_list_alloc_vertices(list, vtx_count);
9898         nk_draw_index *ids = nk_draw_list_alloc_elements(list, idx_count);
```

```
9899
9900        nk_size size = 0;
9901        struct nk_vec2 *normals = 0;
9902        unsigned int vtx_inner_idx = (unsigned int)(index + 0);
9903        unsigned int vtx_outer_idx = (unsigned int)(index + 1);
9904        if (!vtx || !ids) return;
9905
9906        /* temporary allocate normals */
9907        vertex_offset = (nk_size)((nk_byte*)vtx - (nk_byte*)list->vertices->memory.ptr);
9908        nk_buffer_mark(list->vertices, NK_BUFFER_FRONT);
9909        size = pnt_size * points_count;
9910        normals = (struct nk_vec2*) nk_buffer_alloc(list->vertices, NK_BUFFER_FRONT, size, pnt_align);
9911        if (!normals) return;
9912        vtx = (void*)((nk_byte*)list->vertices->memory.ptr + vertex_offset);
9913
9914        /* add elements */
9915        for (i = 2; i < points_count; i++) {
9916            ids[0] = (nk_draw_index)(vtx_inner_idx);
9917            ids[1] = (nk_draw_index)(vtx_inner_idx + ((i-1) % 1));
9918            ids[2] = (nk_draw_index)(vtx_inner_idx + (i % 1));
9919            ids += 3;
9920        }
9921
9922        /* compute normals */
9923        for (i0 = points_count-1, i1 = 0; i1 < points_count; i0 = i1++) {
9924            struct nk_vec2 p0 = points[i0];
9925            struct nk_vec2 p1 = points[i1];
9926            struct nk_vec2 diff = nk_vec2_sub(p1, p0);
9927
9928            /* vec2 inverted length  */
9929            float len = nk_vec2_len_sqr(diff);
9930            if (len != 0.0f)
9931                len = nk_inv_sqrt(len);
9932            else len = 1.0f;
9933            diff = nk_vec2_muls(diff, len);
9934
9935            normals[i0].x = diff.y;
9936            normals[i0].y = -diff.x;
9937        }
9938
9939        /* add vertices + indexes */
9940        for (i0 = points_count-1, i1 = 0; i1 < points_count; i0 = i1++) {
9941            const struct nk_vec2 uv = list->config.null.uv;
9942            struct nk_vec2 n0 = normals[i0];
9943            struct nk_vec2 n1 = normals[i1];
9944            struct nk_vec2 dm = nk_vec2_muls(nk_vec2_add(n0, n1), 0.5f);
9945            float dmr2 = dm.x*dm.x + dm.y*dm.y;
9946            if (dmr2 > 0.000001f) {
9947                float scale = 1.0f / dmr2;
9948                scale = NK_MIN(scale, 100.0f);
9949                dm = nk_vec2_muls(dm, scale);
9950            }
9951            dm = nk_vec2_muls(dm, AA_SIZE * 0.5f);
9952
9953            /* add vertices */
9954            vtx = nk_draw_vertex(vtx, &list->config, nk_vec2_sub(points[i1], dm), uv, col);
9955            vtx = nk_draw_vertex(vtx, &list->config, nk_vec2_add(points[i1], dm), uv, col_trans);
9956
9957            /* add indexes */
9958            ids[0] = (nk_draw_index)(vtx_inner_idx+(i1<1));
9959            ids[1] = (nk_draw_index)(vtx_inner_idx+(i0<1));
9960            ids[2] = (nk_draw_index)(vtx_outer_idx+(i0<1));
9961            ids[3] = (nk_draw_index)(vtx_outer_idx+(i0<1));
9962            ids[4] = (nk_draw_index)(vtx_outer_idx+(i1<1));
9963            ids[5] = (nk_draw_index)(vtx_inner_idx+(i1<1));
9964            ids += 6;
9965        }
9966        /* free temporary normals + points */
9967        nk_buffer_reset(list->vertices, NK_BUFFER_FRONT);
9968    } else {
9969        nk_size i = 0;
9970        nk_size index = list->vertex_count;
9971        const nk_size idx_count = (points_count-2)*3;
9972        const nk_size vtx_count = points_count;
9973        void *vtx = nk_draw_list_alloc_vertices(list, vtx_count);
9974        nk_draw_index *ids = nk_draw_list_alloc_elements(list, idx_count);
9975
9976        if (!vtx || !ids) return;
9977        for (i = 0; i < vtx_count; ++i)
9978            vtx = nk_draw_vertex(vtx, &list->config, points[i], list->config.null.uv, col);
9979        for (i = 2; i < points_count; ++i) {
9980            ids[0] = (nk_draw_index)index;
9981            ids[1] = (nk_draw_index)(index+ i - 1);
9982            ids[2] = (nk_draw_index)(index+i);
9983            ids += 3;
9984        }
9985    }
```

```
9986 }
9987 NK_API void
9988 nk_draw_list_path_clear(struct nk_draw_list *list)
9989 {
9990     NK_ASSERT(list);
9991     if (!list) return;
9992     nk_buffer_reset(list->buffer, NK_BUFFER_FRONT);
9993     list->path_count = 0;
9994     list->path_offset = 0;
9995 }
9996 NK_API void
9997 nk_draw_list_path_line_to(struct nk_draw_list *list, struct nk_vec2 pos)
9998 {
9999     struct nk_vec2 *points = 0;
10000     struct nk_draw_command *cmd = 0;
10001     NK_ASSERT(list);
10002     if (!list) return;
10003     if (!list->cmd_count)
10004         nk_draw_list_add_clip(list, nk_null_rect);
10005
10006     cmd = nk_draw_list_command_last(list);
10007     if (cmd && cmd->texture.ptr != list->config.null.texture.ptr)
10008         nk_draw_list_push_image(list, list->config.null.texture);
10009
10010     points = nk_draw_list_alloc_path(list, 1);
10011     if (!points) return;
10012     points[0] = pos;
10013 }
10014 NK_API void
10015 nk_draw_list_path_arc_to_fast(struct nk_draw_list *list, struct nk_vec2 center,
10016     float radius, int a_min, int a_max)
10017 {
10018     int a = 0;
10019     NK_ASSERT(list);
10020     if (!list) return;
10021     if (a_min <= a_max) {
10022         for (a = a_min; a <= a_max; a++) {
10023             const struct nk_vec2 c = list->circle_vtx[(nk_size)a % NK_LEN(list->circle_vtx)];
10024             const float x = center.x + c.x * radius;
10025             const float y = center.y + c.y * radius;
10026             nk_draw_list_path_line_to(list, nk_vec2(x, y));
10027         }
10028     }
10029 }
10030 NK_API void
10031 nk_draw_list_path_arc_to(struct nk_draw_list *list, struct nk_vec2 center,
10032     float radius, float a_min, float a_max, unsigned int segments)
10033 {
10034     unsigned int i = 0;
10035     NK_ASSERT(list);
10036     if (!list) return;
10037     if (radius == 0.0f) return;
10038
10039     /*  This algorithm for arc drawing relies on these two trigonometric identities[1]:
10040             sin(a + b) = sin(a) * cos(b) + cos(a) * sin(b)
10041             cos(a + b) = cos(a) * cos(b) - sin(a) * sin(b)
10042
10043         Two coordinates (x, y) of a point on a circle centered on
10044         the origin can be written in polar form as:
10045             x = r * cos(a)
10046             y = r * sin(a)
10047         where r is the radius of the circle,
10048             a is the angle between (x, y) and the origin.
10049
10050         This allows us to rotate the coordinates around the
10051         origin by an angle b using the following transformation:
10052             x' = r * cos(a + b) = x * cos(b) - y * sin(b)
10053             y' = r * sin(a + b) = y * cos(b) + x * sin(b)
10054
10055         [1]
      https://en.wikipedia.org/wiki/List_of_trigonometric_identities#Angle_sum_and_difference_identities
10056     */
10057     {const float d_angle = (a_max - a_min) / (float)segments;
10058     const float sin_d = (float)NK_SIN(d_angle);
10059     const float cos_d = (float)NK_COS(d_angle);
10060
10061     float cx = (float)NK_COS(a_min) * radius;
10062     float cy = (float)NK_SIN(a_min) * radius;
10063     for(i = 0; i <= segments; ++i) {
10064         float new_cx, new_cy;
10065         const float x = center.x + cx;
10066         const float y = center.y + cy;
10067         nk_draw_list_path_line_to(list, nk_vec2(x, y));
10068
10069         new_cx = cx * cos_d - cy * sin_d;
10070         new_cy = cy * cos_d + cx * sin_d;
10071         cx = new_cx;
```

```
10072         cy = new_cy;
10073     }}
10074 }
10075 NK_API void
10076 nk_draw_list_path_rect_to(struct nk_draw_list *list, struct nk_vec2 a,
10077     struct nk_vec2 b, float rounding)
10078 {
10079     float r;
10080     NK_ASSERT(list);
10081     if (!list) return;
10082     r = rounding;
10083     r = NK_MIN(r, ((b.x-a.x) < 0) ? -(b.x-a.x): (b.x-a.x));
10084     r = NK_MIN(r, ((b.y-a.y) < 0) ? -(b.y-a.y): (b.y-a.y));
10085
10086     if (r == 0.0f) {
10087         nk_draw_list_path_line_to(list, a);
10088         nk_draw_list_path_line_to(list, nk_vec2(b.x,a.y));
10089         nk_draw_list_path_line_to(list, b);
10090         nk_draw_list_path_line_to(list, nk_vec2(a.x,b.y));
10091     } else {
10092         nk_draw_list_path_arc_to_fast(list, nk_vec2(a.x + r, a.y + r), r, 6, 9);
10093         nk_draw_list_path_arc_to_fast(list, nk_vec2(b.x - r, a.y + r), r, 9, 12);
10094         nk_draw_list_path_arc_to_fast(list, nk_vec2(b.x - r, b.y - r), r, 0, 3);
10095         nk_draw_list_path_arc_to_fast(list, nk_vec2(a.x + r, b.y - r), r, 3, 6);
10096     }
10097 }
10098 NK_API void
10099 nk_draw_list_path_curve_to(struct nk_draw_list *list, struct nk_vec2 p2,
10100     struct nk_vec2 p3, struct nk_vec2 p4, unsigned int num_segments)
10101 {
10102     float t_step;
10103     unsigned int i_step;
10104     struct nk_vec2 p1;
10105
10106     NK_ASSERT(list);
10107     NK_ASSERT(list->path_count);
10108     if (!list || !list->path_count) return;
10109     num_segments = NK_MAX(num_segments, 1);
10110
10111     p1 = nk_draw_list_path_last(list);
10112     t_step = 1.0f/(float)num_segments;
10113     for (i_step = 1; i_step <= num_segments; ++i_step) {
10114         float t = t_step * (float)i_step;
10115         float u = 1.0f - t;
10116         float w1 = u*u*u;
10117         float w2 = 3*u*u*t;
10118         float w3 = 3*u*t*t;
10119         float w4 = t * t *t;
10120         float x = w1 * p1.x + w2 * p2.x + w3 * p3.x + w4 * p4.x;
10121         float y = w1 * p1.y + w2 * p2.y + w3 * p3.y + w4 * p4.y;
10122         nk_draw_list_path_line_to(list, nk_vec2(x,y));
10123     }
10124 }
10125 NK_API void
10126 nk_draw_list_path_fill(struct nk_draw_list *list, struct nk_color color)
10127 {
10128     struct nk_vec2 *points;
10129     NK_ASSERT(list);
10130     if (!list) return;
10131     points = (struct nk_vec2*)nk_buffer_memory(list->buffer);
10132     nk_draw_list_fill_poly_convex(list, points, list->path_count, color, list->config.shape_AA);
10133     nk_draw_list_path_clear(list);
10134 }
10135 NK_API void
10136 nk_draw_list_path_stroke(struct nk_draw_list *list, struct nk_color color,
10137     enum nk_draw_list_stroke closed, float thickness)
10138 {
10139     struct nk_vec2 *points;
10140     NK_ASSERT(list);
10141     if (!list) return;
10142     points = (struct nk_vec2*)nk_buffer_memory(list->buffer);
10143     nk_draw_list_stroke_poly_line(list, points, list->path_count, color,
10144         closed, thickness, list->config.line_AA);
10145     nk_draw_list_path_clear(list);
10146 }
10147 NK_API void
10148 nk_draw_list_stroke_line(struct nk_draw_list *list, struct nk_vec2 a,
10149     struct nk_vec2 b, struct nk_color col, float thickness)
10150 {
10151     NK_ASSERT(list);
10152     if (!list || !col.a) return;
10153     if (list->line_AA == NK_ANTI_ALIASING_ON) {
10154         nk_draw_list_path_line_to(list, a);
10155         nk_draw_list_path_line_to(list, b);
10156     } else {
10157         nk_draw_list_path_line_to(list, nk_vec2_sub(a,nk_vec2(0.5f,0.5f)));
10158         nk_draw_list_path_line_to(list, nk_vec2_sub(b,nk_vec2(0.5f,0.5f)));
```

```
10159     }
10160     nk_draw_list_path_stroke(list,  col, NK_STROKE_OPEN, thickness);
10161 }
10162 NK_API void
10163 nk_draw_list_fill_rect(struct nk_draw_list *list, struct nk_rect rect,
10164     struct nk_color col, float rounding)
10165 {
10166     NK_ASSERT(list);
10167     if (!list || !col.a) return;
10168
10169     if (list->line_AA == NK_ANTI_ALIASING_ON) {
10170         nk_draw_list_path_rect_to(list, nk_vec2(rect.x, rect.y),
10171             nk_vec2(rect.x + rect.w, rect.y + rect.h), rounding);
10172     } else {
10173         nk_draw_list_path_rect_to(list, nk_vec2(rect.x-0.5f, rect.y-0.5f),
10174             nk_vec2(rect.x + rect.w, rect.y + rect.h), rounding);
10175     } nk_draw_list_path_fill(list,  col);
10176 }
10177 NK_API void
10178 nk_draw_list_stroke_rect(struct nk_draw_list *list, struct nk_rect rect,
10179     struct nk_color col, float rounding, float thickness)
10180 {
10181     NK_ASSERT(list);
10182     if (!list || !col.a) return;
10183     if (list->line_AA == NK_ANTI_ALIASING_ON) {
10184         nk_draw_list_path_rect_to(list, nk_vec2(rect.x, rect.y),
10185             nk_vec2(rect.x + rect.w, rect.y + rect.h), rounding);
10186     } else {
10187         nk_draw_list_path_rect_to(list, nk_vec2(rect.x-0.5f, rect.y-0.5f),
10188             nk_vec2(rect.x + rect.w, rect.y + rect.h), rounding);
10189     } nk_draw_list_path_stroke(list,  col, NK_STROKE_CLOSED, thickness);
10190 }
10191 NK_API void
10192 nk_draw_list_fill_rect_multi_color(struct nk_draw_list *list, struct nk_rect rect,
10193     struct nk_color left, struct nk_color top, struct nk_color right,
10194     struct nk_color bottom)
10195 {
10196     void *vtx;
10197     struct nk_colorf col_left, col_top;
10198     struct nk_colorf col_right, col_bottom;
10199     nk_draw_index *idx;
10200     nk_draw_index index;
10201
10202     nk_color_fv(&col_left.r, left);
10203     nk_color_fv(&col_right.r, right);
10204     nk_color_fv(&col_top.r, top);
10205     nk_color_fv(&col_bottom.r, bottom);
10206
10207     NK_ASSERT(list);
10208     if (!list) return;
10209
10210     nk_draw_list_push_image(list, list->config.null.texture);
10211     index = (nk_draw_index)list->vertex_count;
10212     vtx = nk_draw_list_alloc_vertices(list, 4);
10213     idx = nk_draw_list_alloc_elements(list, 6);
10214     if (!vtx || !idx) return;
10215
10216     idx[0] = (nk_draw_index)(index+0); idx[1] = (nk_draw_index)(index+1);
10217     idx[2] = (nk_draw_index)(index+2); idx[3] = (nk_draw_index)(index+0);
10218     idx[4] = (nk_draw_index)(index+2); idx[5] = (nk_draw_index)(index+3);
10219
10220     vtx = nk_draw_vertex(vtx, &list->config, nk_vec2(rect.x, rect.y), list->config.null.uv, col_left);
10221     vtx = nk_draw_vertex(vtx, &list->config, nk_vec2(rect.x + rect.w, rect.y), list->config.null.uv,
      col_top);
10222     vtx = nk_draw_vertex(vtx, &list->config, nk_vec2(rect.x + rect.w, rect.y + rect.h),
      list->config.null.uv, col_right);
10223     vtx = nk_draw_vertex(vtx, &list->config, nk_vec2(rect.x, rect.y + rect.h), list->config.null.uv,
      col_bottom);
10224 }
10225 NK_API void
10226 nk_draw_list_fill_triangle(struct nk_draw_list *list, struct nk_vec2 a,
10227     struct nk_vec2 b, struct nk_vec2 c, struct nk_color col)
10228 {
10229     NK_ASSERT(list);
10230     if (!list || !col.a) return;
10231     nk_draw_list_path_line_to(list, a);
10232     nk_draw_list_path_line_to(list, b);
10233     nk_draw_list_path_line_to(list, c);
10234     nk_draw_list_path_fill(list, col);
10235 }
10236 NK_API void
10237 nk_draw_list_stroke_triangle(struct nk_draw_list *list, struct nk_vec2 a,
10238     struct nk_vec2 b, struct nk_vec2 c, struct nk_color col, float thickness)
10239 {
10240     NK_ASSERT(list);
10241     if (!list || !col.a) return;
10242     nk_draw_list_path_line_to(list, a);
```

```
10243     nk_draw_list_path_line_to(list, b);
10244     nk_draw_list_path_line_to(list, c);
10245     nk_draw_list_path_stroke(list, col, NK_STROKE_CLOSED, thickness);
10246 }
10247 NK_API void
10248 nk_draw_list_fill_circle(struct nk_draw_list *list, struct nk_vec2 center,
10249     float radius, struct nk_color col, unsigned int segs)
10250 {
10251     float a_max;
10252     NK_ASSERT(list);
10253     if (!list || !col.a) return;
10254     a_max = NK_PI * 2.0f * ((float)segs - 1.0f) / (float)segs;
10255     nk_draw_list_path_arc_to(list, center, radius, 0.0f, a_max, segs);
10256     nk_draw_list_path_fill(list, col);
10257 }
10258 NK_API void
10259 nk_draw_list_stroke_circle(struct nk_draw_list *list, struct nk_vec2 center,
10260     float radius, struct nk_color col, unsigned int segs, float thickness)
10261 {
10262     float a_max;
10263     NK_ASSERT(list);
10264     if (!list || !col.a) return;
10265     a_max = NK_PI * 2.0f * ((float)segs - 1.0f) / (float)segs;
10266     nk_draw_list_path_arc_to(list, center, radius, 0.0f, a_max, segs);
10267     nk_draw_list_path_stroke(list, col, NK_STROKE_CLOSED, thickness);
10268 }
10269 NK_API void
10270 nk_draw_list_stroke_curve(struct nk_draw_list *list, struct nk_vec2 p0,
10271     struct nk_vec2 cp0, struct nk_vec2 cp1, struct nk_vec2 p1,
10272     struct nk_color col, unsigned int segments, float thickness)
10273 {
10274     NK_ASSERT(list);
10275     if (!list || !col.a) return;
10276     nk_draw_list_path_line_to(list, p0);
10277     nk_draw_list_path_curve_to(list, cp0, cp1, p1, segments);
10278     nk_draw_list_path_stroke(list, col, NK_STROKE_OPEN, thickness);
10279 }
10280 NK_INTERN void
10281 nk_draw_list_push_rect_uv(struct nk_draw_list *list, struct nk_vec2 a,
10282     struct nk_vec2 c, struct nk_vec2 uva, struct nk_vec2 uvc,
10283     struct nk_color color)
10284 {
10285     void *vtx;
10286     struct nk_vec2 uvb;
10287     struct nk_vec2 uvd;
10288     struct nk_vec2 b;
10289     struct nk_vec2 d;
10290
10291     struct nk_colorf col;
10292     nk_draw_index *idx;
10293     nk_draw_index index;
10294     NK_ASSERT(list);
10295     if (!list) return;
10296
10297     nk_color_fv(&col.r, color);
10298     uvb = nk_vec2(uvc.x, uva.y);
10299     uvd = nk_vec2(uva.x, uvc.y);
10300     b = nk_vec2(c.x, a.y);
10301     d = nk_vec2(a.x, c.y);
10302
10303     index = (nk_draw_index)list->vertex_count;
10304     vtx = nk_draw_list_alloc_vertices(list, 4);
10305     idx = nk_draw_list_alloc_elements(list, 6);
10306     if (!vtx || !idx) return;
10307
10308     idx[0] = (nk_draw_index)(index+0); idx[1] = (nk_draw_index)(index+1);
10309     idx[2] = (nk_draw_index)(index+2); idx[3] = (nk_draw_index)(index+0);
10310     idx[4] = (nk_draw_index)(index+2); idx[5] = (nk_draw_index)(index+3);
10311
10312     vtx = nk_draw_vertex(vtx, &list->config, a, uva, col);
10313     vtx = nk_draw_vertex(vtx, &list->config, b, uvb, col);
10314     vtx = nk_draw_vertex(vtx, &list->config, c, uvc, col);
10315     vtx = nk_draw_vertex(vtx, &list->config, d, uvd, col);
10316 }
10317 NK_API void
10318 nk_draw_list_add_image(struct nk_draw_list *list, struct nk_image texture,
10319     struct nk_rect rect, struct nk_color color)
10320 {
10321     NK_ASSERT(list);
10322     if (!list) return;
10323     /* push new command with given texture */
10324     nk_draw_list_push_image(list, texture.handle);
10325     if (nk_image_is_subimage(&texture)) {
10326         /* add region inside of the texture  */
10327         struct nk_vec2 uv[2];
10328         uv[0].x = (float)texture.region[0]/(float)texture.w;
10329         uv[0].y = (float)texture.region[1]/(float)texture.h;
```

```
10330         uv[1].x = (float)(texture.region[0] + texture.region[2])/(float)texture.w;
10331         uv[1].y = (float)(texture.region[1] + texture.region[3])/(float)texture.h;
10332         nk_draw_list_push_rect_uv(list, nk_vec2(rect.x, rect.y),
10333             nk_vec2(rect.x + rect.w, rect.y + rect.h),  uv[0], uv[1], color);
10334     } else nk_draw_list_push_rect_uv(list, nk_vec2(rect.x, rect.y),
10335             nk_vec2(rect.x + rect.w, rect.y + rect.h),
10336             nk_vec2(0.0f, 0.0f), nk_vec2(1.0f, 1.0f),color);
10337 }
10338 NK_API void
10339 nk_draw_list_add_text(struct nk_draw_list *list, const struct nk_user_font *font,
10340     struct nk_rect rect, const char *text, int len, float font_height,
10341     struct nk_color fg)
10342 {
10343     float x = 0;
10344     int text_len = 0;
10345     nk_rune unicode = 0;
10346     nk_rune next = 0;
10347     int glyph_len = 0;
10348     int next_glyph_len = 0;
10349     struct nk_user_font_glyph g;
10350
10351     NK_ASSERT(list);
10352     if (!list || !len || !text) return;
10353     if (!NK_INTERSECT(rect.x, rect.y, rect.w, rect.h,
10354         list->clip_rect.x, list->clip_rect.y, list->clip_rect.w, list->clip_rect.h)) return;
10355
10356     nk_draw_list_push_image(list, font->texture);
10357     x = rect.x;
10358     glyph_len = nk_utf_decode(text, &unicode, len);
10359     if (!glyph_len) return;
10360
10361     /* draw every glyph image */
10362     fg.a = (nk_byte)((float)fg.a * list->config.global_alpha);
10363     while (text_len < len && glyph_len) {
10364         float gx, gy, gh, gw;
10365         float char_width = 0;
10366         if (unicode == NK_UTF_INVALID) break;
10367
10368         /* query currently drawn glyph information */
10369         next_glyph_len = nk_utf_decode(text + text_len + glyph_len, &next, (int)len - text_len);
10370         font->query(font->userdata, font_height, &g, unicode,
10371                     (next == NK_UTF_INVALID) ? '\0' : next);
10372
10373         /* calculate and draw glyph drawing rectangle and image */
10374         gx = x + g.offset.x;
10375         gy = rect.y + g.offset.y;
10376         gw = g.width; gh = g.height;
10377         char_width = g.xadvance;
10378         nk_draw_list_push_rect_uv(list, nk_vec2(gx,gy), nk_vec2(gx + gw, gy+ gh),
10379             g.uv[0], g.uv[1], fg);
10380
10381         /* offset next glyph */
10382         text_len += glyph_len;
10383         x += char_width;
10384         glyph_len = next_glyph_len;
10385         unicode = next;
10386     }
10387 }
10388 NK_API nk_flags
10389 nk_convert(struct nk_context *ctx, struct nk_buffer *cmds,
10390     struct nk_buffer *vertices, struct nk_buffer *elements,
10391     const struct nk_convert_config *config)
10392 {
10393     nk_flags res = NK_CONVERT_SUCCESS;
10394     const struct nk_command *cmd;
10395     NK_ASSERT(ctx);
10396     NK_ASSERT(cmds);
10397     NK_ASSERT(vertices);
10398     NK_ASSERT(elements);
10399     NK_ASSERT(config);
10400     NK_ASSERT(config->vertex_layout);
10401     NK_ASSERT(config->vertex_size);
10402     if (!ctx || !cmds || !vertices || !elements || !config || !config->vertex_layout)
10403         return NK_CONVERT_INVALID_PARAM;
10404
10405     nk_draw_list_setup(&ctx->draw_list, config, cmds, vertices, elements,
10406         config->line_AA, config->shape_AA);
10407     nk_foreach(cmd, ctx)
10408     {
10409 #ifdef NK_INCLUDE_COMMAND_USERDATA
10410         ctx->draw_list.userdata = cmd->userdata;
10411 #endif
10412         switch (cmd->type) {
10413         case NK_COMMAND_NOP: break;
10414         case NK_COMMAND_SCISSOR: {
10415             const struct nk_command_scissor *s = (const struct nk_command_scissor*)cmd;
10416             nk_draw_list_add_clip(&ctx->draw_list, nk_rect(s->x, s->y, s->w, s->h));
```

```
10417         } break;
10418         case NK_COMMAND_LINE: {
10419             const struct nk_command_line *l = (const struct nk_command_line*)cmd;
10420             nk_draw_list_stroke_line(&ctx->draw_list, nk_vec2(l->begin.x, l->begin.y),
10421                 nk_vec2(l->end.x, l->end.y), l->color, l->line_thickness);
10422         } break;
10423         case NK_COMMAND_CURVE: {
10424             const struct nk_command_curve *q = (const struct nk_command_curve*)cmd;
10425             nk_draw_list_stroke_curve(&ctx->draw_list, nk_vec2(q->begin.x, q->begin.y),
10426                 nk_vec2(q->ctrl[0].x, q->ctrl[0].y), nk_vec2(q->ctrl[1].x,
10427                 q->ctrl[1].y), nk_vec2(q->end.x, q->end.y), q->color,
10428                 config->curve_segment_count, q->line_thickness);
10429         } break;
10430         case NK_COMMAND_RECT: {
10431             const struct nk_command_rect *r = (const struct nk_command_rect*)cmd;
10432             nk_draw_list_stroke_rect(&ctx->draw_list, nk_rect(r->x, r->y, r->w, r->h),
10433                 r->color, (float)r->rounding, r->line_thickness);
10434         } break;
10435         case NK_COMMAND_RECT_FILLED: {
10436             const struct nk_command_rect_filled *r = (const struct nk_command_rect_filled*)cmd;
10437             nk_draw_list_fill_rect(&ctx->draw_list, nk_rect(r->x, r->y, r->w, r->h),
10438                 r->color, (float)r->rounding);
10439         } break;
10440         case NK_COMMAND_RECT_MULTI_COLOR: {
10441             const struct nk_command_rect_multi_color *r = (const struct
    nk_command_rect_multi_color*)cmd;
10442             nk_draw_list_fill_rect_multi_color(&ctx->draw_list, nk_rect(r->x, r->y, r->w, r->h),
10443                 r->left, r->top, r->right, r->bottom);
10444         } break;
10445         case NK_COMMAND_CIRCLE: {
10446             const struct nk_command_circle *c = (const struct nk_command_circle*)cmd;
10447             nk_draw_list_stroke_circle(&ctx->draw_list, nk_vec2((float)c->x + (float)c->w/2,
10448                 (float)c->y + (float)c->h/2), (float)c->w/2, c->color,
10449                 config->circle_segment_count, c->line_thickness);
10450         } break;
10451         case NK_COMMAND_CIRCLE_FILLED: {
10452             const struct nk_command_circle_filled *c = (const struct nk_command_circle_filled *)cmd;
10453             nk_draw_list_fill_circle(&ctx->draw_list, nk_vec2((float)c->x + (float)c->w/2,
10454                 (float)c->y + (float)c->h/2), (float)c->w/2, c->color,
10455                 config->circle_segment_count);
10456         } break;
10457         case NK_COMMAND_ARC: {
10458             const struct nk_command_arc *c = (const struct nk_command_arc*)cmd;
10459             nk_draw_list_path_line_to(&ctx->draw_list, nk_vec2(c->cx, c->cy));
10460             nk_draw_list_path_arc_to(&ctx->draw_list, nk_vec2(c->cx, c->cy), c->r,
10461                 c->a[0], c->a[1], config->arc_segment_count);
10462             nk_draw_list_path_stroke(&ctx->draw_list, c->color, NK_STROKE_CLOSED, c->line_thickness);
10463         } break;
10464         case NK_COMMAND_ARC_FILLED: {
10465             const struct nk_command_arc_filled *c = (const struct nk_command_arc_filled*)cmd;
10466             nk_draw_list_path_line_to(&ctx->draw_list, nk_vec2(c->cx, c->cy));
10467             nk_draw_list_path_arc_to(&ctx->draw_list, nk_vec2(c->cx, c->cy), c->r,
10468                 c->a[0], c->a[1], config->arc_segment_count);
10469             nk_draw_list_path_fill(&ctx->draw_list, c->color);
10470         } break;
10471         case NK_COMMAND_TRIANGLE: {
10472             const struct nk_command_triangle *t = (const struct nk_command_triangle*)cmd;
10473             nk_draw_list_stroke_triangle(&ctx->draw_list, nk_vec2(t->a.x, t->a.y),
10474                 nk_vec2(t->b.x, t->b.y), nk_vec2(t->c.x, t->c.y), t->color,
10475                 t->line_thickness);
10476         } break;
10477         case NK_COMMAND_TRIANGLE_FILLED: {
10478             const struct nk_command_triangle_filled *t = (const struct
    nk_command_triangle_filled*)cmd;
10479             nk_draw_list_fill_triangle(&ctx->draw_list, nk_vec2(t->a.x, t->a.y),
10480                 nk_vec2(t->b.x, t->b.y), nk_vec2(t->c.x, t->c.y), t->color);
10481         } break;
10482         case NK_COMMAND_POLYGON: {
10483             int i;
10484             const struct nk_command_polygon*p = (const struct nk_command_polygon*)cmd;
10485             for (i = 0; i < p->point_count; ++i) {
10486                 struct nk_vec2 pnt = nk_vec2((float)p->points[i].x, (float)p->points[i].y);
10487                 nk_draw_list_path_line_to(&ctx->draw_list, pnt);
10488             }
10489             nk_draw_list_path_stroke(&ctx->draw_list, p->color, NK_STROKE_CLOSED, p->line_thickness);
10490         } break;
10491         case NK_COMMAND_POLYGON_FILLED: {
10492             int i;
10493             const struct nk_command_polygon_filled *p = (const struct nk_command_polygon_filled*)cmd;
10494             for (i = 0; i < p->point_count; ++i) {
10495                 struct nk_vec2 pnt = nk_vec2((float)p->points[i].x, (float)p->points[i].y);
10496                 nk_draw_list_path_line_to(&ctx->draw_list, pnt);
10497             }
10498             nk_draw_list_path_fill(&ctx->draw_list, p->color);
10499         } break;
10500         case NK_COMMAND_POLYLINE: {
10501             int i;
```

```
10502             const struct nk_command_polyline *p = (const struct nk_command_polyline*)cmd;
10503             for (i = 0; i < p->point_count; ++i) {
10504                 struct nk_vec2 pnt = nk_vec2((float)p->points[i].x, (float)p->points[i].y);
10505                 nk_draw_list_path_line_to(&ctx->draw_list, pnt);
10506             }
10507             nk_draw_list_path_stroke(&ctx->draw_list, p->color, NK_STROKE_OPEN, p->line_thickness);
10508         } break;
10509         case NK_COMMAND_TEXT: {
10510             const struct nk_command_text *t = (const struct nk_command_text*)cmd;
10511             nk_draw_list_add_text(&ctx->draw_list, t->font, nk_rect(t->x, t->y, t->w, t->h),
10512                 t->string, t->length, t->height, t->foreground);
10513         } break;
10514         case NK_COMMAND_IMAGE: {
10515             const struct nk_command_image *i = (const struct nk_command_image*)cmd;
10516             nk_draw_list_add_image(&ctx->draw_list, i->img, nk_rect(i->x, i->y, i->w, i->h), i->col);
10517         } break;
10518         case NK_COMMAND_CUSTOM: {
10519             const struct nk_command_custom *c = (const struct nk_command_custom*)cmd;
10520             c->callback(&ctx->draw_list, c->x, c->y, c->w, c->h, c->callback_data);
10521         } break;
10522         default: break;
10523         }
10524     }
10525     res |= (cmds->needed > cmds->allocated + (cmds->memory.size - cmds->size)) ?
      NK_CONVERT_COMMAND_BUFFER_FULL: 0;
10526     res |= (vertices->needed > vertices->allocated) ? NK_CONVERT_VERTEX_BUFFER_FULL: 0;
10527     res |= (elements->needed > elements->allocated) ? NK_CONVERT_ELEMENT_BUFFER_FULL: 0;
10528     return res;
10529 }
10530 NK_API const struct nk_draw_command*
10531 nk__draw_begin(const struct nk_context *ctx,
10532     const struct nk_buffer *buffer)
10533 {
10534     return nk__draw_list_begin(&ctx->draw_list, buffer);
10535 }
10536 NK_API const struct nk_draw_command*
10537 nk__draw_end(const struct nk_context *ctx, const struct nk_buffer *buffer)
10538 {
10539     return nk__draw_list_end(&ctx->draw_list, buffer);
10540 }
10541 NK_API const struct nk_draw_command*
10542 nk__draw_next(const struct nk_draw_command *cmd,
10543     const struct nk_buffer *buffer, const struct nk_context *ctx)
10544 {
10545     return nk__draw_list_next(cmd, buffer, &ctx->draw_list);
10546 }
10547 #endif
10548
10549
10550
10551
10552
10553 #ifdef NK_INCLUDE_FONT_BAKING
10554 /* ---------------------------------------------------------------
10555  *
10556  *                          RECT PACK
10557  *
10558  * ---------------------------------------------------------------*/
10559 /* stb_rect_pack.h - v0.05 - public domain - rectangle packing */
10560 /* Sean Barrett 2014 */
10561 #define NK_RP__MAXVAL  0xffff
10562 typedef unsigned short nk_rp_coord;
10563
10564 struct nk_rp_rect {
10565     /* reserved for your use: */
10566     int id;
10567     /* input: */
10568     nk_rp_coord w, h;
10569     /* output: */
10570     nk_rp_coord x, y;
10571     int was_packed;
10572     /* non-zero if valid packing */
10573 }; /* 16 bytes, nominally */
10574
10575 struct nk_rp_node {
10576     nk_rp_coord  x,y;
10577     struct nk_rp_node  *next;
10578 };
10579
10580 struct nk_rp_context {
10581     int width;
10582     int height;
10583     int align;
10584     int init_mode;
10585     int heuristic;
10586     int num_nodes;
10587     struct nk_rp_node *active_head;
```

```
10588     struct nk_rp_node *free_head;
10589     struct nk_rp_node extra[2];
10590     /* we allocate two extra nodes so optimal user-node-count is 'width' not 'width+2' */
10591 };
10592
10593 struct nk_rp__findresult {
10594     int x,y;
10595     struct nk_rp_node **prev_link;
10596 };
10597
10598 enum NK_RP_HEURISTIC {
10599     NK_RP_HEURISTIC_Skyline_default=0,
10600     NK_RP_HEURISTIC_Skyline_BL_sortHeight = NK_RP_HEURISTIC_Skyline_default,
10601     NK_RP_HEURISTIC_Skyline_BF_sortHeight
10602 };
10603 enum NK_RP_INIT_STATE{NK_RP__INIT_skyline = 1};
10604
10605 NK_INTERN void
10606 nk_rp_setup_allow_out_of_mem(struct nk_rp_context *context, int allow_out_of_mem)
10607 {
10608     if (allow_out_of_mem)
10609         /* if it's ok to run out of memory, then don't bother aligning them; */
10610         /* this gives better packing, but may fail due to OOM (even though */
10611         /* the rectangles easily fit). @TODO a smarter approach would be to only */
10612         /* quantize once we've hit OOM, then we could get rid of this parameter. */
10613         context->align = 1;
10614     else {
10615         /* if it's not ok to run out of memory, then quantize the widths */
10616         /* so that num_nodes is always enough nodes. */
10617         /* */
10618         /* I.e. num_nodes * align >= width */
10619         /*                  align >= width / num_nodes */
10620         /*                  align = ceil(width/num_nodes) */
10621         context->align = (context->width + context->num_nodes-1) / context->num_nodes;
10622     }
10623 }
10624 NK_INTERN void
10625 nk_rp_init_target(struct nk_rp_context *context, int width, int height,
10626     struct nk_rp_node *nodes, int num_nodes)
10627 {
10628     int i;
10629 #ifndef STBRP_LARGE_RECTS
10630     NK_ASSERT(width <= 0xffff && height <= 0xffff);
10631 #endif
10632
10633     for (i=0; i < num_nodes-1; ++i)
10634         nodes[i].next = &nodes[i+1];
10635     nodes[i].next = 0;
10636     context->init_mode = NK_RP__INIT_skyline;
10637     context->heuristic = NK_RP_HEURISTIC_Skyline_default;
10638     context->free_head = &nodes[0];
10639     context->active_head = &context->extra[0];
10640     context->width = width;
10641     context->height = height;
10642     context->num_nodes = num_nodes;
10643     nk_rp_setup_allow_out_of_mem(context, 0);
10644
10645     /* node 0 is the full width, node 1 is the sentinel (lets us not store width explicitly) */
10646     context->extra[0].x = 0;
10647     context->extra[0].y = 0;
10648     context->extra[0].next = &context->extra[1];
10649     context->extra[1].x = (nk_rp_coord) width;
10650     context->extra[1].y = 65535;
10651     context->extra[1].next = 0;
10652 }
10653 /* find minimum y position if it starts at x1 */
10654 NK_INTERN int
10655 nk_rp__skyline_find_min_y(struct nk_rp_context *c, struct nk_rp_node *first,
10656     int x0, int width, int *pwaste)
10657 {
10658     struct nk_rp_node *node = first;
10659     int x1 = x0 + width;
10660     int min_y, visited_width, waste_area;
10661     NK_ASSERT(first->x <= x0);
10662     NK_UNUSED(c);
10663
10664     NK_ASSERT(node->next->x > x0);
10665     /* we ended up handling this in the caller for efficiency */
10666     NK_ASSERT(node->x <= x0);
10667
10668     min_y = 0;
10669     waste_area = 0;
10670     visited_width = 0;
10671     while (node->x < x1)
10672     {
10673         if (node->y > min_y) {
10674             /* raise min_y higher. */
```

```
10675             /* we've accounted for all waste up to min_y, */
10676             /* but we'll now add more waste for everything we've visited */
10677             waste_area += visited_width * (node->y - min_y);
10678             min_y = node->y;
10679             /* the first time through, visited_width might be reduced */
10680             if (node->x < x0)
10681             visited_width += node->next->x - x0;
10682             else
10683             visited_width += node->next->x - node->x;
10684         } else {
10685             /* add waste area */
10686             int under_width = node->next->x - node->x;
10687             if (under_width + visited_width > width)
10688             under_width = width - visited_width;
10689             waste_area += under_width * (min_y - node->y);
10690             visited_width += under_width;
10691         }
10692         node = node->next;
10693     }
10694     *pwaste = waste_area;
10695     return min_y;
10696 }
10697 NK_INTERN struct nk_rp__findresult
10698 nk_rp__skyline_find_best_pos(struct nk_rp_context *c, int width, int height)
10699 {
10700     int best_waste = (1«30), best_x, best_y = (1 « 30);
10701     struct nk_rp__findresult fr;
10702     struct nk_rp_node **prev, *node, *tail, **best = 0;
10703
10704     /* align to multiple of c->align */
10705     width = (width + c->align - 1);
10706     width -= width % c->align;
10707     NK_ASSERT(width % c->align == 0);
10708
10709     node = c->active_head;
10710     prev = &c->active_head;
10711     while (node->x + width <= c->width) {
10712         int y,waste;
10713         y = nk_rp__skyline_find_min_y(c, node, node->x, width, &waste);
10714         /* actually just want to test BL */
10715         if (c->heuristic == NK_RP_HEURISTIC_Skyline_BL_sortHeight) {
10716             /* bottom left */
10717             if (y < best_y) {
10718             best_y = y;
10719             best = prev;
10720             }
10721         } else {
10722             /* best-fit */
10723             if (y + height <= c->height) {
10724                 /* can only use it if it first vertically */
10725                 if (y < best_y || (y == best_y && waste < best_waste)) {
10726                     best_y = y;
10727                     best_waste = waste;
10728                     best = prev;
10729                 }
10730             }
10731         }
10732         prev = &node->next;
10733         node = node->next;
10734     }
10735     best_x = (best == 0) ? 0 : (*best)->x;
10736
10737     /* if doing best-fit (BF), we also have to try aligning right edge to each node position */
10738     /* */
10739     /* e.g, if fitting */
10740     /* */
10741     /*     ____________________ */
10742     /*    |____________________| */
10743     /* */
10744     /*            into */
10745     /* */
10746     /*   |                          | */
10747     /*   |              ____________| */
10748     /*   |_____________| */
10749     /* */
10750     /* then right-aligned reduces waste, but bottom-left BL is always chooses left-aligned */
10751     /* */
10752     /* This makes BF take about 2x the time */
10753     if (c->heuristic == NK_RP_HEURISTIC_Skyline_BF_sortHeight)
10754     {
10755         tail = c->active_head;
10756         node = c->active_head;
10757         prev = &c->active_head;
10758         /* find first node that's admissible */
10759         while (tail->x < width)
10760             tail = tail->next;
10761         while (tail)
```

```
10762         {
10763             int xpos = tail->x - width;
10764             int y,waste;
10765             NK_ASSERT(xpos >= 0);
10766             /* find the left position that matches this */
10767             while (node->next->x <= xpos) {
10768                 prev = &node->next;
10769                 node = node->next;
10770             }
10771             NK_ASSERT(node->next->x > xpos && node->x <= xpos);
10772             y = nk_rp__skyline_find_min_y(c, node, xpos, width, &waste);
10773             if (y + height < c->height) {
10774                 if (y <= best_y) {
10775                     if (y < best_y || waste < best_waste || (waste==best_waste && xpos < best_x)) {
10776                         best_x = xpos;
10777                         NK_ASSERT(y <= best_y);
10778                         best_y = y;
10779                         best_waste = waste;
10780                         best = prev;
10781                     }
10782                 }
10783             }
10784             tail = tail->next;
10785         }
10786     }
10787     fr.prev_link = best;
10788     fr.x = best_x;
10789     fr.y = best_y;
10790     return fr;
10791 }
10792 NK_INTERN struct nk_rp__findresult
10793 nk_rp__skyline_pack_rectangle(struct nk_rp_context *context, int width, int height)
10794 {
10795     /* find best position according to heuristic */
10796     struct nk_rp__findresult res = nk_rp__skyline_find_best_pos(context, width, height);
10797     struct nk_rp_node *node, *cur;
10798
10799     /* bail if: */
10800     /*    1. it failed */
10801     /*    2. the best node doesn't fit (we don't always check this) */
10802     /*    3. we're out of memory */
10803     if (res.prev_link == 0 || res.y + height > context->height || context->free_head == 0) {
10804         res.prev_link = 0;
10805         return res;
10806     }
10807
10808     /* on success, create new node */
10809     node = context->free_head;
10810     node->x = (nk_rp_coord) res.x;
10811     node->y = (nk_rp_coord) (res.y + height);
10812
10813     context->free_head = node->next;
10814
10815     /* insert the new node into the right starting point, and */
10816     /* let 'cur' point to the remaining nodes needing to be */
10817     /* stitched back in */
10818     cur = *res.prev_link;
10819     if (cur->x < res.x) {
10820         /* preserve the existing one, so start testing with the next one */
10821         struct nk_rp_node *next = cur->next;
10822         cur->next = node;
10823         cur = next;
10824     } else {
10825         *res.prev_link = node;
10826     }
10827
10828     /* from here, traverse cur and free the nodes, until we get to one */
10829     /* that shouldn't be freed */
10830     while (cur->next && cur->next->x <= res.x + width) {
10831         struct nk_rp_node *next = cur->next;
10832         /* move the current node to the free list */
10833         cur->next = context->free_head;
10834         context->free_head = cur;
10835         cur = next;
10836     }
10837     /* stitch the list back in */
10838     node->next = cur;
10839
10840     if (cur->x < res.x + width)
10841         cur->x = (nk_rp_coord) (res.x + width);
10842     return res;
10843 }
10844 NK_INTERN int
10845 nk_rect_height_compare(const void *a, const void *b)
10846 {
10847     const struct nk_rp_rect *p = (const struct nk_rp_rect *) a;
10848     const struct nk_rp_rect *q = (const struct nk_rp_rect *) b;
```

```
10849     if (p->h > q->h)
10850         return -1;
10851     if (p->h < q->h)
10852         return  1;
10853     return (p->w > q->w) ? -1 : (p->w < q->w);
10854 }
10855 NK_INTERN int
10856 nk_rect_original_order(const void *a, const void *b)
10857 {
10858     const struct nk_rp_rect *p = (const struct nk_rp_rect *) a;
10859     const struct nk_rp_rect *q = (const struct nk_rp_rect *) b;
10860     return (p->was_packed < q->was_packed) ? -1 : (p->was_packed > q->was_packed);
10861 }
10862 NK_INTERN void
10863 nk_rp_qsort(struct nk_rp_rect *array, unsigned int len, int(*cmp)(const void*,const void*))
10864 {
10865     /* iterative quick sort */
10866     #define NK_MAX_SORT_STACK 64
10867     unsigned right, left = 0, stack[NK_MAX_SORT_STACK], pos = 0;
10868     unsigned seed = len/2 * 69069+1;
10869     for (;;) {
10870         for (; left+1 < len; len++) {
10871             struct nk_rp_rect pivot, tmp;
10872             if (pos == NK_MAX_SORT_STACK) len = stack[pos = 0];
10873             pivot = array[left+seed%(len-left)];
10874             seed = seed * 69069 + 1;
10875             stack[pos++] = len;
10876             for (right = left-1;;) {
10877                 while (cmp(&array[++right], &pivot) < 0);
10878                 while (cmp(&pivot, &array[--len]) < 0);
10879                 if (right >= len) break;
10880                 tmp = array[right];
10881                 array[right] = array[len];
10882                 array[len] = tmp;
10883             }
10884         }
10885         if (pos == 0) break;
10886         left = len;
10887         len = stack[--pos];
10888     }
10889     #undef NK_MAX_SORT_STACK
10890 }
10891 NK_INTERN void
10892 nk_rp_pack_rects(struct nk_rp_context *context, struct nk_rp_rect *rects, int num_rects)
10893 {
10894     int i;
10895     /* we use the 'was_packed' field internally to allow sorting/unsorting */
10896     for (i=0; i < num_rects; ++i) {
10897         rects[i].was_packed = i;
10898     }
10899
10900     /* sort according to heuristic */
10901     nk_rp_qsort(rects, (unsigned)num_rects, nk_rect_height_compare);
10902
10903     for (i=0; i < num_rects; ++i) {
10904         struct nk_rp__findresult fr = nk_rp__skyline_pack_rectangle(context, rects[i].w, rects[i].h);
10905         if (fr.prev_link) {
10906             rects[i].x = (nk_rp_coord) fr.x;
10907             rects[i].y = (nk_rp_coord) fr.y;
10908         } else {
10909             rects[i].x = rects[i].y = NK_RP__MAXVAL;
10910         }
10911     }
10912
10913     /* unsort */
10914     nk_rp_qsort(rects, (unsigned)num_rects, nk_rect_original_order);
10915
10916     /* set was_packed flags */
10917     for (i=0; i < num_rects; ++i)
10918         rects[i].was_packed = !(rects[i].x == NK_RP__MAXVAL && rects[i].y == NK_RP__MAXVAL);
10919 }
10920
10921 /*
10922  * ==============================================================
10923  *
10924  *                          TRUETYPE
10925  *
10926  * ==============================================================
10927  */
10928 /* stb_truetype.h - v1.07 - public domain */
10929 #define NK_TT_MAX_OVERSAMPLE   8
10930 #define NK_TT__OVER_MASK  (NK_TT_MAX_OVERSAMPLE-1)
10931
10932 struct nk_tt_bakedchar {
10933     unsigned short x0,y0,x1,y1;
10934     /* coordinates of bbox in bitmap */
10935     float xoff,yoff,xadvance;
```

```
10936 };
10937
10938 struct nk_tt_aligned_quad{
10939     float x0,y0,s0,t0; /* top-left */
10940     float x1,y1,s1,t1; /* bottom-right */
10941 };
10942
10943 struct nk_tt_packedchar {
10944     unsigned short x0,y0,x1,y1;
10945     /* coordinates of bbox in bitmap */
10946     float xoff,yoff,xadvance;
10947     float xoff2,yoff2;
10948 };
10949
10950 struct nk_tt_pack_range {
10951     float font_size;
10952     int first_unicode_codepoint_in_range;
10953     /* if non-zero, then the chars are continuous, and this is the first codepoint */
10954     int *array_of_unicode_codepoints;
10955     /* if non-zero, then this is an array of unicode codepoints */
10956     int num_chars;
10957     struct nk_tt_packedchar *chardata_for_range; /* output */
10958     unsigned char h_oversample, v_oversample;
10959     /* don't set these, they're used internally */
10960 };
10961
10962 struct nk_tt_pack_context {
10963     void *pack_info;
10964     int   width;
10965     int   height;
10966     int   stride_in_bytes;
10967     int   padding;
10968     unsigned int   h_oversample, v_oversample;
10969     unsigned char *pixels;
10970     void  *nodes;
10971 };
10972
10973 struct nk_tt_fontinfo {
10974     const unsigned char* data; /* pointer to .ttf file */
10975     int fontstart;/* offset of start of font */
10976     int numGlyphs;/* number of glyphs, needed for range checking */
10977     int loca,head,glyf,hhea,hmtx,kern; /* table locations as offset from start of .ttf */
10978     int index_map; /* a cmap mapping for our chosen character encoding */
10979     int indexToLocFormat; /* format needed to map from glyph index to glyph */
10980 };
10981
10982 enum {
10983   NK_TT_vmove=1,
10984   NK_TT_vline,
10985   NK_TT_vcurve
10986 };
10987
10988 struct nk_tt_vertex {
10989     short x,y,cx,cy;
10990     unsigned char type,padding;
10991 };
10992
10993 struct nk_tt__bitmap{
10994    int w,h,stride;
10995    unsigned char *pixels;
10996 };
10997
10998 struct nk_tt__hheap_chunk {
10999     struct nk_tt__hheap_chunk *next;
11000 };
11001 struct nk_tt__hheap {
11002     struct nk_allocator alloc;
11003     struct nk_tt__hheap_chunk *head;
11004     void   *first_free;
11005     int    num_remaining_in_head_chunk;
11006 };
11007
11008 struct nk_tt__edge {
11009     float x0,y0, x1,y1;
11010     int invert;
11011 };
11012
11013 struct nk_tt__active_edge {
11014     struct nk_tt__active_edge *next;
11015     float fx,fdx,fdy;
11016     float direction;
11017     float sy;
11018     float ey;
11019 };
11020 struct nk_tt__point {float x,y;};
11021
11022 #define NK_TT_MACSTYLE_DONTCARE     0
```

```
11023 #define NK_TT_MACSTYLE_BOLD         1
11024 #define NK_TT_MACSTYLE_ITALIC       2
11025 #define NK_TT_MACSTYLE_UNDERSCORE   4
11026 #define NK_TT_MACSTYLE_NONE         8
11027 /* <= not same as 0, this makes us check the bitfield is 0 */
11028
11029 enum { /* platformID */
11030    NK_TT_PLATFORM_ID_UNICODE   =0,
11031    NK_TT_PLATFORM_ID_MAC       =1,
11032    NK_TT_PLATFORM_ID_ISO       =2,
11033    NK_TT_PLATFORM_ID_MICROSOFT =3
11034 };
11035
11036 enum { /* encodingID for NK_TT_PLATFORM_ID_UNICODE */
11037    NK_TT_UNICODE_EID_UNICODE_1_0    =0,
11038    NK_TT_UNICODE_EID_UNICODE_1_1    =1,
11039    NK_TT_UNICODE_EID_ISO_10646      =2,
11040    NK_TT_UNICODE_EID_UNICODE_2_0_BMP=3,
11041    NK_TT_UNICODE_EID_UNICODE_2_0_FULL=4
11042 };
11043
11044 enum { /* encodingID for NK_TT_PLATFORM_ID_MICROSOFT */
11045    NK_TT_MS_EID_SYMBOL        =0,
11046    NK_TT_MS_EID_UNICODE_BMP   =1,
11047    NK_TT_MS_EID_SHIFTJIS      =2,
11048    NK_TT_MS_EID_UNICODE_FULL  =10
11049 };
11050
11051 enum { /* encodingID for NK_TT_PLATFORM_ID_MAC; same as Script Manager codes */
11052    NK_TT_MAC_EID_ROMAN        =0,   NK_TT_MAC_EID_ARABIC       =4,
11053    NK_TT_MAC_EID_JAPANESE     =1,   NK_TT_MAC_EID_HEBREW       =5,
11054    NK_TT_MAC_EID_CHINESE_TRAD =2,   NK_TT_MAC_EID_GREEK        =6,
11055    NK_TT_MAC_EID_KOREAN       =3,   NK_TT_MAC_EID_RUSSIAN      =7
11056 };
11057
11058 enum { /* languageID for NK_TT_PLATFORM_ID_MICROSOFT; same as LCID... */
11059        /* problematic because there are e.g. 16 english LCIDs and 16 arabic LCIDs */
11060    NK_TT_MS_LANG_ENGLISH     =0x0409,   NK_TT_MS_LANG_ITALIAN     =0x0410,
11061    NK_TT_MS_LANG_CHINESE     =0x0804,   NK_TT_MS_LANG_JAPANESE    =0x0411,
11062    NK_TT_MS_LANG_DUTCH       =0x0413,   NK_TT_MS_LANG_KOREAN      =0x0412,
11063    NK_TT_MS_LANG_FRENCH      =0x040c,   NK_TT_MS_LANG_RUSSIAN     =0x0419,
11064    NK_TT_MS_LANG_GERMAN      =0x0407,   NK_TT_MS_LANG_SPANISH     =0x0409,
11065    NK_TT_MS_LANG_HEBREW      =0x040d,   NK_TT_MS_LANG_SWEDISH     =0x041D
11066 };
11067
11068 enum { /* languageID for NK_TT_PLATFORM_ID_MAC */
11069    NK_TT_MAC_LANG_ENGLISH      =0 ,   NK_TT_MAC_LANG_JAPANESE     =11,
11070    NK_TT_MAC_LANG_ARABIC       =12,   NK_TT_MAC_LANG_KOREAN       =23,
11071    NK_TT_MAC_LANG_DUTCH        =4 ,   NK_TT_MAC_LANG_RUSSIAN      =32,
11072    NK_TT_MAC_LANG_FRENCH       =1 ,   NK_TT_MAC_LANG_SPANISH      =6 ,
11073    NK_TT_MAC_LANG_GERMAN       =2 ,   NK_TT_MAC_LANG_SWEDISH      =5 ,
11074    NK_TT_MAC_LANG_HEBREW       =10,   NK_TT_MAC_LANG_CHINESE_SIMPLIFIED =33,
11075    NK_TT_MAC_LANG_ITALIAN      =3 ,   NK_TT_MAC_LANG_CHINESE_TRAD =19
11076 };
11077
11078 #define nk_ttBYTE(p)     (* (const nk_byte *) (p))
11079 #define nk_ttCHAR(p)     (* (const char *) (p))
11080
11081 #if defined(NK_BIGENDIAN) && !defined(NK_ALLOW_UNALIGNED_TRUETYPE)
11082    #define nk_ttUSHORT(p)   (* (nk_ushort *) (p))
11083    #define nk_ttSHORT(p)    (* (nk_short *) (p))
11084    #define nk_ttULONG(p)    (* (nk_uint *) (p))
11085    #define nk_ttLONG(p)     (* (nk_int *) (p))
11086 #else
11087     static nk_ushort nk_ttUSHORT(const nk_byte *p) { return (nk_ushort)(p[0]*256 + p[1]); }
11088     static nk_short nk_ttSHORT(const nk_byte *p)   { return (nk_short)(p[0]*256 + p[1]); }
11089     static nk_uint nk_ttULONG(const nk_byte *p)  { return (nk_uint)((p[0]«24) + (p[1]«16) + (p[2]«8) +
       p[3]); }
11090 #endif
11091
11092 #define nk_tt_tag4(p,c0,c1,c2,c3)\
11093     ((p)[0] == (c0) && (p)[1] == (c1) && (p)[2] == (c2) && (p)[3] == (c3))
11094 #define nk_tt_tag(p,str) nk_tt_tag4(p,str[0],str[1],str[2],str[3])
11095
11096 NK_INTERN int nk_tt_GetGlyphShape(const struct nk_tt_fontinfo *info, struct nk_allocator *alloc,
11097                                   int glyph_index, struct nk_tt_vertex **pvertices);
11098
11099 NK_INTERN nk_uint
11100 nk_tt__find_table(const nk_byte *data, nk_uint fontstart, const char *tag)
11101 {
11102     /* @OPTIMIZE: binary search */
11103     nk_int num_tables = nk_ttUSHORT(data+fontstart+4);
11104     nk_uint tabledir = fontstart + 12;
11105     nk_int i;
11106     for (i = 0; i < num_tables; ++i) {
11107         nk_uint loc = tabledir + (nk_uint)(16*i);
11108         if (nk_tt_tag(data+loc+0, tag))
```

```
11109             return nk_ttULONG(data+loc+8);
11110     }
11111     return 0;
11112 }
11113 NK_INTERN int
11114 nk_tt_InitFont(struct nk_tt_fontinfo *info, const unsigned char *data2, int fontstart)
11115 {
11116     nk_uint cmap, t;
11117     nk_int i,numTables;
11118     const nk_byte *data = (const nk_byte *) data2;
11119
11120     info->data = data;
11121     info->fontstart = fontstart;
11122
11123     cmap = nk_tt__find_table(data, (nk_uint)fontstart, "cmap");       /* required */
11124     info->loca = (int)nk_tt__find_table(data, (nk_uint)fontstart, "loca"); /* required */
11125     info->head = (int)nk_tt__find_table(data, (nk_uint)fontstart, "head"); /* required */
11126     info->glyf = (int)nk_tt__find_table(data, (nk_uint)fontstart, "glyf"); /* required */
11127     info->hhea = (int)nk_tt__find_table(data, (nk_uint)fontstart, "hhea"); /* required */
11128     info->hmtx = (int)nk_tt__find_table(data, (nk_uint)fontstart, "hmtx"); /* required */
11129     info->kern = (int)nk_tt__find_table(data, (nk_uint)fontstart, "kern"); /* not required */
11130     if (!cmap || !info->loca || !info->head || !info->glyf || !info->hhea || !info->hmtx)
11131         return 0;
11132
11133     t = nk_tt__find_table(data, (nk_uint)fontstart, "maxp");
11134     if (t) info->numGlyphs = nk_ttUSHORT(data+t+4);
11135     else info->numGlyphs = 0xffff;
11136
11137     /* find a cmap encoding table we understand *now* to avoid searching */
11138     /* later. (todo: could make this installable) */
11139     /* the same regardless of glyph. */
11140     numTables = nk_ttUSHORT(data + cmap + 2);
11141     info->index_map = 0;
11142     for (i=0; i < numTables; ++i)
11143     {
11144         nk_uint encoding_record = cmap + 4 + 8 * (nk_uint)i;
11145         /* find an encoding we understand: */
11146         switch(nk_ttUSHORT(data+encoding_record)) {
11147         case NK_TT_PLATFORM_ID_MICROSOFT:
11148             switch (nk_ttUSHORT(data+encoding_record+2)) {
11149             case NK_TT_MS_EID_UNICODE_BMP:
11150             case NK_TT_MS_EID_UNICODE_FULL:
11151                 /* MS/Unicode */
11152                 info->index_map = (int)(cmap + nk_ttULONG(data+encoding_record+4));
11153                 break;
11154             default: break;
11155             } break;
11156         case NK_TT_PLATFORM_ID_UNICODE:
11157             /* Mac/iOS has these */
11158             /* all the encodingIDs are unicode, so we don't bother to check it */
11159             info->index_map = (int)(cmap + nk_ttULONG(data+encoding_record+4));
11160             break;
11161         default: break;
11162         }
11163     }
11164     if (info->index_map == 0)
11165         return 0;
11166     info->indexToLocFormat = nk_ttUSHORT(data+info->head + 50);
11167     return 1;
11168 }
11169 NK_INTERN int
11170 nk_tt_FindGlyphIndex(const struct nk_tt_fontinfo *info, int unicode_codepoint)
11171 {
11172     const nk_byte *data = info->data;
11173     nk_uint index_map = (nk_uint)info->index_map;
11174
11175     nk_ushort format = nk_ttUSHORT(data + index_map + 0);
11176     if (format == 0) { /* apple byte encoding */
11177         nk_int bytes = nk_ttUSHORT(data + index_map + 2);
11178         if (unicode_codepoint < bytes-6)
11179             return nk_ttBYTE(data + index_map + 6 + unicode_codepoint);
11180         return 0;
11181     } else if (format == 6) {
11182         nk_uint first = nk_ttUSHORT(data + index_map + 6);
11183         nk_uint count = nk_ttUSHORT(data + index_map + 8);
11184         if ((nk_uint) unicode_codepoint >= first && (nk_uint) unicode_codepoint < first+count)
11185             return nk_ttUSHORT(data + index_map + 10 + (unicode_codepoint - (int)first)*2);
11186         return 0;
11187     } else if (format == 2) {
11188         NK_ASSERT(0); /* @TODO: high-byte mapping for japanese/chinese/korean */
11189         return 0;
11190     } else if (format == 4) { /* standard mapping for windows fonts: binary search collection of
      ranges */
11191         nk_ushort segcount = nk_ttUSHORT(data+index_map+6) » 1;
11192         nk_ushort searchRange = nk_ttUSHORT(data+index_map+8) » 1;
11193         nk_ushort entrySelector = nk_ttUSHORT(data+index_map+10);
11194         nk_ushort rangeShift = nk_ttUSHORT(data+index_map+12) » 1;
```

```
11195
11196         /* do a binary search of the segments */
11197         nk_uint endCount = index_map + 14;
11198         nk_uint search = endCount;
11199
11200         if (unicode_codepoint > 0xffff)
11201             return 0;
11202
11203         /* they lie from endCount .. endCount + segCount */
11204         /* but searchRange is the nearest power of two, so... */
11205         if (unicode_codepoint >= nk_ttUSHORT(data + search + rangeShift*2))
11206             search += (nk_uint)(rangeShift*2);
11207
11208         /* now decrement to bias correctly to find smallest */
11209         search -= 2;
11210         while (entrySelector) {
11211             nk_ushort end;
11212             searchRange »= 1;
11213             end = nk_ttUSHORT(data + search + searchRange*2);
11214             if (unicode_codepoint > end)
11215                 search += (nk_uint)(searchRange*2);
11216             --entrySelector;
11217         }
11218         search += 2;
11219
11220       {
11221          nk_ushort offset, start;
11222          nk_ushort item = (nk_ushort) ((search - endCount) » 1);
11223
11224          NK_ASSERT(unicode_codepoint <= nk_ttUSHORT(data + endCount + 2*item));
11225          start = nk_ttUSHORT(data + index_map + 14 + segcount*2 + 2 + 2*item);
11226          if (unicode_codepoint < start)
11227             return 0;
11228
11229          offset = nk_ttUSHORT(data + index_map + 14 + segcount*6 + 2 + 2*item);
11230          if (offset == 0)
11231             return (nk_ushort) (unicode_codepoint + nk_ttSHORT(data + index_map + 14 + segcount*4 + 2
     + 2*item));
11232
11233          return nk_ttUSHORT(data + offset + (unicode_codepoint-start)*2 + index_map + 14 + segcount*6
     + 2 + 2*item);
11234       }
11235     } else if (format == 12 || format == 13) {
11236         nk_uint ngroups = nk_ttULONG(data+index_map+12);
11237         nk_int low,high;
11238         low = 0; high = (nk_int)ngroups;
11239         /* Binary search the right group. */
11240         while (low < high) {
11241             nk_int mid = low + ((high-low) » 1); /* rounds down, so low <= mid < high */
11242             nk_uint start_char = nk_ttULONG(data+index_map+16+mid*12);
11243             nk_uint end_char = nk_ttULONG(data+index_map+16+mid*12+4);
11244             if ((nk_uint) unicode_codepoint < start_char)
11245                 high = mid;
11246             else if ((nk_uint) unicode_codepoint > end_char)
11247                 low = mid+1;
11248             else {
11249                 nk_uint start_glyph = nk_ttULONG(data+index_map+16+mid*12+8);
11250                 if (format == 12)
11251                     return (int)start_glyph + (int)unicode_codepoint - (int)start_char;
11252                 else /* format == 13 */
11253                     return (int)start_glyph;
11254             }
11255         }
11256         return 0; /* not found */
11257     }
11258     /* @TODO */
11259     NK_ASSERT(0);
11260     return 0;
11261 }
11262 NK_INTERN void
11263 nk_tt_setvertex(struct nk_tt_vertex *v, nk_byte type, nk_int x, nk_int y, nk_int cx, nk_int cy)
11264 {
11265     v->type = type;
11266     v->x = (nk_short) x;
11267     v->y = (nk_short) y;
11268     v->cx = (nk_short) cx;
11269     v->cy = (nk_short) cy;
11270 }
11271 NK_INTERN int
11272 nk_tt__GetGlyfOffset(const struct nk_tt_fontinfo *info, int glyph_index)
11273 {
11274     int g1,g2;
11275     if (glyph_index >= info->numGlyphs) return -1; /* glyph index out of range */
11276     if (info->indexToLocFormat >= 2)    return -1; /* unknown index->glyph map format */
11277
11278     if (info->indexToLocFormat == 0) {
11279         g1 = info->glyf + nk_ttUSHORT(info->data + info->loca + glyph_index * 2) * 2;
```

```
11280         g2 = info->glyf + nk_ttUSHORT(info->data + info->loca + glyph_index * 2 + 2) * 2;
11281     } else {
11282         g1 = info->glyf + (int)nk_ttULONG (info->data + info->loca + glyph_index * 4);
11283         g2 = info->glyf + (int)nk_ttULONG (info->data + info->loca + glyph_index * 4 + 4);
11284     }
11285     return g1==g2 ? -1 : g1; /* if length is 0, return -1 */
11286 }
11287 NK_INTERN int
11288 nk_tt_GetGlyphBox(const struct nk_tt_fontinfo *info, int glyph_index,
11289     int *x0, int *y0, int *x1, int *y1)
11290 {
11291     int g = nk_tt__GetGlyfOffset(info, glyph_index);
11292     if (g < 0) return 0;
11293
11294     if (x0) *x0 = nk_ttSHORT(info->data + g + 2);
11295     if (y0) *y0 = nk_ttSHORT(info->data + g + 4);
11296     if (x1) *x1 = nk_ttSHORT(info->data + g + 6);
11297     if (y1) *y1 = nk_ttSHORT(info->data + g + 8);
11298     return 1;
11299 }
11300 NK_INTERN int
11301 nk_tt__close_shape(struct nk_tt_vertex *vertices, int num_vertices, int was_off,
11302     int start_off, nk_int sx, nk_int sy, nk_int scx, nk_int scy, nk_int cx, nk_int cy)
11303 {
11304    if (start_off) {
11305       if (was_off)
11306          nk_tt_setvertex(&vertices[num_vertices++], NK_TT_vcurve, (cx+scx)»1, (cy+scy)»1, cx,cy);
11307       nk_tt_setvertex(&vertices[num_vertices++], NK_TT_vcurve, sx,sy,scx,scy);
11308    } else {
11309       if (was_off)
11310          nk_tt_setvertex(&vertices[num_vertices++], NK_TT_vcurve,sx,sy,cx,cy);
11311       else
11312          nk_tt_setvertex(&vertices[num_vertices++], NK_TT_vline,sx,sy,0,0);
11313    }
11314    return num_vertices;
11315 }
11316 NK_INTERN int
11317 nk_tt_GetGlyphShape(const struct nk_tt_fontinfo *info, struct nk_allocator *alloc,
11318     int glyph_index, struct nk_tt_vertex **pvertices)
11319 {
11320     nk_short numberOfContours;
11321     const nk_byte *endPtsOfContours;
11322     const nk_byte *data = info->data;
11323     struct nk_tt_vertex *vertices=0;
11324     int num_vertices=0;
11325     int g = nk_tt__GetGlyfOffset(info, glyph_index);
11326     *pvertices = 0;
11327
11328     if (g < 0) return 0;
11329     numberOfContours = nk_ttSHORT(data + g);
11330     if (numberOfContours > 0) {
11331         nk_byte flags=0,flagcount;
11332         nk_int ins, i,j=0,m,n, next_move, was_off=0, off, start_off=0;
11333         nk_int x,y,cx,cy,sx,sy, scx,scy;
11334         const nk_byte *points;
11335         endPtsOfContours = (data + g + 10);
11336         ins = nk_ttUSHORT(data + g + 10 + numberOfContours * 2);
11337         points = data + g + 10 + numberOfContours * 2 + 2 + ins;
11338
11339         n = 1+nk_ttUSHORT(endPtsOfContours + numberOfContours*2-2);
11340         m = n + 2*numberOfContours;  /* a loose bound on how many vertices we might need */
11341         vertices = (struct nk_tt_vertex *)alloc->alloc(alloc->userdata, 0, (nk_size)m *
     sizeof(vertices[0]));
11342         if (vertices == 0)
11343             return 0;
11344
11345         next_move = 0;
11346         flagcount=0;
11347
11348         /* in first pass, we load uninterpreted data into the allocated array */
11349         /* above, shifted to the end of the array so we won't overwrite it when */
11350         /* we create our final data starting from the front */
11351         off = m - n; /* starting offset for uninterpreted data, regardless of how m ends up being
     calculated */
11352
11353         /* first load flags */
11354         for (i=0; i < n; ++i) {
11355             if (flagcount == 0) {
11356                 flags = *points++;
11357                 if (flags & 8)
11358                     flagcount = *points++;
11359             } else --flagcount;
11360             vertices[off+i].type = flags;
11361         }
11362
11363         /* now load x coordinates */
11364         x=0;
```

```
11365         for (i=0; i < n; ++i) {
11366             flags = vertices[off+i].type;
11367             if (flags & 2) {
11368                 nk_short dx = *points++;
11369                 x += (flags & 16) ? dx : -dx; /* ??? */
11370             } else {
11371                 if (!(flags & 16)) {
11372                     x = x + (nk_short) (points[0]*256 + points[1]);
11373                     points += 2;
11374                 }
11375             }
11376             vertices[off+i].x = (nk_short) x;
11377         }
11378 
11379         /* now load y coordinates */
11380         y=0;
11381         for (i=0; i < n; ++i) {
11382             flags = vertices[off+i].type;
11383             if (flags & 4) {
11384                 nk_short dy = *points++;
11385                 y += (flags & 32) ? dy : -dy; /* ??? */
11386             } else {
11387                 if (!(flags & 32)) {
11388                     y = y + (nk_short) (points[0]*256 + points[1]);
11389                     points += 2;
11390                 }
11391             }
11392             vertices[off+i].y = (nk_short) y;
11393         }
11394 
11395         /* now convert them to our format */
11396         num_vertices=0;
11397         sx = sy = cx = cy = scx = scy = 0;
11398         for (i=0; i < n; ++i)
11399         {
11400             flags = vertices[off+i].type;
11401             x     = (nk_short) vertices[off+i].x;
11402             y     = (nk_short) vertices[off+i].y;
11403 
11404             if (next_move == i) {
11405                 if (i != 0)
11406                     num_vertices = nk_tt__close_shape(vertices, num_vertices, was_off, start_off,
      sx,sy,scx,scy,cx,cy);
11407 
11408                 /* now start the new one                */
11409                 start_off = !(flags & 1);
11410                 if (start_off) {
11411                     /* if we start off with an off-curve point, then when we need to find a point on
      the curve */
11412                     /* where we can start, and we need to save some state for when we wraparound. */
11413                     scx = x;
11414                     scy = y;
11415                     if (!(vertices[off+i+1].type & 1)) {
11416                         /* next point is also a curve point, so interpolate an on-point curve */
11417                         sx = (x + (nk_int) vertices[off+i+1].x) » 1;
11418                         sy = (y + (nk_int) vertices[off+i+1].y) » 1;
11419                     } else {
11420                         /* otherwise just use the next point as our start point */
11421                         sx = (nk_int) vertices[off+i+1].x;
11422                         sy = (nk_int) vertices[off+i+1].y;
11423                         ++i; /* we're using point i+1 as the starting point, so skip it */
11424                     }
11425                 } else {
11426                     sx = x;
11427                     sy = y;
11428                 }
11429                 nk_tt_setvertex(&vertices[num_vertices++], NK_TT_vmove,sx,sy,0,0);
11430                 was_off = 0;
11431                 next_move = 1 + nk_ttUSHORT(endPtsOfContours+j*2);
11432                 ++j;
11433             } else {
11434                 if (!(flags & 1))
11435                 { /* if it's a curve */
11436                     if (was_off) /* two off-curve control points in a row means interpolate an
      on-curve midpoint */
11437                         nk_tt_setvertex(&vertices[num_vertices++], NK_TT_vcurve, (cx+x)»1, (cy+y)»1,
      cx, cy);
11438                     cx = x;
11439                     cy = y;
11440                     was_off = 1;
11441                 } else {
11442                     if (was_off)
11443                         nk_tt_setvertex(&vertices[num_vertices++], NK_TT_vcurve, x,y, cx, cy);
11444                     else nk_tt_setvertex(&vertices[num_vertices++], NK_TT_vline, x,y,0,0);
11445                     was_off = 0;
11446                 }
11447             }
```

```
11448         }
11449         num_vertices = nk_tt__close_shape(vertices, num_vertices, was_off, start_off,
      sx,sy,scx,scy,cx,cy);
11450     } else if (numberOfContours == -1) {
11451         /* Compound shapes. */
11452         int more = 1;
11453         const nk_byte *comp = data + g + 10;
11454         num_vertices = 0;
11455         vertices = 0;
11456
11457         while (more)
11458         {
11459             nk_ushort flags, gidx;
11460             int comp_num_verts = 0, i;
11461             struct nk_tt_vertex *comp_verts = 0, *tmp = 0;
11462             float mtx[6] = {1,0,0,1,0,0}, m, n;
11463
11464             flags = (nk_ushort)nk_ttSHORT(comp); comp+=2;
11465             gidx = (nk_ushort)nk_ttSHORT(comp); comp+=2;
11466
11467             if (flags & 2) { /* XY values */
11468                 if (flags & 1) { /* shorts */
11469                     mtx[4] = nk_ttSHORT(comp); comp+=2;
11470                     mtx[5] = nk_ttSHORT(comp); comp+=2;
11471                 } else {
11472                     mtx[4] = nk_ttCHAR(comp); comp+=1;
11473                     mtx[5] = nk_ttCHAR(comp); comp+=1;
11474                 }
11475             } else {
11476                 /* @TODO handle matching point */
11477                 NK_ASSERT(0);
11478             }
11479             if (flags & (1«3)) { /* WE_HAVE_A_SCALE */
11480                 mtx[0] = mtx[3] = nk_ttSHORT(comp)/16384.0f; comp+=2;
11481                 mtx[1] = mtx[2] = 0;
11482             } else if (flags & (1«6)) { /* WE_HAVE_AN_X_AND_YSCALE */
11483                 mtx[0] = nk_ttSHORT(comp)/16384.0f; comp+=2;
11484                 mtx[1] = mtx[2] = 0;
11485                 mtx[3] = nk_ttSHORT(comp)/16384.0f; comp+=2;
11486             } else if (flags & (1«7)) { /* WE_HAVE_A_TWO_BY_TWO */
11487                 mtx[0] = nk_ttSHORT(comp)/16384.0f; comp+=2;
11488                 mtx[1] = nk_ttSHORT(comp)/16384.0f; comp+=2;
11489                 mtx[2] = nk_ttSHORT(comp)/16384.0f; comp+=2;
11490                 mtx[3] = nk_ttSHORT(comp)/16384.0f; comp+=2;
11491             }
11492
11493              /* Find transformation scales. */
11494             m = (float) NK_SQRT(mtx[0]*mtx[0] + mtx[1]*mtx[1]);
11495             n = (float) NK_SQRT(mtx[2]*mtx[2] + mtx[3]*mtx[3]);
11496
11497              /* Get indexed glyph. */
11498             comp_num_verts = nk_tt_GetGlyphShape(info, alloc, gidx, &comp_verts);
11499             if (comp_num_verts > 0)
11500             {
11501                 /* Transform vertices. */
11502                 for (i = 0; i < comp_num_verts; ++i) {
11503                     struct nk_tt_vertex* v = &comp_verts[i];
11504                     short x,y;
11505                     x=v->x; y=v->y;
11506                     v->x = (short)(m * (mtx[0]*x + mtx[2]*y + mtx[4]));
11507                     v->y = (short)(n * (mtx[1]*x + mtx[3]*y + mtx[5]));
11508                     x=v->cx; y=v->cy;
11509                     v->cx = (short)(m * (mtx[0]*x + mtx[2]*y + mtx[4]));
11510                     v->cy = (short)(n * (mtx[1]*x + mtx[3]*y + mtx[5]));
11511                 }
11512                 /* Append vertices. */
11513                 tmp = (struct nk_tt_vertex*)alloc->alloc(alloc->userdata, 0,
11514                     (nk_size)(num_vertices+comp_num_verts)*sizeof(struct nk_tt_vertex));
11515                 if (!tmp) {
11516                     if (vertices) alloc->free(alloc->userdata, vertices);
11517                     if (comp_verts) alloc->free(alloc->userdata, comp_verts);
11518                     return 0;
11519                 }
11520                 if (num_vertices > 0) NK_MEMCPY(tmp, vertices, (nk_size)num_vertices*sizeof(struct
      nk_tt_vertex));
11521                 NK_MEMCPY(tmp+num_vertices, comp_verts, (nk_size)comp_num_verts*sizeof(struct
      nk_tt_vertex));
11522                 if (vertices) alloc->free(alloc->userdata,vertices);
11523                 vertices = tmp;
11524                 alloc->free(alloc->userdata,comp_verts);
11525                 num_vertices += comp_num_verts;
11526             }
11527             /* More components ? */
11528             more = flags & (1«5);
11529         }
11530     } else if (numberOfContours < 0) {
11531         /* @TODO other compound variations? */
```

```
11532         NK_ASSERT(0);
11533     } else {
11534         /* numberOfCounters == 0, do nothing */
11535     }
11536     *pvertices = vertices;
11537     return num_vertices;
11538 }
11539 NK_INTERN void
11540 nk_tt_GetGlyphHMetrics(const struct nk_tt_fontinfo *info, int glyph_index,
11541     int *advanceWidth, int *leftSideBearing)
11542 {
11543     nk_ushort numOfLongHorMetrics = nk_ttUSHORT(info->data+info->hhea + 34);
11544     if (glyph_index < numOfLongHorMetrics) {
11545         if (advanceWidth)
11546             *advanceWidth    = nk_ttSHORT(info->data + info->hmtx + 4*glyph_index);
11547         if (leftSideBearing)
11548             *leftSideBearing = nk_ttSHORT(info->data + info->hmtx + 4*glyph_index + 2);
11549     } else {
11550         if (advanceWidth)
11551             *advanceWidth    = nk_ttSHORT(info->data + info->hmtx + 4*(numOfLongHorMetrics-1));
11552         if (leftSideBearing)
11553             *leftSideBearing = nk_ttSHORT(info->data + info->hmtx + 4*numOfLongHorMetrics +
       2*(glyph_index - numOfLongHorMetrics));
11554     }
11555 }
11556 NK_INTERN void
11557 nk_tt_GetFontVMetrics(const struct nk_tt_fontinfo *info,
11558     int *ascent, int *descent, int *lineGap)
11559 {
11560    if (ascent ) *ascent  = nk_ttSHORT(info->data+info->hhea + 4);
11561    if (descent) *descent = nk_ttSHORT(info->data+info->hhea + 6);
11562    if (lineGap) *lineGap = nk_ttSHORT(info->data+info->hhea + 8);
11563 }
11564 NK_INTERN float
11565 nk_tt_ScaleForPixelHeight(const struct nk_tt_fontinfo *info, float height)
11566 {
11567    int fheight = nk_ttSHORT(info->data + info->hhea + 4) - nk_ttSHORT(info->data + info->hhea + 6);
11568    return (float) height / (float)fheight;
11569 }
11570 NK_INTERN float
11571 nk_tt_ScaleForMappingEmToPixels(const struct nk_tt_fontinfo *info, float pixels)
11572 {
11573    int unitsPerEm = nk_ttUSHORT(info->data + info->head + 18);
11574    return pixels / (float)unitsPerEm;
11575 }
11576
11577 /*-------------------------------------------------------------
11578  *            antialiasing software rasterizer
11579  * -------------------------------------------------------------*/
11580 NK_INTERN void
11581 nk_tt_GetGlyphBitmapBoxSubpixel(const struct nk_tt_fontinfo *font,
11582     int glyph, float scale_x, float scale_y,float shift_x, float shift_y,
11583     int *ix0, int *iy0, int *ix1, int *iy1)
11584 {
11585     int x0,y0,x1,y1;
11586     if (!nk_tt_GetGlyphBox(font, glyph, &x0,&y0,&x1,&y1)) {
11587         /* e.g. space character */
11588         if (ix0) *ix0 = 0;
11589         if (iy0) *iy0 = 0;
11590         if (ix1) *ix1 = 0;
11591         if (iy1) *iy1 = 0;
11592     } else {
11593         /* move to integral bboxes (treating pixels as little squares, what pixels get touched)? */
11594         if (ix0) *ix0 = nk_ifloorf((float)x0 * scale_x + shift_x);
11595         if (iy0) *iy0 = nk_ifloorf((float)-y1 * scale_y + shift_y);
11596         if (ix1) *ix1 = nk_iceilf ((float)x1 * scale_x + shift_x);
11597         if (iy1) *iy1 = nk_iceilf ((float)-y0 * scale_y + shift_y);
11598     }
11599 }
11600 NK_INTERN void
11601 nk_tt_GetGlyphBitmapBox(const struct nk_tt_fontinfo *font, int glyph,
11602     float scale_x, float scale_y, int *ix0, int *iy0, int *ix1, int *iy1)
11603 {
11604    nk_tt_GetGlyphBitmapBoxSubpixel(font, glyph, scale_x, scale_y,0.0f,0.0f, ix0, iy0, ix1, iy1);
11605 }
11606
11607 /*-------------------------------------------------------------
11608  *                          Rasterizer
11609  * -------------------------------------------------------------*/
11610 NK_INTERN void*
11611 nk_tt__hheap_alloc(struct nk_tt__hheap *hh, nk_size size)
11612 {
11613     if (hh->first_free) {
11614         void *p = hh->first_free;
11615         hh->first_free = * (void **) p;
11616         return p;
11617     } else {
```

```
11618         if (hh->num_remaining_in_head_chunk == 0) {
11619             int count = (size < 32 ? 2000 : size < 128 ? 800 : 100);
11620             struct nk_tt__hheap_chunk *c = (struct nk_tt__hheap_chunk *)
11621                 hh->alloc.alloc(hh->alloc.userdata, 0,
11622                 sizeof(struct nk_tt__hheap_chunk) + size * (nk_size)count);
11623             if (c == 0) return 0;
11624             c->next = hh->head;
11625             hh->head = c;
11626             hh->num_remaining_in_head_chunk = count;
11627         }
11628         --hh->num_remaining_in_head_chunk;
11629         return (char *) (hh->head) + size * (nk_size)hh->num_remaining_in_head_chunk;
11630     }
11631 }
11632 NK_INTERN void
11633 nk_tt__hheap_free(struct nk_tt__hheap *hh, void *p)
11634 {
11635     *(void **) p = hh->first_free;
11636     hh->first_free = p;
11637 }
11638 NK_INTERN void
11639 nk_tt__hheap_cleanup(struct nk_tt__hheap *hh)
11640 {
11641     struct nk_tt__hheap_chunk *c = hh->head;
11642     while (c) {
11643         struct nk_tt__hheap_chunk *n = c->next;
11644         hh->alloc.free(hh->alloc.userdata, c);
11645         c = n;
11646     }
11647 }
11648 NK_INTERN struct nk_tt__active_edge*
11649 nk_tt__new_active(struct nk_tt__hheap *hh, struct nk_tt__edge *e,
11650     int off_x, float start_point)
11651 {
11652     struct nk_tt__active_edge *z = (struct nk_tt__active_edge *)
11653         nk_tt__hheap_alloc(hh, sizeof(*z));
11654     float dxdy = (e->x1 - e->x0) / (e->y1 - e->y0);
11655     /*STBTT_assert(e->y0 <= start_point); */
11656     if (!z) return z;
11657     z->fdx = dxdy;
11658     z->fdy = (dxdy != 0) ? (1/dxdy): 0;
11659     z->fx = e->x0 + dxdy * (start_point - e->y0);
11660     z->fx -= (float)off_x;
11661     z->direction = e->invert ? 1.0f : -1.0f;
11662     z->sy = e->y0;
11663     z->ey = e->y1;
11664     z->next = 0;
11665     return z;
11666 }
11667 NK_INTERN void
11668 nk_tt__handle_clipped_edge(float *scanline, int x, struct nk_tt__active_edge *e,
11669     float x0, float y0, float x1, float y1)
11670 {
11671     if (y0 == y1) return;
11672     NK_ASSERT(y0 < y1);
11673     NK_ASSERT(e->sy <= e->ey);
11674     if (y0 > e->ey) return;
11675     if (y1 < e->sy) return;
11676     if (y0 < e->sy) {
11677         x0 += (x1-x0) * (e->sy - y0) / (y1-y0);
11678         y0 = e->sy;
11679     }
11680     if (y1 > e->ey) {
11681         x1 += (x1-x0) * (e->ey - y1) / (y1-y0);
11682         y1 = e->ey;
11683     }
11684
11685     if (x0 == x) NK_ASSERT(x1 <= x+1);
11686     else if (x0 == x+1) NK_ASSERT(x1 >= x);
11687     else if (x0 <= x) NK_ASSERT(x1 <= x);
11688     else if (x0 >= x+1) NK_ASSERT(x1 >= x+1);
11689     else NK_ASSERT(x1 >= x && x1 <= x+1);
11690
11691     if (x0 <= x && x1 <= x)
11692         scanline[x] += e->direction * (y1-y0);
11693     else if (x0 >= x+1 && x1 >= x+1);
11694     else {
11695         NK_ASSERT(x0 >= x && x0 <= x+1 && x1 >= x && x1 <= x+1);
11696         /* coverage = 1 - average x position */
11697         scanline[x] += (float)e->direction * (float)(y1-y0) *
       (1.0f-((x0-(float)x)+(x1-(float)x))/2.0f);
11698     }
11699 }
11700 NK_INTERN void
11701 nk_tt__fill_active_edges_new(float *scanline, float *scanline_fill, int len,
11702     struct nk_tt__active_edge *e, float y_top)
11703 {
```

```
11704     float y_bottom = y_top+1;
11705     while (e)
11706     {
11707         /* brute force every pixel */
11708         /* compute intersection points with top & bottom */
11709         NK_ASSERT(e->ey >= y_top);
11710         if (e->fdx == 0) {
11711             float x0 = e->fx;
11712             if (x0 < len) {
11713                 if (x0 >= 0) {
11714                     nk_tt__handle_clipped_edge(scanline,(int) x0,e, x0,y_top, x0,y_bottom);
11715                     nk_tt__handle_clipped_edge(scanline_fill-1,(int) x0+1,e, x0,y_top, x0,y_bottom);
11716                 } else {
11717                     nk_tt__handle_clipped_edge(scanline_fill-1,0,e, x0,y_top, x0,y_bottom);
11718                 }
11719             }
11720         } else {
11721             float x0 = e->fx;
11722             float dx = e->fdx;
11723             float xb = x0 + dx;
11724             float x_top, x_bottom;
11725             float y0,y1;
11726             float dy = e->fdy;
11727             NK_ASSERT(e->sy <= y_bottom && e->ey >= y_top);
11728
11729             /* compute endpoints of line segment clipped to this scanline (if the */
11730             /* line segment starts on this scanline. x0 is the intersection of the */
11731             /* line with y_top, but that may be off the line segment. */
11732             if (e->sy > y_top) {
11733                 x_top = x0 + dx * (e->sy - y_top);
11734                 y0 = e->sy;
11735             } else {
11736                 x_top = x0;
11737                 y0 = y_top;
11738             }
11739
11740             if (e->ey < y_bottom) {
11741                 x_bottom = x0 + dx * (e->ey - y_top);
11742                 y1 = e->ey;
11743             } else {
11744                 x_bottom = xb;
11745                 y1 = y_bottom;
11746             }
11747
11748             if (x_top >= 0 && x_bottom >= 0 && x_top < len && x_bottom < len)
11749             {
11750                 /* from here on, we don't have to range check x values */
11751                 if ((int) x_top == (int) x_bottom) {
11752                     float height;
11753                     /* simple case, only spans one pixel */
11754                     int x = (int) x_top;
11755                     height = y1 - y0;
11756                     NK_ASSERT(x >= 0 && x < len);
11757                     scanline[x] += e->direction * (1.0f-(((float)x_top - (float)x) +
    ((float)x_bottom-(float)x))/2.0f)  * (float)height;
11758                     scanline_fill[x] += e->direction * (float)height; /* everything right of this
    pixel is filled */
11759                 } else {
11760                     int x,x1,x2;
11761                     float y_crossing, step, sign, area;
11762                     /* covers 2+ pixels */
11763                     if (x_top > x_bottom)
11764                     {
11765                         /* flip scanline vertically; signed area is the same */
11766                         float t;
11767                         y0 = y_bottom - (y0 - y_top);
11768                         y1 = y_bottom - (y1 - y_top);
11769                         t = y0; y0 = y1; y1 = t;
11770                         t = x_bottom; x_bottom = x_top; x_top = t;
11771                         dx = -dx;
11772                         dy = -dy;
11773                         t = x0; x0 = xb; xb = t;
11774                     }
11775
11776                     x1 = (int) x_top;
11777                     x2 = (int) x_bottom;
11778                     /* compute intersection with y axis at x1+1 */
11779                     y_crossing = ((float)x1+1 - (float)x0) * (float)dy + (float)y_top;
11780
11781                     sign = e->direction;
11782                     /* area of the rectangle covered from y0..y_crossing */
11783                     area = sign * (y_crossing-y0);
11784                     /* area of the triangle (x_top,y0), (x+1,y0), (x+1,y_crossing) */
11785                     scanline[x1] += area * (1.0f-((float)((float)x_top -
    (float)x1)+(float)(x1+1-x1))/2.0f);
11786
11787                     step = sign * dy;
```

```
11788                     for (x = x1+1; x < x2; ++x) {
11789                         scanline[x] += area + step/2;
11790                         area += step;
11791                     }
11792                     y_crossing += (float)dy * (float)(x2 - (x1+1));
11793
11794                     scanline[x2] += area + sign *
      (1.0f-((float)(x2-x2)+((float)x_bottom-(float)x2))/2.0f) * (y1-y_crossing);
11795                     scanline_fill[x2] += sign * (y1-y0);
11796                 }
11797             }
11798             else
11799             {
11800                 /* if edge goes outside of box we're drawing, we require */
11801                 /* clipping logic. since this does not match the intended use */
11802                 /* of this library, we use a different, very slow brute */
11803                 /* force implementation */
11804                 int x;
11805                 for (x=0; x < len; ++x)
11806                 {
11807                     /* cases: */
11808                     /* */
11809                     /* there can be up to two intersections with the pixel. any intersection */
11810                     /* with left or right edges can be handled by splitting into two (or three) */
11811                     /* regions. intersections with top & bottom do not necessitate case-wise logic. */
11812                     /* */
11813                     /* the old way of doing this found the intersections with the left & right edges,
      */
11814                     /* then used some simple logic to produce up to three segments in sorted order */
11815                     /* from top-to-bottom. however, this had a problem: if an x edge was epsilon */
11816                     /* across the x border, then the corresponding y position might not be distinct */
11817                     /* from the other y segment, and it might ignored as an empty segment. to avoid */
11818                     /* that, we need to explicitly produce segments based on x positions. */
11819
11820                     /* rename variables to clear pairs */
11821                     float ya = y_top;
11822                     float x1 = (float) (x);
11823                     float x2 = (float) (x+1);
11824                     float x3 = xb;
11825                     float y3 = y_bottom;
11826                     float yb,y2;
11827
11828                     yb = ((float)x - x0) / dx + y_top;
11829                     y2 = ((float)x+1 - x0) / dx + y_top;
11830
11831                     if (x0 < x1 && x3 > x2) {         /* three segments descending down-right */
11832                         nk_tt__handle_clipped_edge(scanline,x,e, x0,ya, x1,yb);
11833                         nk_tt__handle_clipped_edge(scanline,x,e, x1,yb, x2,y2);
11834                         nk_tt__handle_clipped_edge(scanline,x,e, x2,y2, x3,y3);
11835                     } else if (x3 < x1 && x0 > x2) {  /* three segments descending down-left */
11836                         nk_tt__handle_clipped_edge(scanline,x,e, x0,ya, x2,y2);
11837                         nk_tt__handle_clipped_edge(scanline,x,e, x2,y2, x1,yb);
11838                         nk_tt__handle_clipped_edge(scanline,x,e, x1,yb, x3,y3);
11839                     } else if (x0 < x1 && x3 > x1) {  /* two segments across x, down-right */
11840                         nk_tt__handle_clipped_edge(scanline,x,e, x0,ya, x1,yb);
11841                         nk_tt__handle_clipped_edge(scanline,x,e, x1,yb, x3,y3);
11842                     } else if (x3 < x1 && x0 > x1) {  /* two segments across x, down-left */
11843                         nk_tt__handle_clipped_edge(scanline,x,e, x0,ya, x1,yb);
11844                         nk_tt__handle_clipped_edge(scanline,x,e, x1,yb, x3,y3);
11845                     } else if (x0 < x2 && x3 > x2) {  /* two segments across x+1, down-right */
11846                         nk_tt__handle_clipped_edge(scanline,x,e, x0,ya, x2,y2);
11847                         nk_tt__handle_clipped_edge(scanline,x,e, x2,y2, x3,y3);
11848                     } else if (x3 < x2 && x0 > x2) {  /* two segments across x+1, down-left */
11849                         nk_tt__handle_clipped_edge(scanline,x,e, x0,ya, x2,y2);
11850                         nk_tt__handle_clipped_edge(scanline,x,e, x2,y2, x3,y3);
11851                     } else {  /* one segment */
11852                         nk_tt__handle_clipped_edge(scanline,x,e, x0,ya, x3,y3);
11853                     }
11854                 }
11855             }
11856         }
11857         e = e->next;
11858     }
11859 }
11860 NK_INTERN void
11861 nk_tt__rasterize_sorted_edges(struct nk_tt__bitmap *result, struct nk_tt__edge *e,
11862     int n, int vsubsample, int off_x, int off_y, struct nk_allocator *alloc)
11863 {
11864     /* directly AA rasterize edges w/o supersampling */
11865     struct nk_tt__hheap hh;
11866     struct nk_tt__active_edge *active = 0;
11867     int y,j=0, i;
11868     float scanline_data[129], *scanline, *scanline2;
11869
11870     NK_UNUSED(vsubsample);
11871     nk_zero_struct(hh);
11872     hh.alloc = *alloc;
```

```
11873
11874     if (result->w > 64)
11875         scanline = (float *) alloc->alloc(alloc->userdata,0, (nk_size)(result->w*2+1) *
      sizeof(float));
11876     else scanline = scanline_data;
11877
11878     scanline2 = scanline + result->w;
11879     y = off_y;
11880     e[n].y0 = (float) (off_y + result->h) + 1;
11881
11882     while (j < result->h)
11883     {
11884         /* find center of pixel for this scanline */
11885         float scan_y_top    = (float)y + 0.0f;
11886         float scan_y_bottom = (float)y + 1.0f;
11887         struct nk_tt__active_edge **step = &active;
11888
11889         NK_MEMSET(scanline , 0, (nk_size)result->w*sizeof(scanline[0]));
11890         NK_MEMSET(scanline2, 0, (nk_size)(result->w+1)*sizeof(scanline[0]));
11891
11892         /* update all active edges; */
11893         /* remove all active edges that terminate before the top of this scanline */
11894         while (*step) {
11895             struct nk_tt__active_edge * z = *step;
11896             if (z->ey <= scan_y_top) {
11897                 *step = z->next; /* delete from list */
11898                 NK_ASSERT(z->direction);
11899                 z->direction = 0;
11900                 nk_tt__hheap_free(&hh, z);
11901             } else {
11902                 step = &((*step)->next); /* advance through list */
11903             }
11904         }
11905
11906         /* insert all edges that start before the bottom of this scanline */
11907         while (e->y0 <= scan_y_bottom) {
11908             if (e->y0 != e->y1) {
11909                 struct nk_tt__active_edge *z = nk_tt__new_active(&hh, e, off_x, scan_y_top);
11910                 if (z != 0) {
11911                     NK_ASSERT(z->ey >= scan_y_top);
11912                     /* insert at front */
11913                     z->next = active;
11914                     active = z;
11915                 }
11916             }
11917             ++e;
11918         }
11919
11920         /* now process all active edges */
11921         if (active)
11922             nk_tt__fill_active_edges_new(scanline, scanline2+1, result->w, active, scan_y_top);
11923
11924         {
11925             float sum = 0;
11926             for (i=0; i < result->w; ++i) {
11927                 float k;
11928                 int m;
11929                 sum += scanline2[i];
11930                 k = scanline[i] + sum;
11931                 k = (float) NK_ABS(k) * 255.0f + 0.5f;
11932                 m = (int) k;
11933                 if (m > 255) m = 255;
11934                 result->pixels[j*result->stride + i] = (unsigned char) m;
11935             }
11936         }
11937         /* advance all the edges */
11938         step = &active;
11939         while (*step) {
11940             struct nk_tt__active_edge *z = *step;
11941             z->fx += z->fdx; /* advance to position for current scanline */
11942             step = &((*step)->next); /* advance through list */
11943         }
11944         ++y;
11945         ++j;
11946     }
11947     nk_tt__hheap_cleanup(&hh);
11948     if (scanline != scanline_data)
11949         alloc->free(alloc->userdata, scanline);
11950 }
11951 NK_INTERN void
11952 nk_tt__sort_edges_ins_sort(struct nk_tt__edge *p, int n)
11953 {
11954     int i,j;
11955     #define NK_TT__COMPARE(a,b)  ((a)->y0 < (b)->y0)
11956     for (i=1; i < n; ++i) {
11957         struct nk_tt__edge t = p[i], *a = &t;
11958         j = i;
```

```
11959         while (j > 0) {
11960             struct nk_tt__edge *b = &p[j-1];
11961             int c = NK_TT__COMPARE(a,b);
11962             if (!c) break;
11963             p[j] = p[j-1];
11964             --j;
11965         }
11966         if (i != j)
11967             p[j] = t;
11968     }
11969 }
11970 NK_INTERN void
11971 nk_tt__sort_edges_quicksort(struct nk_tt__edge *p, int n)
11972 {
11973     /* threshold for transitioning to insertion sort */
11974     while (n > 12) {
11975         struct nk_tt__edge t;
11976         int c01,c12,c,m,i,j;
11977
11978         /* compute median of three */
11979         m = n >> 1;
11980         c01 = NK_TT__COMPARE(&p[0],&p[m]);
11981         c12 = NK_TT__COMPARE(&p[m],&p[n-1]);
11982
11983         /* if 0 >= mid >= end, or 0 < mid < end, then use mid */
11984         if (c01 != c12) {
11985             /* otherwise, we'll need to swap something else to middle */
11986             int z;
11987             c = NK_TT__COMPARE(&p[0],&p[n-1]);
11988             /* 0>mid && mid<n:  0>n => n; 0<n => 0 */
11989             /* 0<mid && mid>n:  0>n => 0; 0<n => n */
11990             z = (c == c12) ? 0 : n-1;
11991             t = p[z];
11992             p[z] = p[m];
11993             p[m] = t;
11994         }
11995
11996         /* now p[m] is the median-of-three */
11997         /* swap it to the beginning so it won't move around */
11998         t = p[0];
11999         p[0] = p[m];
12000         p[m] = t;
12001
12002         /* partition loop */
12003         i=1;
12004         j=n-1;
12005         for(;;) {
12006             /* handling of equality is crucial here */
12007             /* for sentinels & efficiency with duplicates */
12008             for (;;++i) {
12009                 if (!NK_TT__COMPARE(&p[i], &p[0])) break;
12010             }
12011             for (;;--j) {
12012                 if (!NK_TT__COMPARE(&p[0], &p[j])) break;
12013             }
12014
12015             /* make sure we haven't crossed */
12016              if (i >= j) break;
12017              t = p[i];
12018              p[i] = p[j];
12019              p[j] = t;
12020
12021             ++i;
12022             --j;
12023
12024         }
12025
12026         /* recurse on smaller side, iterate on larger */
12027         if (j < (n-i)) {
12028             nk_tt__sort_edges_quicksort(p,j);
12029             p = p+i;
12030             n = n-i;
12031         } else {
12032             nk_tt__sort_edges_quicksort(p+i, n-i);
12033             n = j;
12034         }
12035     }
12036 }
12037 NK_INTERN void
12038 nk_tt__sort_edges(struct nk_tt__edge *p, int n)
12039 {
12040    nk_tt__sort_edges_quicksort(p, n);
12041    nk_tt__sort_edges_ins_sort(p, n);
12042 }
12043 NK_INTERN void
12044 nk_tt__rasterize(struct nk_tt__bitmap *result, struct nk_tt__point *pts,
12045     int *wcount, int windings, float scale_x, float scale_y,
```

```
12046     float shift_x, float shift_y, int off_x, int off_y, int invert,
12047     struct nk_allocator *alloc)
12048 {
12049     float y_scale_inv = invert ? -scale_y : scale_y;
12050     struct nk_tt__edge *e;
12051     int n,i,j,k,m;
12052     int vsubsample = 1;
12053     /* vsubsample should divide 255 evenly; otherwise we won't reach full opacity */
12054
12055     /* now we have to blow out the windings into explicit edge lists */
12056     n = 0;
12057     for (i=0; i < windings; ++i)
12058         n += wcount[i];
12059
12060     e = (struct nk_tt__edge*)
12061        alloc->alloc(alloc->userdata, 0,(sizeof(*e) * (nk_size)(n+1)));
12062     if (e == 0) return;
12063     n = 0;
12064
12065     m=0;
12066     for (i=0; i < windings; ++i)
12067     {
12068         struct nk_tt__point *p = pts + m;
12069         m += wcount[i];
12070         j = wcount[i]-1;
12071         for (k=0; k < wcount[i]; j=k++) {
12072             int a=k,b=j;
12073             /* skip the edge if horizontal */
12074             if (p[j].y == p[k].y)
12075                 continue;
12076
12077             /* add edge from j to k to the list */
12078             e[n].invert = 0;
12079             if (invert ? p[j].y > p[k].y : p[j].y < p[k].y) {
12080                 e[n].invert = 1;
12081                 a=j,b=k;
12082             }
12083             e[n].x0 = p[a].x * scale_x + shift_x;
12084             e[n].y0 = (p[a].y * y_scale_inv + shift_y) * (float)vsubsample;
12085             e[n].x1 = p[b].x * scale_x + shift_x;
12086             e[n].y1 = (p[b].y * y_scale_inv + shift_y) * (float)vsubsample;
12087             ++n;
12088         }
12089     }
12090
12091     /* now sort the edges by their highest point (should snap to integer, and then by x) */
12092     /*STBTT_sort(e, n, sizeof(e[0]), nk_tt__edge_compare); */
12093     nk_tt__sort_edges(e, n);
12094     /* now, traverse the scanlines and find the intersections on each scanline, use xor winding rule
       */
12095     nk_tt__rasterize_sorted_edges(result, e, n, vsubsample, off_x, off_y, alloc);
12096     alloc->free(alloc->userdata, e);
12097 }
12098 NK_INTERN void
12099 nk_tt__add_point(struct nk_tt__point *points, int n, float x, float y)
12100 {
12101     if (!points) return; /* during first pass, it's unallocated */
12102     points[n].x = x;
12103     points[n].y = y;
12104 }
12105 NK_INTERN int
12106 nk_tt__tesselate_curve(struct nk_tt__point *points, int *num_points,
12107     float x0, float y0, float x1, float y1, float x2, float y2,
12108     float objspace_flatness_squared, int n)
12109 {
12110     /* tesselate until threshold p is happy...
12111      * @TODO warped to compensate for non-linear stretching */
12112     /* midpoint */
12113     float mx = (x0 + 2*x1 + x2)/4;
12114     float my = (y0 + 2*y1 + y2)/4;
12115     /* versus directly drawn line */
12116     float dx = (x0+x2)/2 - mx;
12117     float dy = (y0+y2)/2 - my;
12118     if (n > 16) /* 65536 segments on one curve better be enough! */
12119         return 1;
12120
12121     /* half-pixel error allowed... need to be smaller if AA */
12122     if (dx*dx+dy*dy > objspace_flatness_squared) {
12123         nk_tt__tesselate_curve(points, num_points, x0,y0,
12124             (x0+x1)/2.0f,(y0+y1)/2.0f, mx,my, objspace_flatness_squared,n+1);
12125         nk_tt__tesselate_curve(points, num_points, mx,my,
12126             (x1+x2)/2.0f,(y1+y2)/2.0f, x2,y2, objspace_flatness_squared,n+1);
12127     } else {
12128         nk_tt__add_point(points, *num_points,x2,y2);
12129         *num_points = *num_points+1;
12130     }
12131     return 1;
```

```
12132 }
12133 NK_INTERN struct nk_tt__point*
12134 nk_tt_FlattenCurves(struct nk_tt_vertex *vertices, int num_verts,
12135     float objspace_flatness, int **contour_lengths, int *num_contours,
12136     struct nk_allocator *alloc)
12137 {
12138     /* returns number of contours */
12139     struct nk_tt__point *points=0;
12140     int num_points=0;
12141     float objspace_flatness_squared = objspace_flatness * objspace_flatness;
12142     int i;
12143     int n=0;
12144     int start=0;
12145     int pass;
12146
12147     /* count how many "moves" there are to get the contour count */
12148     for (i=0; i < num_verts; ++i)
12149         if (vertices[i].type == NK_TT_vmove) ++n;
12150
12151     *num_contours = n;
12152     if (n == 0) return 0;
12153
12154     *contour_lengths = (int *)
12155         alloc->alloc(alloc->userdata,0, (sizeof(**contour_lengths) * (nk_size)n));
12156     if (*contour_lengths == 0) {
12157         *num_contours = 0;
12158         return 0;
12159     }
12160
12161     /* make two passes through the points so we don't need to realloc */
12162     for (pass=0; pass < 2; ++pass)
12163     {
12164         float x=0,y=0;
12165         if (pass == 1) {
12166             points = (struct nk_tt__point *)
12167                 alloc->alloc(alloc->userdata,0, (nk_size)num_points * sizeof(points[0]));
12168             if (points == 0) goto error;
12169         }
12170         num_points = 0;
12171         n= -1;
12172
12173         for (i=0; i < num_verts; ++i)
12174         {
12175             switch (vertices[i].type) {
12176             case NK_TT_vmove:
12177                 /* start the next contour */
12178                 if (n >= 0)
12179                 (*contour_lengths)[n] = num_points - start;
12180                 ++n;
12181                 start = num_points;
12182
12183                 x = vertices[i].x, y = vertices[i].y;
12184                 nk_tt__add_point(points, num_points++, x,y);
12185                 break;
12186             case NK_TT_vline:
12187                x = vertices[i].x, y = vertices[i].y;
12188                nk_tt__add_point(points, num_points++, x, y);
12189                break;
12190             case NK_TT_vcurve:
12191                nk_tt__tesselate_curve(points, &num_points, x,y,
12192                                         vertices[i].cx, vertices[i].cy,
12193                                         vertices[i].x,  vertices[i].y,
12194                                         objspace_flatness_squared, 0);
12195                x = vertices[i].x, y = vertices[i].y;
12196                break;
12197             default: break;
12198          }
12199       }
12200       (*contour_lengths)[n] = num_points - start;
12201    }
12202    return points;
12203
12204 error:
12205    alloc->free(alloc->userdata, points);
12206    alloc->free(alloc->userdata, *contour_lengths);
12207    *contour_lengths = 0;
12208    *num_contours = 0;
12209    return 0;
12210 }
12211 NK_INTERN void
12212 nk_tt_Rasterize(struct nk_tt__bitmap *result, float flatness_in_pixels,
12213     struct nk_tt_vertex *vertices, int num_verts,
12214     float scale_x, float scale_y, float shift_x, float shift_y,
12215     int x_off, int y_off, int invert, struct nk_allocator *alloc)
12216 {
12217     float scale = scale_x > scale_y ? scale_y : scale_x;
12218     int winding_count, *winding_lengths;
```

```
12219     struct nk_tt__point *windings = nk_tt_FlattenCurves(vertices, num_verts,
12220         flatness_in_pixels / scale, &winding_lengths, &winding_count, alloc);
12221
12222     NK_ASSERT(alloc);
12223     if (windings) {
12224         nk_tt__rasterize(result, windings, winding_lengths, winding_count,
12225             scale_x, scale_y, shift_x, shift_y, x_off, y_off, invert, alloc);
12226         alloc->free(alloc->userdata, winding_lengths);
12227         alloc->free(alloc->userdata, windings);
12228     }
12229 }
12230 NK_INTERN void
12231 nk_tt_MakeGlyphBitmapSubpixel(const struct nk_tt_fontinfo *info, unsigned char *output,
12232     int out_w, int out_h, int out_stride, float scale_x, float scale_y,
12233     float shift_x, float shift_y, int glyph, struct nk_allocator *alloc)
12234 {
12235     int ix0,iy0;
12236     struct nk_tt_vertex *vertices;
12237     int num_verts = nk_tt_GetGlyphShape(info, alloc, glyph, &vertices);
12238     struct nk_tt__bitmap gbm;
12239
12240     nk_tt_GetGlyphBitmapBoxSubpixel(info, glyph, scale_x, scale_y, shift_x,
12241         shift_y, &ix0,&iy0,0,0);
12242     gbm.pixels = output;
12243     gbm.w = out_w;
12244     gbm.h = out_h;
12245     gbm.stride = out_stride;
12246
12247     if (gbm.w && gbm.h)
12248         nk_tt_Rasterize(&gbm, 0.35f, vertices, num_verts, scale_x, scale_y,
12249             shift_x, shift_y, ix0,iy0, 1, alloc);
12250     alloc->free(alloc->userdata, vertices);
12251 }
12252
12253 /*-------------------------------------------------------------
12254  *                          Bitmap baking
12255  * ------------------------------------------------------------*/
12256 NK_INTERN int
12257 nk_tt_PackBegin(struct nk_tt_pack_context *spc, unsigned char *pixels,
12258     int pw, int ph, int stride_in_bytes, int padding, struct nk_allocator *alloc)
12259 {
12260     int num_nodes = pw - padding;
12261     struct nk_rp_context *context = (struct nk_rp_context *)
12262         alloc->alloc(alloc->userdata,0, sizeof(*context));
12263     struct nk_rp_node *nodes = (struct nk_rp_node*)
12264         alloc->alloc(alloc->userdata,0, (sizeof(*nodes  ) * (nk_size)num_nodes));
12265
12266     if (context == 0 || nodes == 0) {
12267         if (context != 0) alloc->free(alloc->userdata, context);
12268         if (nodes   != 0) alloc->free(alloc->userdata, nodes);
12269         return 0;
12270     }
12271
12272     spc->width = pw;
12273     spc->height = ph;
12274     spc->pixels = pixels;
12275     spc->pack_info = context;
12276     spc->nodes = nodes;
12277     spc->padding = padding;
12278     spc->stride_in_bytes = (stride_in_bytes != 0) ? stride_in_bytes : pw;
12279     spc->h_oversample = 1;
12280     spc->v_oversample = 1;
12281
12282     nk_rp_init_target(context, pw-padding, ph-padding, nodes, num_nodes);
12283     if (pixels)
12284         NK_MEMSET(pixels, 0, (nk_size)(pw*ph)); /* background of 0 around pixels */
12285     return 1;
12286 }
12287 NK_INTERN void
12288 nk_tt_PackEnd(struct nk_tt_pack_context *spc, struct nk_allocator *alloc)
12289 {
12290     alloc->free(alloc->userdata, spc->nodes);
12291     alloc->free(alloc->userdata, spc->pack_info);
12292 }
12293 NK_INTERN void
12294 nk_tt_PackSetOversampling(struct nk_tt_pack_context *spc,
12295     unsigned int h_oversample, unsigned int v_oversample)
12296 {
12297    NK_ASSERT(h_oversample <= NK_TT_MAX_OVERSAMPLE);
12298    NK_ASSERT(v_oversample <= NK_TT_MAX_OVERSAMPLE);
12299    if (h_oversample <= NK_TT_MAX_OVERSAMPLE)
12300       spc->h_oversample = h_oversample;
12301    if (v_oversample <= NK_TT_MAX_OVERSAMPLE)
12302       spc->v_oversample = v_oversample;
12303 }
12304 NK_INTERN void
12305 nk_tt__h_prefilter(unsigned char *pixels, int w, int h, int stride_in_bytes,
```

```
12306     int kernel_width)
12307 {
12308     unsigned char buffer[NK_TT_MAX_OVERSAMPLE];
12309     int safe_w = w - kernel_width;
12310     int j;
12311
12312     for (j=0; j < h; ++j)
12313     {
12314         int i;
12315         unsigned int total;
12316         NK_MEMSET(buffer, 0, (nk_size)kernel_width);
12317
12318         total = 0;
12319
12320         /* make kernel_width a constant in common cases so compiler can optimize out the divide */
12321         switch (kernel_width) {
12322         case 2:
12323             for (i=0; i <= safe_w; ++i) {
12324                 total += (unsigned int)(pixels[i] - buffer[i & NK_TT__OVER_MASK]);
12325                 buffer[(i+kernel_width) & NK_TT__OVER_MASK] = pixels[i];
12326                 pixels[i] = (unsigned char) (total / 2);
12327             }
12328             break;
12329         case 3:
12330             for (i=0; i <= safe_w; ++i) {
12331                 total += (unsigned int)(pixels[i] - buffer[i & NK_TT__OVER_MASK]);
12332                 buffer[(i+kernel_width) & NK_TT__OVER_MASK] = pixels[i];
12333                 pixels[i] = (unsigned char) (total / 3);
12334             }
12335             break;
12336         case 4:
12337             for (i=0; i <= safe_w; ++i) {
12338                 total += (unsigned int)pixels[i] - buffer[i & NK_TT__OVER_MASK];
12339                 buffer[(i+kernel_width) & NK_TT__OVER_MASK] = pixels[i];
12340                 pixels[i] = (unsigned char) (total / 4);
12341             }
12342             break;
12343         case 5:
12344             for (i=0; i <= safe_w; ++i) {
12345                 total += (unsigned int)(pixels[i] - buffer[i & NK_TT__OVER_MASK]);
12346                 buffer[(i+kernel_width) & NK_TT__OVER_MASK] = pixels[i];
12347                 pixels[i] = (unsigned char) (total / 5);
12348             }
12349             break;
12350         default:
12351             for (i=0; i <= safe_w; ++i) {
12352                 total += (unsigned int)(pixels[i] - buffer[i & NK_TT__OVER_MASK]);
12353                 buffer[(i+kernel_width) & NK_TT__OVER_MASK] = pixels[i];
12354                 pixels[i] = (unsigned char) (total / (unsigned int)kernel_width);
12355             }
12356             break;
12357         }
12358
12359         for (; i < w; ++i) {
12360             NK_ASSERT(pixels[i] == 0);
12361             total -= (unsigned int)(buffer[i & NK_TT__OVER_MASK]);
12362             pixels[i] = (unsigned char) (total / (unsigned int)kernel_width);
12363         }
12364         pixels += stride_in_bytes;
12365     }
12366 }
12367 NK_INTERN void
12368 nk_tt__v_prefilter(unsigned char *pixels, int w, int h, int stride_in_bytes,
12369     int kernel_width)
12370 {
12371     unsigned char buffer[NK_TT_MAX_OVERSAMPLE];
12372     int safe_h = h - kernel_width;
12373     int j;
12374
12375     for (j=0; j < w; ++j)
12376     {
12377         int i;
12378         unsigned int total;
12379         NK_MEMSET(buffer, 0, (nk_size)kernel_width);
12380
12381         total = 0;
12382
12383         /* make kernel_width a constant in common cases so compiler can optimize out the divide */
12384         switch (kernel_width) {
12385         case 2:
12386             for (i=0; i <= safe_h; ++i) {
12387                 total += (unsigned int)(pixels[i*stride_in_bytes] - buffer[i & NK_TT__OVER_MASK]);
12388                 buffer[(i+kernel_width) & NK_TT__OVER_MASK] = pixels[i*stride_in_bytes];
12389                 pixels[i*stride_in_bytes] = (unsigned char) (total / 2);
12390             }
12391             break;
12392          case 3:
```

```
12393             for (i=0; i <= safe_h; ++i) {
12394                 total += (unsigned int)(pixels[i*stride_in_bytes] - buffer[i & NK_TT__OVER_MASK]);
12395                 buffer[(i+kernel_width) & NK_TT__OVER_MASK] = pixels[i*stride_in_bytes];
12396                 pixels[i*stride_in_bytes] = (unsigned char) (total / 3);
12397             }
12398             break;
12399          case 4:
12400             for (i=0; i <= safe_h; ++i) {
12401                 total += (unsigned int)(pixels[i*stride_in_bytes] - buffer[i & NK_TT__OVER_MASK]);
12402                 buffer[(i+kernel_width) & NK_TT__OVER_MASK] = pixels[i*stride_in_bytes];
12403                 pixels[i*stride_in_bytes] = (unsigned char) (total / 4);
12404             }
12405             break;
12406          case 5:
12407             for (i=0; i <= safe_h; ++i) {
12408                 total += (unsigned int)(pixels[i*stride_in_bytes] - buffer[i & NK_TT__OVER_MASK]);
12409                 buffer[(i+kernel_width) & NK_TT__OVER_MASK] = pixels[i*stride_in_bytes];
12410                 pixels[i*stride_in_bytes] = (unsigned char) (total / 5);
12411             }
12412             break;
12413          default:
12414             for (i=0; i <= safe_h; ++i) {
12415                 total += (unsigned int)(pixels[i*stride_in_bytes] - buffer[i & NK_TT__OVER_MASK]);
12416                 buffer[(i+kernel_width) & NK_TT__OVER_MASK] = pixels[i*stride_in_bytes];
12417                 pixels[i*stride_in_bytes] = (unsigned char) (total / (unsigned int)kernel_width);
12418             }
12419             break;
12420          }
12421
12422          for (; i < h; ++i) {
12423             NK_ASSERT(pixels[i*stride_in_bytes] == 0);
12424             total -= (unsigned int)(buffer[i & NK_TT__OVER_MASK]);
12425             pixels[i*stride_in_bytes] = (unsigned char) (total / (unsigned int)kernel_width);
12426          }
12427          pixels += 1;
12428      }
12429 }
12430 NK_INTERN float
12431 nk_tt__oversample_shift(int oversample)
12432 {
12433     if (!oversample)
12434         return 0.0f;
12435
12436     /* The prefilter is a box filter of width "oversample", */
12437     /* which shifts phase by (oversample - 1)/2 pixels in */
12438     /* oversampled space. We want to shift in the opposite */
12439     /* direction to counter this. */
12440     return (float)-(oversample - 1) / (2.0f * (float)oversample);
12441 }
12442 NK_INTERN int
12443 nk_tt_PackFontRangesGatherRects(struct nk_tt_pack_context *spc,
12444     struct nk_tt_fontinfo *info, struct nk_tt_pack_range *ranges,
12445     int num_ranges, struct nk_rp_rect *rects)
12446 {
12447     /* rects array must be big enough to accommodate all characters in the given ranges */
12448     int i,j,k;
12449     k = 0;
12450
12451     for (i=0; i < num_ranges; ++i) {
12452         float fh = ranges[i].font_size;
12453         float scale = (fh > 0) ? nk_tt_ScaleForPixelHeight(info, fh):
12454             nk_tt_ScaleForMappingEmToPixels(info, -fh);
12455         ranges[i].h_oversample = (unsigned char) spc->h_oversample;
12456         ranges[i].v_oversample = (unsigned char) spc->v_oversample;
12457         for (j=0; j < ranges[i].num_chars; ++j) {
12458             int x0,y0,x1,y1;
12459             int codepoint = ranges[i].first_unicode_codepoint_in_range ?
12460                 ranges[i].first_unicode_codepoint_in_range + j :
12461                 ranges[i].array_of_unicode_codepoints[j];
12462
12463             int glyph = nk_tt_FindGlyphIndex(info, codepoint);
12464             nk_tt_GetGlyphBitmapBoxSubpixel(info,glyph, scale * (float)spc->h_oversample,
12465                 scale * (float)spc->v_oversample, 0,0, &x0,&y0,&x1,&y1);
12466             rects[k].w = (nk_rp_coord) (x1-x0 + spc->padding + (int)spc->h_oversample-1);
12467             rects[k].h = (nk_rp_coord) (y1-y0 + spc->padding + (int)spc->v_oversample-1);
12468             ++k;
12469         }
12470     }
12471     return k;
12472 }
12473 NK_INTERN int
12474 nk_tt_PackFontRangesRenderIntoRects(struct nk_tt_pack_context *spc,
12475     struct nk_tt_fontinfo *info, struct nk_tt_pack_range *ranges,
12476     int num_ranges, struct nk_rp_rect *rects, struct nk_allocator *alloc)
12477 {
12478     int i,j,k, return_value = 1;
12479     /* save current values */
```

```
12480     int old_h_over = (int)spc->h_oversample;
12481     int old_v_over = (int)spc->v_oversample;
12482     /* rects array must be big enough to accommodate all characters in the given ranges */
12483
12484     k = 0;
12485     for (i=0; i < num_ranges; ++i)
12486     {
12487         float fh = ranges[i].font_size;
12488         float recip_h,recip_v,sub_x,sub_y;
12489         float scale = fh > 0 ? nk_tt_ScaleForPixelHeight(info, fh):
12490             nk_tt_ScaleForMappingEmToPixels(info, -fh);
12491
12492         spc->h_oversample = ranges[i].h_oversample;
12493         spc->v_oversample = ranges[i].v_oversample;
12494
12495         recip_h = 1.0f / (float)spc->h_oversample;
12496         recip_v = 1.0f / (float)spc->v_oversample;
12497
12498         sub_x = nk_tt__oversample_shift((int)spc->h_oversample);
12499         sub_y = nk_tt__oversample_shift((int)spc->v_oversample);
12500
12501         for (j=0; j < ranges[i].num_chars; ++j)
12502         {
12503             struct nk_rp_rect *r = &rects[k];
12504             if (r->was_packed)
12505             {
12506                 struct nk_tt_packedchar *bc = &ranges[i].chardata_for_range[j];
12507                 int advance, lsb, x0,y0,x1,y1;
12508                 int codepoint = ranges[i].first_unicode_codepoint_in_range ?
12509                     ranges[i].first_unicode_codepoint_in_range + j :
12510                     ranges[i].array_of_unicode_codepoints[j];
12511                 int glyph = nk_tt_FindGlyphIndex(info, codepoint);
12512                 nk_rp_coord pad = (nk_rp_coord) spc->padding;
12513
12514                 /* pad on left and top */
12515                 r->x = (nk_rp_coord)((int)r->x + (int)pad);
12516                 r->y = (nk_rp_coord)((int)r->y + (int)pad);
12517                 r->w = (nk_rp_coord)((int)r->w - (int)pad);
12518                 r->h = (nk_rp_coord)((int)r->h - (int)pad);
12519
12520                 nk_tt_GetGlyphHMetrics(info, glyph, &advance, &lsb);
12521                 nk_tt_GetGlyphBitmapBox(info, glyph, scale * (float)spc->h_oversample,
12522                         (scale * (float)spc->v_oversample), &x0,&y0,&x1,&y1);
12523                 nk_tt_MakeGlyphBitmapSubpixel(info, spc->pixels + r->x + r->y*spc->stride_in_bytes,
12524                     (int)(r->w - spc->h_oversample+1), (int)(r->h - spc->v_oversample+1),
12525                     spc->stride_in_bytes, scale * (float)spc->h_oversample,
12526                     scale * (float)spc->v_oversample, 0,0, glyph, alloc);
12527
12528                 if (spc->h_oversample > 1)
12529                    nk_tt__h_prefilter(spc->pixels + r->x + r->y*spc->stride_in_bytes,
12530                         r->w, r->h, spc->stride_in_bytes, (int)spc->h_oversample);
12531
12532                 if (spc->v_oversample > 1)
12533                    nk_tt__v_prefilter(spc->pixels + r->x + r->y*spc->stride_in_bytes,
12534                         r->w, r->h, spc->stride_in_bytes, (int)spc->v_oversample);
12535
12536                 bc->x0       = (nk_ushort)  r->x;
12537                 bc->y0       = (nk_ushort)  r->y;
12538                 bc->x1       = (nk_ushort) (r->x + r->w);
12539                 bc->y1       = (nk_ushort) (r->y + r->h);
12540                 bc->xadvance = scale * (float)advance;
12541                 bc->xoff     = (float)  x0 * recip_h + sub_x;
12542                 bc->yoff     = (float)  y0 * recip_v + sub_y;
12543                 bc->xoff2    = ((float)x0 + r->w) * recip_h + sub_x;
12544                 bc->yoff2    = ((float)y0 + r->h) * recip_v + sub_y;
12545             } else {
12546                 return_value = 0; /* if any fail, report failure */
12547             }
12548             ++k;
12549         }
12550     }
12551     /* restore original values */
12552     spc->h_oversample = (unsigned int)old_h_over;
12553     spc->v_oversample = (unsigned int)old_v_over;
12554     return return_value;
12555 }
12556 NK_INTERN void
12557 nk_tt_GetPackedQuad(struct nk_tt_packedchar *chardata, int pw, int ph,
12558     int char_index, float *xpos, float *ypos, struct nk_tt_aligned_quad *q,
12559     int align_to_integer)
12560 {
12561     float ipw = 1.0f / (float)pw, iph = 1.0f / (float)ph;
12562     struct nk_tt_packedchar *b = (struct nk_tt_packedchar*)(chardata + char_index);
12563     if (align_to_integer) {
12564         int tx = nk_ifloorf((*xpos + b->xoff) + 0.5f);
12565         int ty = nk_ifloorf((*ypos + b->yoff) + 0.5f);
12566
```

```
12567         float x = (float)tx;
12568         float y = (float)ty;
12569
12570         q->x0 = x;
12571         q->y0 = y;
12572         q->x1 = x + b->xoff2 - b->xoff;
12573         q->y1 = y + b->yoff2 - b->yoff;
12574     } else {
12575         q->x0 = *xpos + b->xoff;
12576         q->y0 = *ypos + b->yoff;
12577         q->x1 = *xpos + b->xoff2;
12578         q->y1 = *ypos + b->yoff2;
12579     }
12580     q->s0 = b->x0 * ipw;
12581     q->t0 = b->y0 * iph;
12582     q->s1 = b->x1 * ipw;
12583     q->t1 = b->y1 * iph;
12584     *xpos += b->xadvance;
12585 }
12586
12587 /* ---------------------------------------------------------------
12588  *
12589  *                          FONT BAKING
12590  *
12591  * ---------------------------------------------------------------*/
12592 struct nk_font_bake_data {
12593     struct nk_tt_fontinfo info;
12594     struct nk_rp_rect *rects;
12595     struct nk_tt_pack_range *ranges;
12596     nk_rune range_count;
12597 };
12598
12599 struct nk_font_baker {
12600     struct nk_allocator alloc;
12601     struct nk_tt_pack_context spc;
12602     struct nk_font_bake_data *build;
12603     struct nk_tt_packedchar *packed_chars;
12604     struct nk_rp_rect *rects;
12605     struct nk_tt_pack_range *ranges;
12606 };
12607
12608 NK_GLOBAL const nk_size nk_rect_align = NK_ALIGNOF(struct nk_rp_rect);
12609 NK_GLOBAL const nk_size nk_range_align = NK_ALIGNOF(struct nk_tt_pack_range);
12610 NK_GLOBAL const nk_size nk_char_align = NK_ALIGNOF(struct nk_tt_packedchar);
12611 NK_GLOBAL const nk_size nk_build_align = NK_ALIGNOF(struct nk_font_bake_data);
12612 NK_GLOBAL const nk_size nk_baker_align = NK_ALIGNOF(struct nk_font_baker);
12613
12614 NK_INTERN int
12615 nk_range_count(const nk_rune *range)
12616 {
12617     const nk_rune *iter = range;
12618     NK_ASSERT(range);
12619     if (!range) return 0;
12620     while (*(iter++) != 0);
12621     return (iter == range) ? 0 : (int)((iter - range)/2);
12622 }
12623 NK_INTERN int
12624 nk_range_glyph_count(const nk_rune *range, int count)
12625 {
12626     int i = 0;
12627     int total_glyphs = 0;
12628     for (i = 0; i < count; ++i) {
12629         int diff;
12630         nk_rune f = range[(i*2)+0];
12631         nk_rune t = range[(i*2)+1];
12632         NK_ASSERT(t >= f);
12633         diff = (int)((t - f) + 1);
12634         total_glyphs += diff;
12635     }
12636     return total_glyphs;
12637 }
12638 NK_API const nk_rune*
12639 nk_font_default_glyph_ranges(void)
12640 {
12641     NK_STORAGE const nk_rune ranges[] = {0x0020, 0x00FF, 0};
12642     return ranges;
12643 }
12644 NK_API const nk_rune*
12645 nk_font_chinese_glyph_ranges(void)
12646 {
12647     NK_STORAGE const nk_rune ranges[] = {
12648         0x0020, 0x00FF,
12649         0x3000, 0x30FF,
12650         0x31F0, 0x31FF,
12651         0xFF00, 0xFFEF,
12652         0x4e00, 0x9FAF,
12653         0
```

```
12654     };
12655     return ranges;
12656 }
12657 NK_API const nk_rune*
12658 nk_font_cyrillic_glyph_ranges(void)
12659 {
12660     NK_STORAGE const nk_rune ranges[] = {
12661         0x0020, 0x00FF,
12662         0x0400, 0x052F,
12663         0x2DE0, 0x2DFF,
12664         0xA640, 0xA69F,
12665         0
12666     };
12667     return ranges;
12668 }
12669 NK_API const nk_rune*
12670 nk_font_korean_glyph_ranges(void)
12671 {
12672     NK_STORAGE const nk_rune ranges[] = {
12673         0x0020, 0x00FF,
12674         0x3131, 0x3163,
12675         0xAC00, 0xD79D,
12676         0
12677     };
12678     return ranges;
12679 }
12680 NK_INTERN void
12681 nk_font_baker_memory(nk_size *temp, int *glyph_count,
12682     struct nk_font_config *config_list, int count)
12683 {
12684     int range_count = 0;
12685     int total_range_count = 0;
12686     struct nk_font_config *iter, *i;
12687
12688     NK_ASSERT(config_list);
12689     NK_ASSERT(glyph_count);
12690     if (!config_list) {
12691         *temp = 0;
12692         *glyph_count = 0;
12693         return;
12694     }
12695     *glyph_count = 0;
12696     for (iter = config_list; iter; iter = iter->next) {
12697         i = iter;
12698         do {if (!i->range) iter->range = nk_font_default_glyph_ranges();
12699             range_count = nk_range_count(i->range);
12700             total_range_count += range_count;
12701             *glyph_count += nk_range_glyph_count(i->range, range_count);
12702         } while ((i = i->n) != iter);
12703     }
12704     *temp = (nk_size)*glyph_count * sizeof(struct nk_rp_rect);
12705     *temp += (nk_size)total_range_count * sizeof(struct nk_tt_pack_range);
12706     *temp += (nk_size)*glyph_count * sizeof(struct nk_tt_packedchar);
12707     *temp += (nk_size)count * sizeof(struct nk_font_bake_data);
12708     *temp += sizeof(struct nk_font_baker);
12709     *temp += nk_rect_align + nk_range_align + nk_char_align;
12710     *temp += nk_build_align + nk_baker_align;
12711 }
12712 NK_INTERN struct nk_font_baker*
12713 nk_font_baker(void *memory, int glyph_count, int count, struct nk_allocator *alloc)
12714 {
12715     struct nk_font_baker *baker;
12716     if (!memory) return 0;
12717     /* setup baker inside a memory block  */
12718     baker = (struct nk_font_baker*)NK_ALIGN_PTR(memory, nk_baker_align);
12719     baker->build = (struct nk_font_bake_data*)NK_ALIGN_PTR((baker + 1), nk_build_align);
12720     baker->packed_chars = (struct nk_tt_packedchar*)NK_ALIGN_PTR((baker->build + count),
      nk_char_align);
12721     baker->rects = (struct nk_rp_rect*)NK_ALIGN_PTR((baker->packed_chars + glyph_count),
      nk_rect_align);
12722     baker->ranges = (struct nk_tt_pack_range*)NK_ALIGN_PTR((baker->rects + glyph_count),
      nk_range_align);
12723     baker->alloc = *alloc;
12724     return baker;
12725 }
12726 NK_INTERN int
12727 nk_font_bake_pack(struct nk_font_baker *baker,
12728     nk_size *image_memory, int *width, int *height, struct nk_recti *custom,
12729     const struct nk_font_config *config_list, int count,
12730     struct nk_allocator *alloc)
12731 {
12732     NK_STORAGE const nk_size max_height = 1024 * 32;
12733     const struct nk_font_config *config_iter, *it;
12734     int total_glyph_count = 0;
12735     int total_range_count = 0;
12736     int range_count = 0;
12737     int i = 0;
```

```
12738
12739     NK_ASSERT(image_memory);
12740     NK_ASSERT(width);
12741     NK_ASSERT(height);
12742     NK_ASSERT(config_list);
12743     NK_ASSERT(count);
12744     NK_ASSERT(alloc);
12745
12746     if (!image_memory || !width || !height || !config_list || !count) return nk_false;
12747     for (config_iter = config_list; config_iter; config_iter = config_iter->next) {
12748         it = config_iter;
12749         do {range_count = nk_range_count(it->range);
12750             total_range_count += range_count;
12751             total_glyph_count += nk_range_glyph_count(it->range, range_count);
12752         } while ((it = it->n) != config_iter);
12753     }
12754     /* setup font baker from temporary memory */
12755     for (config_iter = config_list; config_iter; config_iter = config_iter->next) {
12756         it = config_iter;
12757         do {if (!nk_tt_InitFont(&baker->build[i++].info, (const unsigned char*)it->ttf_blob, 0))
12758             return nk_false;
12759         } while ((it = it->n) != config_iter);
12760     }
12761     *height = 0;
12762     *width = (total_glyph_count > 1000) ? 1024 : 512;
12763     nk_tt_PackBegin(&baker->spc, 0, (int)*width, (int)max_height, 0, 1, alloc);
12764     {
12765         int input_i = 0;
12766         int range_n = 0;
12767         int rect_n = 0;
12768         int char_n = 0;
12769
12770         if (custom) {
12771             /* pack custom user data first so it will be in the upper left corner*/
12772             struct nk_rp_rect custom_space;
12773             nk_zero(&custom_space, sizeof(custom_space));
12774             custom_space.w = (nk_rp_coord)(custom->w);
12775             custom_space.h = (nk_rp_coord)(custom->h);
12776
12777             nk_tt_PackSetOversampling(&baker->spc, 1, 1);
12778             nk_rp_pack_rects((struct nk_rp_context*)baker->spc.pack_info, &custom_space, 1);
12779             *height = NK_MAX(*height, (int)(custom_space.y + custom_space.h));
12780
12781             custom->x = (short)custom_space.x;
12782             custom->y = (short)custom_space.y;
12783             custom->w = (short)custom_space.w;
12784             custom->h = (short)custom_space.h;
12785         }
12786
12787         /* first font pass: pack all glyphs */
12788         for (input_i = 0, config_iter = config_list; input_i < count && config_iter;
12789             config_iter = config_iter->next) {
12790             it = config_iter;
12791             do {int n = 0;
12792                 int glyph_count;
12793                 const nk_rune *in_range;
12794                 const struct nk_font_config *cfg = it;
12795                 struct nk_font_bake_data *tmp = &baker->build[input_i++];
12796
12797                 /* count glyphs + ranges in current font */
12798                 glyph_count = 0; range_count = 0;
12799                 for (in_range = cfg->range; in_range[0] && in_range[1]; in_range += 2) {
12800                     glyph_count += (int)(in_range[1] - in_range[0]) + 1;
12801                     range_count++;
12802                 }
12803
12804                 /* setup ranges  */
12805                 tmp->ranges = baker->ranges + range_n;
12806                 tmp->range_count = (nk_rune)range_count;
12807                 range_n += range_count;
12808                 for (i = 0; i < range_count; ++i) {
12809                     in_range = &cfg->range[i * 2];
12810                     tmp->ranges[i].font_size = cfg->size;
12811                     tmp->ranges[i].first_unicode_codepoint_in_range = (int)in_range[0];
12812                     tmp->ranges[i].num_chars = (int)(in_range[1]- in_range[0]) + 1;
12813                     tmp->ranges[i].chardata_for_range = baker->packed_chars + char_n;
12814                     char_n += tmp->ranges[i].num_chars;
12815                 }
12816
12817                 /* pack */
12818                 tmp->rects = baker->rects + rect_n;
12819                 rect_n += glyph_count;
12820                 nk_tt_PackSetOversampling(&baker->spc, cfg->oversample_h, cfg->oversample_v);
12821                 n = nk_tt_PackFontRangesGatherRects(&baker->spc, &tmp->info,
12822                     tmp->ranges, (int)tmp->range_count, tmp->rects);
12823                 nk_rp_pack_rects((struct nk_rp_context*)baker->spc.pack_info, tmp->rects, (int)n);
12824
```

```
12825                 /* texture height */
12826                 for (i = 0; i < n; ++i) {
12827                     if (tmp->rects[i].was_packed)
12828                         *height = NK_MAX(*height, tmp->rects[i].y + tmp->rects[i].h);
12829                 }
12830             } while ((it = it->n) != config_iter);
12831         }
12832         NK_ASSERT(rect_n == total_glyph_count);
12833         NK_ASSERT(char_n == total_glyph_count);
12834         NK_ASSERT(range_n == total_range_count);
12835     }
12836     *height = (int)nk_round_up_pow2((nk_uint)*height);
12837     *image_memory = (nk_size)(*width) * (nk_size)(*height);
12838     return nk_true;
12839 }
12840 NK_INTERN void
12841 nk_font_bake(struct nk_font_baker *baker, void *image_memory, int width, int height,
12842     struct nk_font_glyph *glyphs, int glyphs_count,
12843     const struct nk_font_config *config_list, int font_count)
12844 {
12845     int input_i = 0;
12846     nk_rune glyph_n = 0;
12847     const struct nk_font_config *config_iter;
12848     const struct nk_font_config *it;
12849
12850     NK_ASSERT(image_memory);
12851     NK_ASSERT(width);
12852     NK_ASSERT(height);
12853     NK_ASSERT(config_list);
12854     NK_ASSERT(baker);
12855     NK_ASSERT(font_count);
12856     NK_ASSERT(glyphs_count);
12857     if (!image_memory || !width || !height || !config_list ||
12858         !font_count || !glyphs || !glyphs_count)
12859         return;
12860
12861     /* second font pass: render glyphs */
12862     nk_zero(image_memory, (nk_size)((nk_size)width * (nk_size)height));
12863     baker->spc.pixels = (unsigned char*)image_memory;
12864     baker->spc.height = (int)height;
12865     for (input_i = 0, config_iter = config_list; input_i < font_count && config_iter;
12866         config_iter = config_iter->next) {
12867         it = config_iter;
12868         do {const struct nk_font_config *cfg = it;
12869             struct nk_font_bake_data *tmp = &baker->build[input_i++];
12870             nk_tt_PackSetOversampling(&baker->spc, cfg->oversample_h, cfg->oversample_v);
12871             nk_tt_PackFontRangesRenderIntoRects(&baker->spc, &tmp->info, tmp->ranges,
12872                 (int)tmp->range_count, tmp->rects, &baker->alloc);
12873         } while ((it = it->n) != config_iter);
12874     } nk_tt_PackEnd(&baker->spc, &baker->alloc);
12875
12876     /* third pass: setup font and glyphs */
12877     for (input_i = 0, config_iter = config_list; input_i < font_count && config_iter;
12878         config_iter = config_iter->next) {
12879         it = config_iter;
12880         do {nk_size i = 0;
12881             int char_idx = 0;
12882             nk_rune glyph_count = 0;
12883             const struct nk_font_config *cfg = it;
12884             struct nk_font_bake_data *tmp = &baker->build[input_i++];
12885             struct nk_baked_font *dst_font = cfg->font;
12886
12887             float font_scale = nk_tt_ScaleForPixelHeight(&tmp->info, cfg->size);
12888             int unscaled_ascent, unscaled_descent, unscaled_line_gap;
12889             nk_tt_GetFontVMetrics(&tmp->info, &unscaled_ascent, &unscaled_descent,
12890                                   &unscaled_line_gap);
12891
12892             /* fill baked font */
12893             if (!cfg->merge_mode) {
12894                 dst_font->ranges = cfg->range;
12895                 dst_font->height = cfg->size;
12896                 dst_font->ascent = ((float)unscaled_ascent * font_scale);
12897                 dst_font->descent = ((float)unscaled_descent * font_scale);
12898                 dst_font->glyph_offset = glyph_n;
12899                 // Need to zero this, or it will carry over from a previous
12900                 // bake, and cause a segfault when accessing glyphs[].
12901                 dst_font->glyph_count = 0;
12902             }
12903
12904             /* fill own baked font glyph array */
12905             for (i = 0; i < tmp->range_count; ++i) {
12906                 struct nk_tt_pack_range *range = &tmp->ranges[i];
12907                 for (char_idx = 0; char_idx < range->num_chars; char_idx++)
12908                 {
12909                     nk_rune codepoint = 0;
12910                     float dummy_x = 0, dummy_y = 0;
12911                     struct nk_tt_aligned_quad q;
```

```
12912                     struct nk_font_glyph *glyph;
12913
12914                     /* query glyph bounds from stb_truetype */
12915                     const struct nk_tt_packedchar *pc = &range->chardata_for_range[char_idx];
12916                     if (!pc->x0 && !pc->x1 && !pc->y0 && !pc->y1) continue;
12917                     codepoint = (nk_rune)(range->first_unicode_codepoint_in_range + char_idx);
12918                     nk_tt_GetPackedQuad(range->chardata_for_range, (int)width,
12919                         (int)height, char_idx, &dummy_x, &dummy_y, &q, 0);
12920
12921                     /* fill own glyph type with data */
12922                     glyph = &glyphs[dst_font->glyph_offset + dst_font->glyph_count + (unsigned
      int)glyph_count];
12923                     glyph->codepoint = codepoint;
12924                     glyph->x0 = q.x0; glyph->y0 = q.y0;
12925                     glyph->x1 = q.x1; glyph->y1 = q.y1;
12926                     glyph->y0 += (dst_font->ascent + 0.5f);
12927                     glyph->y1 += (dst_font->ascent + 0.5f);
12928                     glyph->w = glyph->x1 - glyph->x0 + 0.5f;
12929                     glyph->h = glyph->y1 - glyph->y0;
12930
12931                     if (cfg->coord_type == NK_COORD_PIXEL) {
12932                         glyph->u0 = q.s0 * (float)width;
12933                         glyph->v0 = q.t0 * (float)height;
12934                         glyph->u1 = q.s1 * (float)width;
12935                         glyph->v1 = q.t1 * (float)height;
12936                     } else {
12937                         glyph->u0 = q.s0;
12938                         glyph->v0 = q.t0;
12939                         glyph->u1 = q.s1;
12940                         glyph->v1 = q.t1;
12941                     }
12942                     glyph->xadvance = (pc->xadvance + cfg->spacing.x);
12943                     if (cfg->pixel_snap)
12944                         glyph->xadvance = (float)(int)(glyph->xadvance + 0.5f);
12945                     glyph_count++;
12946                 }
12947             }
12948             dst_font->glyph_count += glyph_count;
12949             glyph_n += glyph_count;
12950         } while ((it = it->n) != config_iter);
12951     }
12952 }
12953 NK_INTERN void
12954 nk_font_bake_custom_data(void *img_memory, int img_width, int img_height,
12955     struct nk_recti img_dst, const char *texture_data_mask, int tex_width,
12956     int tex_height, char white, char black)
12957 {
12958     nk_byte *pixels;
12959     int y = 0;
12960     int x = 0;
12961     int n = 0;
12962
12963     NK_ASSERT(img_memory);
12964     NK_ASSERT(img_width);
12965     NK_ASSERT(img_height);
12966     NK_ASSERT(texture_data_mask);
12967     NK_UNUSED(tex_height);
12968     if (!img_memory || !img_width || !img_height || !texture_data_mask)
12969         return;
12970
12971     pixels = (nk_byte*)img_memory;
12972     for (y = 0, n = 0; y < tex_height; ++y) {
12973         for (x = 0; x < tex_width; ++x, ++n) {
12974             const int off0 = ((img_dst.x + x) + (img_dst.y + y) * img_width);
12975             const int off1 = off0 + 1 + tex_width;
12976             pixels[off0] = (texture_data_mask[n] == white) ? 0xFF : 0x00;
12977             pixels[off1] = (texture_data_mask[n] == black) ? 0xFF : 0x00;
12978         }
12979     }
12980 }
12981 NK_INTERN void
12982 nk_font_bake_convert(void *out_memory, int img_width, int img_height,
12983     const void *in_memory)
12984 {
12985     int n = 0;
12986     nk_rune *dst;
12987     const nk_byte *src;
12988
12989     NK_ASSERT(out_memory);
12990     NK_ASSERT(in_memory);
12991     NK_ASSERT(img_width);
12992     NK_ASSERT(img_height);
12993     if (!out_memory || !in_memory || !img_height || !img_width) return;
12994
12995     dst = (nk_rune*)out_memory;
12996     src = (const nk_byte*)in_memory;
12997     for (n = (int)(img_width * img_height); n > 0; n--)
```

```
12998         *dst++ = ((nk_rune)(*src++) << 24) | 0x00FFFFFF;
12999 }
13000
13001 /* ---------------------------------------------------------------
13002  *
13003  *                          FONT
13004  *
13005  * ---------------------------------------------------------------*/
13006 NK_INTERN float
13007 nk_font_text_width(nk_handle handle, float height, const char *text, int len)
13008 {
13009     nk_rune unicode;
13010     int text_len  = 0;
13011     float text_width = 0;
13012     int glyph_len = 0;
13013     float scale = 0;
13014
13015     struct nk_font *font = (struct nk_font*)handle.ptr;
13016     NK_ASSERT(font);
13017     NK_ASSERT(font->glyphs);
13018     if (!font || !text || !len)
13019         return 0;
13020
13021     scale = height/font->info.height;
13022     glyph_len = text_len = nk_utf_decode(text, &unicode, (int)len);
13023     if (!glyph_len) return 0;
13024     while (text_len <= (int)len && glyph_len) {
13025         const struct nk_font_glyph *g;
13026         if (unicode == NK_UTF_INVALID) break;
13027
13028         /* query currently drawn glyph information */
13029         g = nk_font_find_glyph(font, unicode);
13030         text_width += g->xadvance * scale;
13031
13032         /* offset next glyph */
13033         glyph_len = nk_utf_decode(text + text_len, &unicode, (int)len - text_len);
13034         text_len += glyph_len;
13035     }
13036     return text_width;
13037 }
13038 #ifdef NK_INCLUDE_VERTEX_BUFFER_OUTPUT
13039 NK_INTERN void
13040 nk_font_query_font_glyph(nk_handle handle, float height,
13041     struct nk_user_font_glyph *glyph, nk_rune codepoint, nk_rune next_codepoint)
13042 {
13043     float scale;
13044     const struct nk_font_glyph *g;
13045     struct nk_font *font;
13046
13047     NK_ASSERT(glyph);
13048     NK_UNUSED(next_codepoint);
13049
13050     font = (struct nk_font*)handle.ptr;
13051     NK_ASSERT(font);
13052     NK_ASSERT(font->glyphs);
13053     if (!font || !glyph)
13054         return;
13055
13056     scale = height/font->info.height;
13057     g = nk_font_find_glyph(font, codepoint);
13058     glyph->width = (g->x1 - g->x0) * scale;
13059     glyph->height = (g->y1 - g->y0) * scale;
13060     glyph->offset = nk_vec2(g->x0 * scale, g->y0 * scale);
13061     glyph->xadvance = (g->xadvance * scale);
13062     glyph->uv[0] = nk_vec2(g->u0, g->v0);
13063     glyph->uv[1] = nk_vec2(g->u1, g->v1);
13064 }
13065 #endif
13066 NK_API const struct nk_font_glyph*
13067 nk_font_find_glyph(struct nk_font *font, nk_rune unicode)
13068 {
13069     int i = 0;
13070     int count;
13071     int total_glyphs = 0;
13072     const struct nk_font_glyph *glyph = 0;
13073     const struct nk_font_config *iter = 0;
13074
13075     NK_ASSERT(font);
13076     NK_ASSERT(font->glyphs);
13077     NK_ASSERT(font->info.ranges);
13078     if (!font || !font->glyphs) return 0;
13079
13080     glyph = font->fallback;
13081     iter = font->config;
13082     do {count = nk_range_count(iter->range);
13083         for (i = 0; i < count; ++i) {
13084             nk_rune f = iter->range[(i*2)+0];
```

```
13085             nk_rune t = iter->range[(i*2)+1];
13086             int diff = (int)((t - f) + 1);
13087             if (unicode >= f && unicode <= t)
13088                 return &font->glyphs[((nk_rune)total_glyphs + (unicode - f))];
13089             total_glyphs += diff;
13090         }
13091     } while ((iter = iter->n) != font->config);
13092     return glyph;
13093 }
13094 NK_INTERN void
13095 nk_font_init(struct nk_font *font, float pixel_height,
13096     nk_rune fallback_codepoint, struct nk_font_glyph *glyphs,
13097     const struct nk_baked_font *baked_font, nk_handle atlas)
13098 {
13099     struct nk_baked_font baked;
13100     NK_ASSERT(font);
13101     NK_ASSERT(glyphs);
13102     NK_ASSERT(baked_font);
13103     if (!font || !glyphs || !baked_font)
13104         return;
13105
13106     baked = *baked_font;
13107     font->fallback = 0;
13108     font->info = baked;
13109     font->scale = (float)pixel_height / (float)font->info.height;
13110     font->glyphs = &glyphs[baked_font->glyph_offset];
13111     font->texture = atlas;
13112     font->fallback_codepoint = fallback_codepoint;
13113     font->fallback = nk_font_find_glyph(font, fallback_codepoint);
13114
13115     font->handle.height = font->info.height * font->scale;
13116     font->handle.width = nk_font_text_width;
13117     font->handle.userdata.ptr = font;
13118 #ifdef NK_INCLUDE_VERTEX_BUFFER_OUTPUT
13119     font->handle.query = nk_font_query_font_glyph;
13120     font->handle.texture = font->texture;
13121 #endif
13122 }
13123
13124 /* -------------------------------------------------------------------------
13125  *
13126  *                              DEFAULT FONT
13127  *
13128  * ProggyClean.ttf
13129  * Copyright (c) 2004, 2005 Tristan Grimmer
13130  * MIT license (see License.txt in http://www.upperbounds.net/download/ProggyClean.ttf.zip)
13131  * Download and more information at http://upperbounds.net
13132  *-------------------------------------------------------------------------*/
13133 #ifdef __clang__
13134 #pragma clang diagnostic push
13135 #pragma clang diagnostic ignored "-Woverlength-strings"
13136 #elif defined(__GNUC__) || defined(__GNUG__)
13137 #pragma GCC diagnostic push
13138 #pragma GCC diagnostic ignored "-Woverlength-strings"
13139 #endif
13140
13141 #ifdef NK_INCLUDE_DEFAULT_FONT
13142
13143 NK_GLOBAL const char nk_proggy_clean_ttf_compressed_data_base85[11980+1] =
13144
    "7])#######hV0qs'/###[),##/l:$#Q6>##5[n42>c-TH`-»#/e>11NNV=Bv(*:.F?uu#(gRU.o0XGH`$vhLG1hxt9?W`#,5LsCp#-i>.r$<$6pD>Lb';
13145
    "2*>]b(MC;$jPfY.;h^`IWM9<Lh2TlS+f-s$o6Q<BWH`YiU.xfLq$N;$0iR/GX:U(jcW2p/W*q?-qmnUCI;jHSAiFWM.R*kU@C=GH?a9wp8f$e.-4^Qg1)(
13146
    "`8ND>Qo#t'X#(v#Y9w0#1D$CIf;W'#pWUPXOuxXuU(H9M(1<q-UE31#^-V'8IRUo7Qf./L>=Ke$$'5F%)]0^#0X@U.a<r:QLtFsLcL6##lOj)#.Y5<-R&
13147
    "i@^rMl9t=cWq6##weg>$FBjVQTSDgEKnIS7EM9>ZY9w0#L;»#Mx&4Mvt//L[MkA#W@lK.N'[0#7RL_&#w+F%HtG9M#XL`N&.,GM4Pg;-<nLENhvx>-VsM
13148
    "kfimM2J,W-jXS:)r0wK#@Fge$U>`w'N7G#$#fB#$E^$#:9:hk+eOe--6x)F7*E%?76%^GMHePW-Z5l'&GiF#$956:rS?dA#fiK:)Yr+`&#0j@'DbG&#^$
13149
    "*b=%Q6pia-Xg8I$<MR&,VdJe$<(7G;Ckl'&hF;;$<_=X(b.RS%%)###MPBuuE1V:v&cX&#2m#(&cV]`k9OhLMbn%s$G2,B$BfD3X*sp5#l,$R#]x_X1xK
13150
    "tm+/Us9pG)XPu`<0s-)WTt(gCRxIg(%6sfh=ktMKn3j)<6<b5Sk_/0(^]AaN#(p/L>&VZ>1i%h1S9u5o@YaaW$e+b<TWFn/Z:Oh(Cx2$lNEoN^e)#CFY@(
13151
    "ow0i(?$Q[cjOd[P4d)]>ROPOpxTO7Stwi1::iB1q)C_=dV26J;2,]7op$]uQr@_V7$q^%lQwtuHY]=DX,n3L#0PHDO4f9>dC@O>HBuKPpP*E,N+b3L#lpI
13152
    "x]Ip.PH^'/aqUO/$1WxLoW0[iLA<QT;5HKD+@qQ'NQ(3_PLhE48R.qAPSwQ0/WK?Z,[x?-J;jQTWA0X@KJ(_Y8N-:/M74:/-ZpKrUss?d#dZq]DAbkU*J
13153
    "CRUxHPeR`5Mjol(dUWxZa(>STrPkrJiWx`5U7F#.g*jrohGg`cg:lSTvEY/EV_7H4Q9[Z%cnv;JQYZ5q.l7Zeas:HOIZOB?G<Nald$qs]@]L<J7bR*>gv
13154
    "U]W]+fh18.vsai00);D3@4ku5P?DP8aJt+;qUM]=+b'8@;mViBKx0DE[-auGl8:PJ&Dj+M6OC]O^((##]`0i)drT;-7X`=-H3[igUnPG-NZlo.#k@h#=O
13155
    "'/###xe7q.73rI3*pP/$1>s9)W,JrM7SN]'/4C#v$U`0#V.[0>xQsH$fEmPMgY2u7Kh(G%siIfLSoS+MK2eTM$=5,M8p`A.;_R%#u[K#$x4AG8.kK/HSB
13156
    "_3YlQC7(p7q)&](`6_c)$/*JL(L-^(]$wIM`dPtOdGA,U3:w2M-0<q-]L_?^)1vw',MRsqVr.L;aN&#/EgJ)PBc[-f>+WomX2u7lqM2iEumMTcsF?-aT
13157
    "Ft(c%=;Am_Qs@jLooI&NX;]0#j4#F14;gl8-GQpgwhrq8'=l_f-b49'UOqkLu7-##oDY2L(te+Mch&gLYtJ,MEtJfLh'x'M=$CS-ZZ%P]8bZ>#S?YY#%Q
```

```
13158
"/oL?#h7gl85[qW/NDOk%16ij;+:1a'iNIdb-ou8.P*w,v5#EI$TWS>Pot-R*H'-SEpA:g)f+O$%%`kA#G=8RMmG1&O`>to8bC]T&$,n.LoO>29sp3dt-5
13159
"%a2E-grWVM3@2=-k22tL]4$##6We'8UJCKE[d_=%wI;'6X-GsLX4j^SgJ$##R*w,vP3wK#iiW&#*h^D&R?jp7+/u&#(AP##XU8c$fSYW-J95_-Dp[g9wc
13160
"OQFKNX@QI'IoPp7nb,QU//MQ&ZDkKP)X<WSVL(68uVl&#c'[0#(s1X&xm$Y%B7*K:eDA323j998GXbA#pwMs-jgD$9QISB-A_(aN4xoFM^@C58D0+Q+q3
13161
"h$hxua_K]ul92%'BOU&#BRRh-slg8KDlr:%L71Ka:.A;%YULjDPmL<LYs8i#XwJOYaKPKc1h:'9Ke,g)b),78=I39B;xiY$bgGw-&.Zi9InXDuYa%G*f2
13162
"o;#2:;%d&#x9v68C5g?ntX0X)pT`;%pB3q7mgGN)3%(P8nTd5L7GeA-GL@+%J3u2:(Yf>et`e;)f#Km8&+DC$I46>#Kr]]u-[=99tts1.qb#q72g1WJO8
13163
"j%2n8)),?ILR5^.Ibn<-X-Mq7[a82Lq:F&#ce+S9wsCK*x`569E8ew'He]h:sI[2LM$[guka3ZRd6:t%IG:;$%YiJ:Nq=?eAw;/:nnDq0(CYcMpG)qLN4
13164
"sj_$%[HK%'F####QRZJ::Y3EGl4'@%FkiAOg#p[##O`gukTfBHagL<LHw%q&OV0##F=6/:chIm0@eCP8X]:kFI%hl8hgO@RcBhS-@Qb$%+m=hPDLg*%K8
13165
"eXOONTJlh:.RYF%3'p6sq:UIMA945&^HFS87@$EP2iG<-lCO$%c`uKGD3rC$x0BL8aFn--`ke%#HMP'vh1/R&O_J9'um,.<tx[@%wsJk&bUT2`0uMv7gg
13166
"M7-##.l+Au'A&O:-T72L]P`&=;ctp'XScX*rU.>-XTt,%OVU4)S1+R-#dg0/Nn?Ku1^0f$B*P:Rowwm-`0PKjYDDM'3]d39VZHEl4,.j']Pk-M.h^&:0F
13167
"LuH88Fj-ekm>GA#_>568x6(OFRl-IZp`&b,_P'$M<Jnq79VsJW/mWS*PUiq76;]/NM_>hLbxfc$mj`,O;&%W2m`Zh:/)Uetw:aJ%]K9h:TcF]u_-Sj9,V
13168
"%(?A%R$f<->Zts'^kn=-^@c4%-pY6qI%J%1IGxfLU9CP8cbPlXv);C=b),<2mOvP8up,UVf3839acAWAW-W?#ao/^#%KYo8fRULNd2.>%m]UK:n%r$'sw
13169
"Hg*`+RLgv>=4U8guD$I%D:W>-r5V*%j*W:Kvej.Lp$<M-SGZ':+Q_k+uvOSLiEo(<aD/K<CCc''Lx>'?;++O'>()jLR-^u68PHm8ZFWe+ej8h:9r6L*0/
13170
"a_#Ur7FuA#(tRh#.Y5K+@?3<-8m0$PEn;J:rh6?I6uG<-`wMU'ircp0LaE_OtlMb&1#6T.#FDKu#1Lw%u%+GM+X'e?YLfjM[VO0MbuFp7;>Q&#WIo)0@F
13171
"$/V,;(kXZejWO`<[5?\?ewY(*9=%wDc;,u<'9t3W-(H1th3+G]ucQ]kLs7df($/*JL]@*t7Bu_G3_7mp7<iaQjO@.kLg;x3B0lqp7Hf,^Ze7-##@/c58M
13172
"nKnw'Ho8C=Y>pqB>0ie&jhZ[?iLR@@_AvA-iQC(=ksRZRVp7`.=+NpBC%rh&3]R:8XDmE5^V8O(x«aG/1N$#FX$0V5Y6x'aErI3I$7x%E`v<-BY,)%-?P
13173
"7WhH%o'a<-80g0NBxoO(GH<dM]n.+%q@jH?f.UsJ2Ggs&4<-e47&Kl+f//9@`b+?.TeN_&B8Ss?v;^Trk;f#YvJkl&w$]>-+k?'(<S:68tq*WoDfZu';m
13174
")g:T1=^J$&BRV(-lTmNB6xqB[@0*o.erM*<SWF]u2=st-*(6v>^](H.aREZSi,#1:[IXaZFOm<-ui#qUq2$##Ri;u75OK#(RtaW-K-F`S+cF]uN`-KMQ%
13175
"D?@f&1'BW-)Ju<L25gl8uhVm1hL$###*8###'A3/LkKW+(^rWX?5W_8g)a(m&K8P>#bmmWCMkk&#TR`C,5d>g)F;t,4:@_l8G/5h4vUd%&%950:VXD'QdW
13176
"P?^@Po3$##`MSs?DWBZ/S>+4%>fX,VWv/w'KD`LP5IbH;rTV>n3cEK8U#bX]l-/V+^lj3;vlMb&[5YQ8#pekX9JP3XUC72L„?+Ni&co7ApnO*5NK,((W-
13177
"bIu)'Z,*[>br5fX^:FPAWr-m2KgL<LUN098kTF&#lvo58=/vjDo;.;)Ka*hLR#/k=rKbxuV`>Q_nN6'8uTG&#1T5g)uLv:873UpTLgH+#FgpH'_o1780P
13178
"h4CB/5OvmA&,Q&QbUoi$a_%3M01H)4x7I^&KQVgtFnV+;[Pc>[m4k//,]1?#`VY[Jr*3&&slRfLiVZJ:]?=K3Sw=[$=uRB?3xk48@aeg<Z'<$#4H)6,>e
13179
"V8J'(1)G][68hW$5'q[GC&5j`TE?m'esFGNRM)j,ffZ?-qx8;->g4t*:CIP/[Qap7/9'#(1sao7w-.qNUdkJ)tCF&#B^;xGvn2r9FEPFFFcL@.iFNkTve
13180
"sZ88+dKQ)W6>J%CL<KE>`.d*(B`-n8D9oK<Up]c$X$(,)M8Zt7/[rdkqTgl-0cuGMv'?>-XV1q['-5k'cAZ69e;D_?$ZPP&s^+7])$*$#@QYi9,5P&#9r
13181
".m7jilQO2'0-VWAg<a/"3u.=4L$Y)6k/K:_[3=&jvL<L0C/2'v:^;-DIBW,B4E68:kZ;%?8(Q8BH=kO65BW?xSG&#@uU,DS*,?.+(o(#1vCS8#CHF>TlGU
13182
"$&)WHtPm*5_rO0&e%K&#-30j(E4#'Zb.o/(Tpm$>K'f@[PvFl,hfINTNU6u'0pao7%XUp9]5.>%h`8_=VYbxuel.NTSsJfLacFu3B'lQSu/m6-Oqem8T+
13183
"hv^BFpQj:K'#SJ,sB-'#](j.Lg92rTw-*n%@/;39rrJF,l#qV%OrtBeC6/,;qB3ebNW[?,Hqj2L.lNP&GjUR=lD8QaS3Up&@*9wP?+lo7b?@%'k4`p0Z$
13184
"@-W$U%VEQ/„»#)D<h#`)h0:<Q6909ua+&VU%n2:cG3FJ-%@Bj-DgLr`Hw&HAKjKjseK</xKT*)B,N9X3]krc12t'pgTV(Lv-tL[xg_%=M_q7a^x?7Ubd>
13185
"w6)R89tI#6@s'(6Bf7a&?S=^ZI_kS&ai`&=tE72L_D,;^R)7[$s<Eh#c&)q.MXI%#v9ROa5FZO%sF7q7Nwb&#ptUJ:aqJe$Sl68%.D###EC><?-aF&#RN
13186
"ulp]ovUKW&Y%q]'>$l@-[xfn$7ZTp7mM,G,Ko7a&Gu%G[RMxJs[0MM%wci.LFDK)(<c`Q8N)jEIF*+?P2a8g%)$q]o2aH8C&<SibC/q,(e:v;-b#6[$Nt
13187
"d=j.LQf./Ll33+(;q3L-w=8dX$#WF&uIJ@-bfI>%:_i2B5CsR8&9Z&#=mPEnm0f`<&c)QL5uJ#%u%lJj+D-r;BoF&#4DoS97h5g)E#o:&S4weDF,9^Hoe
13188
"6e%B/:=>)N4xeW.*wft-;$'58-ESqr<b?UI(_%@[P46>#U`'6AQ]m&6/`Z>#S?YY#Vc;r7U2&326d=w&H####?TZ`*4?&.MK?LP8Vxg>$[QXc%QJv92.(
13189
"b0v=Pjer]$gG&JXDf->'StvU7505l9$AFvgYRI^&<^b68?j#q9QX4SM'RO#&sL1IM.rJfLUAj221]d##DW=m83u5;'bYx,*Sl0hL(W;;$doB&O/TQ:(Z^
13190
":k$YUWsbn8ogh6rxZ2Z9]%nd+>V#*8U_72Lh+2Q8Cj0i:6hp&$C/:p(HK>T8Y[gHQ4`4)'$Ab(Nof%V'8hL&#<NEdtg(n'=S1A(Q1/I&4([%dM`,Iu'1:
13191
"tlPN9J*rKaPct&?'uBCem^jn%9_K)<,C5K3s=5g&GmJb*[SYq7K;TRLGCsM-$$;S%:Y@r7AK0pprpL<Lrh,q7e/%KWK:50I^+m'vi`3?%Zp+<-d+$L-Sv
13192
"$3WoJSCLweV[aZ'MQIjO<7;X-X;&+dMLvu#^UsGEC9WEc[X(wI7#2.(F0jV*eZf<-Qv3J-c+J5AlrB#$p(H68LvEA'q3n0#m,[`*8Ft)FcYgEud]CWfm6
13193
":d[/;r_ix=:TF`S5H-b<LI&HY(K=h#)]Lk$K14lVfm:x$H<3^Ql<M`$OhapBnkup'D#L$Pb_`N*g]2e;X/Dtg,bsj&K#2[-:iYr'_wgH)NUIR8a1n#S?Y
13194
"7aQC[K8d-(v6GI$x:T<&'Gp5Uf>@M.*J:;$-rv29'M]8qMv-tLp,'886iaC=Hb*YJoKJ,(j%K=H'K.v9HggqBIiZu'QvBT.#=)0ukruV&.)3=(^1`o*Pj
13195
"u][`*S^43933A4rl][`*O4CgLEl]v$1Q3AeF37dbXk,.)vj#x'd';qgbQR%FW,2(?LO=s%Sc68%NP'##Aotl8x=BE#j1UD([3$M(]UI2LX3RpKN@;/#f'
13196
"LwQ'(TTB9.xH'>#MJ+gLq9-##@HuZPN0]u:h7.T..G:;$/Usj(T7`Q8tT72LnYl<-qx8;-HV7Q-&Xdx%1a,hC=0u+HlsV>nuIQL-5<N?)NBS)QN*_I,?&
13197
":^#M*Q+[T.Xri.LYS3v%fF`68h;b-X[/En'CR.q7E)p'/kle2HM,u;^%OKC-N+Ll%F9CF<Nf'^#t2L,;27W:0O@6##U6W7:$rJfLWHj$#)woqBefIZ.PK
13198
"_>@kXQtMacfD.m-VAb8;IReM3$wf0"hra*so568'Ip&vRs849'MRYSp%:t:h5qSgwpEr$B>Q,;s(C#$)`svQuF$##-D,##,g68@2[T;.XSdN9Qe)rpt._
13199
"hd+<-j'Ai*x&&HMkT]C'OSl##5RG[JXaHN;d'uA#x._U;.`PU@(Z3dt4r152@:v,'R.Sj'w#0<-;kPI)FfJ&#AYJ&#//)>-k=m=*XnK$>=)72L]0I%>.G
13200
"^V'9;jY@;)br#q^YQpx:X#Te$Z^'=-=bGhLf:D6&bNwZ9-ZD#n^9HhLMr5G;']d&6'wYmTFmL<LD)F^%[tC'8;+9E#C$g%#5Y>q9wI>P(9mI[>kC-ekLC
13201
```

```
"+A4[7xks.LrNk0&E)wILYF@2L'0Nb$+pv<(2.768/FrY&h$^3i&@+G%JT'<-,v`3;_)I9M^AE]CN?Cl2AZg+%4iTpT3<n-&%H%b<FDj2M<hH=&Eh<2Len
"9s<L<NFSo)B?+<-(GxsF,^-Eh@$4dXhN$+#rxK8'je'D7k`e;)2pYwPA'_p9&@^18ml1^[@g4t*[JOa*[=Qp7(qJ_oOL^('7fB&Hq-:sf,sNj8xq^>$U4
"CdQ*:Ir<($u&)#(&?L9Rg3H)4fiEp^iI9O8KnTj,]H?D*r7'M;PwZ9K0E^k&-cpI;.p/6_vwoFMV<->#%Xi.LxVnrU(4&8/P+:hLSKj$#U%]49t'I:rgM
"hU%<-SRF`Tt:542R_VV$p@[p8DV[A,?1839FWdF<TddF<9Ah-6&9tWoDlh]&1SpGMq>Ti1O*H&#(AL8[_P%.M>v^-))qOT*F5Cq0`Ye%+$B6i:7@0IX<N
"8kD2)2fU/M#$e.)T4,_=8hLim[&);?UkK'-x?'(:siIfL<$pFM`i<?%W(mGDHM%>iWP,##P`%/L<eXi:@Z9C.7o=@(pXdAO/NLQ8lPl+HPOQa8wD8=^GlP
"S(qw%sf/@%#B6;/U7K]uZbi^Oc^2n<bhPmUkMw>%t<)'mEVE”n`WnJra$^TKvX5B>;_aSEK',(hwa0:i4G?.Bci.(X[?b*($,=-n<.Q%`(X=?+@Am*Js0&
"0?t/'_U59@]ddF<#LdF<eWdF<OuN/45rY<-L@&#+fm>69=Lb,OcZV/);TTm8VI;?%OtJ<(b4mq7M6:u?KRdF<gR@2L=FNU-<b[(9c/ML3m;Z[$oF3g)GAW
"+fsd;l#SafT/f*W]0=O'$(Tb<[)*@e775R-:Yob%g*>l*:xP?Yb.5)%w_I?7uk5JC+FS(m#i'k.'a0i)9<7b'fs'59hq$*5Uhv##pi^8+hIEBF`nvo`;'
"M=SO*rfO`+qC`W-On.=AJ56»i2@2LH6A:&5q`?9I3@@'04&p2/LVa*T-4<-i3;M9UvZd+N7>b*eIwg:CC)c<>nO&#<IGe;__.thjZl<%w(Wk2xmp4Q@I#
"?6*C()dOp7:WL,b&3Rg/.cmM9&r^>$(>.Z-I&J(Q0Hd5Q%7Co-b`-c<N(6r@ip+AurK<m86QIth*#v;-OBqi+L7wDE-Ir8K['m+DDSLwK&/.?-V%U_%3:
"Yq[@>P)hI;*_F]u`Rb[.j8_Q/<&>uu+VsH$sM9TA%?)(vmJ80),P7E>)tjD%2L=-t#fK[%`v=Q8<FfNkgg^oIbah*#8/Qt$F&:K*-(N/'+1vMB,u()-a.
">gXm8YB`1d@K#n]76-a$U,mF<fX]idqd)<3,]J7JmW4`6]uks=4-72L(jEk+:bJ0M^q-8Dm_Z?0olP1C9Sa&H[d&c$ooQUj]Exd*3ZM@-WGW2%s',B-_M
"[%$Z$uF6pA6Ki2O5:8w*vP1<-1`[G,)-m#>0`P&#eb#.3i)rtB61(o'$?X3B</R90;eZ]%Ncq;-Tl]#F>2Qft^ae_5tKL9MUe9b*sLEQ95C&`=G?@Mj=w
"wOf7&]1i'S01B+Ev/Nac#9S;=;YQpg_6U`*kVY39xK,[/6Aj7:'1Bm-_1EYfa1+o&o4hp7KN_Q(OlIo@S%;jVdn0'1<Vc52=u`3^o-n1'g4v58Hj&6_t7
"Y(,p'H9rIVY-b,'%bCPF7.J<Up^,(dU1VY*5#WkTU>h19w,WQhLI)3S#f$2(eb,jr*b;3Vw]*7NH%$c4Vs,eD9>XW8?N]o+(*pgC%/72LV-u<Hp,3@e^9
"MlvAF_jCK*.O-^(63adMT->W%iewS8W6m2rtCpo'RS1R84=@paTKt)>=%&1[)*vp'u+x,VrwN;&]kuO9JDbg=pO$J*.jVe;u'm0dr9l,<*wMK*Oe=g8lV_
"i]lR8qQ2oA8wcRCZ^7w/Njh;?.stX?Q1>S1q4Bn$)K1<-rGdO'$Wr.Lc.CG)$/*JL4tNR/,SVO3,aUw'DJN:)Ss;wGn9A32ijw%FL+Z0Fn.U9;reSq)bm
"1*c-(aY168o<`JsSbk-,1N;$>0:OUas(3:8Z972LSfF8eb=c-;>SPw7.6hn3m`9^Xkn(r.qS[0;T%&Qc=+STRxX'q1BNk3&*eu2;&8q$&x>Q#Q7^Tf+6<
"iDDG)g,r%+?,$@?uou5tSe2aN_AQU*<h`e-GI7)?OK2A.d7_c)?wQ5AS@DL3r#7fSkgl6-++D:'A,uq7SvlB$pcpH'q3n0#_%dY#xCpr-l<F0NR@-##FEV
"URQ##V^Fv-XFbGM7Fl(N<3DhLGF%q.1rC$#:T__&Pi68%0xi_&[qFJ(77j_&JWoF.V735&T,[R*:xFR*K5»#`bW-?4Ne_&6Ne_&6Ne_&n`kr-#GJcM6X;
";u.T%fAr%4tJ8&><1=GHZ_+m9/#H1F^R#SC#*N=BA9(D?v[UiFY»^8p,KKF.W]L29uLkLlu/+4T<XoIB&hx=T1PcDaB&;HH+-AFr?(m9HZV)FKS8JCw;SI
"w$)F./^n3+rlo+DB;5sIYGNk+i1t-69Jg--0pao7Sm#K)pdHW&;LuDNH@H>#/X-TI(;P>#,Gc>#0Su>#4`1?#8lC?#<xU?#@.i?#D:%@#HF7@#LRI@#P_
"d<F&#*;G##.GY##2Sl##6`($#:l:$#>xL$#B.`$#F:r$#JF.%#NR@%#R_R%#Vke%#Zww%#_-4&#3^Rh%Sflr-k'MS.o?.5/sWel/wpEM0%3'/1)K^f1-d
"A1OY5EI0;6Ibgr6M$HS7Q<)58C5w,;WoA*#[%T*#`1g*#d=#+#hI5+#lUG+#pbY+#tnl+#x$),#&1;,#*=M,#.I`,#2Ur,#6b.-#;w[H#iQtA#m^0B#qj
"/QHC#3^ZC#7jmC#;v)D#?,<D#C8ND#GDaD#KPsD#O]/E#g1A5#KA*1#gC17#MGd;#8(02#L-d3#rWM4#Hga1#,<w0#T.j<#O#'2#CYN1#qa^:#_4m3#o@
"m3B2#SB[8#Q0@8#i[*9#iOn8#1Nm;#^sN9#qh<9#:=x-#P;K2#$%X9#bC+.#Rg;<#mN=.#MTF.#RZO.#2?)4#Y#(/#[)1/#b;L/#dAU/#0Sv;#lY$0#n`
"TmD<#%JSMFove:CTBEXI:<eh2g)B,3h2^G3i;#d3jD>)4kMYD4lVu`4m`:&5niUA5@(A5BA1]PBB:xlBCC=2CDLXMCEUtiCf&0g2'tN?PGT4CPGT4CPGT4
"GT4CPGT4CPGT4CPGT4CPGT4CPGT4CP-qekC`.9kEg^+F$kwViFJTB&5KTB&5KTB&5KTB&5KTB&5KTB&5KTB&5KTB&5KTB&5KTB&5KTB&5KTB&5KTB&5KT
  "O?;xpO?;xpO?;xpO?;xpO?;xpO?;xpO?;xpO?;xpO?;xpO?;xpO?;xpO?;xpO?;xp;7q-#lLYI:xvD=#";

#endif /* NK_INCLUDE_DEFAULT_FONT */

#define NK_CURSOR_DATA_W 90
#define NK_CURSOR_DATA_H 27
NK_GLOBAL const char nk_custom_cursor_data[NK_CURSOR_DATA_W * NK_CURSOR_DATA_H + 1] =
{
    "..-         -XXXXXXX-    X    -           X           -XXXXXXX          -          XXXXXXX"
    "..-         -X.....X-   X.X   -          X.X          -X.....X          -          X.....X"
    "---         -XXX.XXX-  X...X  -         X...X         -X....X           -           X....X"
    "X           -  X.X   - X.....X -        X.....X        -X...X            -            X...X"
    "XX          -  X.X   -X.......X-       X.......X       -X..X.X           -           X.X..X"
    "X.X         -  X.X   -XXXX.XXXX-       XXXX.XXXX       -X.X X.X          -          X.X X.X"
    "X..X        -  X.X   -   X.X   -          X.X          -XX   X.X         -         X.X   XX"
    "X...X       -  X.X   -   X.X   -    XX    X.X    XX    -      X.X        -        X.X      "
    "X....X      -  X.X   -   X.X   -   X.X    X.X    X.X   -       X.X       -       X.X       "
    "X.....X     -  X.X   -   X.X   -  X..X    X.X    X..X  -        X.X      -      X.X        "
    "X......X    -  X.X   -   X.X   - X...XXXXXX.XXXXXX...X -         X.X   XX-XX   X.X         "
    "X.......X   -  X.X   -   X.X   -X.....................X-          X.X X.X-X.X X.X          "
    "X........X  -  X.X   -   X.X   - X...XXXXXX.XXXXXX...X -           X.X..X-X..X.X           "
    "X.........X -XXX.XXX-   X.X   -  X..X    X.X    X..X  -            X...X-X...X            "
    "X..........X-X.....X-   X.X   -   X.X    X.X    X.X   -           X....X-X....X           "
    "X......XXXXX-XXXXXXX-   X.X   -    XX    X.X    XX    -          X.....X-X.....X          "
    "X...X..X    ---------   X.X   -          X.X          -          XXXXXXX-XXXXXXX          "
    "X..X X..X   -       -XXXX.XXXX-       XXXX.XXXX       ------------------------------------"
    "X.X  X..X   -       -X.......X-       X.......X       -    XX           XX    -           "
    "XX    X..X  -       - X.....X -        X.....X        -   X.X           X.X   -           "
    "      X..X          -  X...X  -         X...X         -  X..X           X..X  -           "
    "       XX           -   X.X   -          X.X          - X...XXXXXXXXXXXXX...X -           "
    "------------        -    X    -           X           -X.....................X-           "
    "                    ----------------------------------- X...XXXXXXXXXXXXX...X -           "
```

```
13261     "                                                  -  X..X          X..X  -          "
13262     "                                                  -   X.X          X.X   -          "
13263     "                                                  -    XX          XX    -          "
13264 };
13265
13266 #ifdef __clang__
13267 #pragma clang diagnostic pop
13268 #elif defined(__GNUC__) || defined(__GNUG__)
13269 #pragma GCC diagnostic pop
13270 #endif
13271
13272 NK_GLOBAL unsigned char *nk__barrier;
13273 NK_GLOBAL unsigned char *nk__barrier2;
13274 NK_GLOBAL unsigned char *nk__barrier3;
13275 NK_GLOBAL unsigned char *nk__barrier4;
13276 NK_GLOBAL unsigned char *nk__dout;
13277
13278 NK_INTERN unsigned int
13279 nk_decompress_length(unsigned char *input)
13280 {
13281     return (unsigned int)((input[8] « 24) + (input[9] « 16) + (input[10] « 8) + input[11]);
13282 }
13283 NK_INTERN void
13284 nk__match(unsigned char *data, unsigned int length)
13285 {
13286     /* INVERSE of memmove... write each byte before copying the next...*/
13287     NK_ASSERT (nk__dout + length <= nk__barrier);
13288     if (nk__dout + length > nk__barrier) { nk__dout += length; return; }
13289     if (data < nk__barrier4) { nk__dout = nk__barrier+1; return; }
13290     while (length--) *nk__dout++ = *data++;
13291 }
13292 NK_INTERN void
13293 nk__lit(unsigned char *data, unsigned int length)
13294 {
13295     NK_ASSERT (nk__dout + length <= nk__barrier);
13296     if (nk__dout + length > nk__barrier) { nk__dout += length; return; }
13297     if (data < nk__barrier2) { nk__dout = nk__barrier+1; return; }
13298     NK_MEMCPY(nk__dout, data, length);
13299     nk__dout += length;
13300 }
13301 NK_INTERN unsigned char*
13302 nk_decompress_token(unsigned char *i)
13303 {
13304     #define nk__in2(x)   ((i[x] « 8) + i[(x)+1])
13305     #define nk__in3(x)   ((i[x] « 16) + nk__in2((x)+1))
13306     #define nk__in4(x)   ((i[x] « 24) + nk__in3((x)+1))
13307
13308     if (*i >= 0x20) { /* use fewer if's for cases that expand small */
13309         if (*i >= 0x80)       nk__match(nk__dout-i[1]-1, (unsigned int)i[0] - 0x80 + 1), i += 2;
13310         else if (*i >= 0x40)  nk__match(nk__dout-(nk__in2(0) - 0x4000 + 1), (unsigned int)i[2]+1), i
 += 3;
13311         else /* *i >= 0x20 */ nk__lit(i+1, (unsigned int)i[0] - 0x20 + 1), i += 1 + (i[0] - 0x20 + 1);
13312     } else { /* more ifs for cases that expand large, since overhead is amortized */
13313         if (*i >= 0x18)       nk__match(nk__dout-(unsigned int)(nk__in3(0) - 0x180000 + 1), (unsigned
 int)i[3]+1), i += 4;
13314         else if (*i >= 0x10)  nk__match(nk__dout-(unsigned int)(nk__in3(0) - 0x100000 + 1), (unsigned
 int)nk__in2(3)+1), i += 5;
13315         else if (*i >= 0x08)  nk__lit(i+2, (unsigned int)nk__in2(0) - 0x0800 + 1), i += 2 +
 (nk__in2(0) - 0x0800 + 1);
13316         else if (*i == 0x07)  nk__lit(i+3, (unsigned int)nk__in2(1) + 1), i += 3 + (nk__in2(1) + 1);
13317         else if (*i == 0x06)  nk__match(nk__dout-(unsigned int)(nk__in3(1)+1), i[4]+1u), i += 5;
13318         else if (*i == 0x04)  nk__match(nk__dout-(unsigned int)(nk__in3(1)+1), (unsigned
 int)nk__in2(4)+1u), i += 6;
13319     }
13320     return i;
13321 }
13322 NK_INTERN unsigned int
13323 nk_adler32(unsigned int adler32, unsigned char *buffer, unsigned int buflen)
13324 {
13325     const unsigned long ADLER_MOD = 65521;
13326     unsigned long s1 = adler32 & 0xffff, s2 = adler32 » 16;
13327     unsigned long blocklen, i;
13328
13329     blocklen = buflen % 5552;
13330     while (buflen) {
13331         for (i=0; i + 7 < blocklen; i += 8) {
13332             s1 += buffer[0]; s2 += s1;
13333             s1 += buffer[1]; s2 += s1;
13334             s1 += buffer[2]; s2 += s1;
13335             s1 += buffer[3]; s2 += s1;
13336             s1 += buffer[4]; s2 += s1;
13337             s1 += buffer[5]; s2 += s1;
13338             s1 += buffer[6]; s2 += s1;
13339             s1 += buffer[7]; s2 += s1;
13340             buffer += 8;
13341         }
13342         for (; i < blocklen; ++i) {
```

```
13343             s1 += *buffer++; s2 += s1;
13344         }
13345
13346         s1 %= ADLER_MOD; s2 %= ADLER_MOD;
13347         buflen -= (unsigned int)blocklen;
13348         blocklen = 5552;
13349     }
13350     return (unsigned int)(s2 « 16) + (unsigned int)s1;
13351 }
13352 NK_INTERN unsigned int
13353 nk_decompress(unsigned char *output, unsigned char *i, unsigned int length)
13354 {
13355     unsigned int olen;
13356     if (nk__in4(0) != 0x57bC0000) return 0;
13357     if (nk__in4(4) != 0)          return 0; /* error! stream is > 4GB */
13358     olen = nk_decompress_length(i);
13359     nk__barrier2 = i;
13360     nk__barrier3 = i+length;
13361     nk__barrier = output + olen;
13362     nk__barrier4 = output;
13363     i += 16;
13364
13365     nk__dout = output;
13366     for (;;) {
13367         unsigned char *old_i = i;
13368         i = nk_decompress_token(i);
13369         if (i == old_i) {
13370             if (*i == 0x05 && i[1] == 0xfa) {
13371                 NK_ASSERT(nk__dout == output + olen);
13372                 if (nk__dout != output + olen) return 0;
13373                 if (nk_adler32(1, output, olen) != (unsigned int) nk__in4(2))
13374                     return 0;
13375                 return olen;
13376             } else {
13377                 NK_ASSERT(0); /* NOTREACHED */
13378                 return 0;
13379             }
13380         }
13381         NK_ASSERT(nk__dout <= output + olen);
13382         if (nk__dout > output + olen)
13383             return 0;
13384     }
13385 }
13386 NK_INTERN unsigned int
13387 nk_decode_85_byte(char c)
13388 {
13389     return (unsigned int)((c >= '\\') ? c-36 : c-35);
13390 }
13391 NK_INTERN void
13392 nk_decode_85(unsigned char* dst, const unsigned char* src)
13393 {
13394     while (*src)
13395     {
13396         unsigned int tmp =
13397             nk_decode_85_byte((char)src[0]) +
13398             85 * (nk_decode_85_byte((char)src[1]) +
13399             85 * (nk_decode_85_byte((char)src[2]) +
13400             85 * (nk_decode_85_byte((char)src[3]) +
13401             85 * nk_decode_85_byte((char)src[4]))));
13402
13403         /* we can't assume little-endianess. */
13404         dst[0] = (unsigned char)((tmp » 0) & 0xFF);
13405         dst[1] = (unsigned char)((tmp » 8) & 0xFF);
13406         dst[2] = (unsigned char)((tmp » 16) & 0xFF);
13407         dst[3] = (unsigned char)((tmp » 24) & 0xFF);
13408
13409         src += 5;
13410         dst += 4;
13411     }
13412 }
13413
13414 /* ---------------------------------------------------------------
13415  *
13416  *                          FONT ATLAS
13417  *
13418  * ---------------------------------------------------------------*/
13419 NK_API struct nk_font_config
13420 nk_font_config(float pixel_height)
13421 {
13422     struct nk_font_config cfg;
13423     nk_zero_struct(cfg);
13424     cfg.ttf_blob = 0;
13425     cfg.ttf_size = 0;
13426     cfg.ttf_data_owned_by_atlas = 0;
13427     cfg.size = pixel_height;
13428     cfg.oversample_h = 3;
13429     cfg.oversample_v = 1;
```

```
13430     cfg.pixel_snap = 0;
13431     cfg.coord_type = NK_COORD_UV;
13432     cfg.spacing = nk_vec2(0,0);
13433     cfg.range = nk_font_default_glyph_ranges();
13434     cfg.merge_mode = 0;
13435     cfg.fallback_glyph = '?';
13436     cfg.font = 0;
13437     cfg.n = 0;
13438     return cfg;
13439 }
13440 #ifdef NK_INCLUDE_DEFAULT_ALLOCATOR
13441 NK_API void
13442 nk_font_atlas_init_default(struct nk_font_atlas *atlas)
13443 {
13444     NK_ASSERT(atlas);
13445     if (!atlas) return;
13446     nk_zero_struct(*atlas);
13447     atlas->temporary.userdata.ptr = 0;
13448     atlas->temporary.alloc = nk_malloc;
13449     atlas->temporary.free = nk_mfree;
13450     atlas->permanent.userdata.ptr = 0;
13451     atlas->permanent.alloc = nk_malloc;
13452     atlas->permanent.free = nk_mfree;
13453 }
13454 #endif
13455 NK_API void
13456 nk_font_atlas_init(struct nk_font_atlas *atlas, struct nk_allocator *alloc)
13457 {
13458     NK_ASSERT(atlas);
13459     NK_ASSERT(alloc);
13460     if (!atlas || !alloc) return;
13461     nk_zero_struct(*atlas);
13462     atlas->permanent = *alloc;
13463     atlas->temporary = *alloc;
13464 }
13465 NK_API void
13466 nk_font_atlas_init_custom(struct nk_font_atlas *atlas,
13467     struct nk_allocator *permanent, struct nk_allocator *temporary)
13468 {
13469     NK_ASSERT(atlas);
13470     NK_ASSERT(permanent);
13471     NK_ASSERT(temporary);
13472     if (!atlas || !permanent || !temporary) return;
13473     nk_zero_struct(*atlas);
13474     atlas->permanent = *permanent;
13475     atlas->temporary = *temporary;
13476 }
13477 NK_API void
13478 nk_font_atlas_begin(struct nk_font_atlas *atlas)
13479 {
13480     NK_ASSERT(atlas);
13481     NK_ASSERT(atlas->temporary.alloc && atlas->temporary.free);
13482     NK_ASSERT(atlas->permanent.alloc && atlas->permanent.free);
13483     if (!atlas || !atlas->permanent.alloc || !atlas->permanent.free ||
13484         !atlas->temporary.alloc || !atlas->temporary.free) return;
13485     if (atlas->glyphs) {
13486         atlas->permanent.free(atlas->permanent.userdata, atlas->glyphs);
13487         atlas->glyphs = 0;
13488     }
13489     if (atlas->pixel) {
13490         atlas->permanent.free(atlas->permanent.userdata, atlas->pixel);
13491         atlas->pixel = 0;
13492     }
13493 }
13494 NK_API struct nk_font*
13495 nk_font_atlas_add(struct nk_font_atlas *atlas, const struct nk_font_config *config)
13496 {
13497     struct nk_font *font = 0;
13498     struct nk_font_config *cfg;
13499
13500     NK_ASSERT(atlas);
13501     NK_ASSERT(atlas->permanent.alloc);
13502     NK_ASSERT(atlas->permanent.free);
13503     NK_ASSERT(atlas->temporary.alloc);
13504     NK_ASSERT(atlas->temporary.free);
13505
13506     NK_ASSERT(config);
13507     NK_ASSERT(config->ttf_blob);
13508     NK_ASSERT(config->ttf_size);
13509     NK_ASSERT(config->size > 0.0f);
13510
13511     if (!atlas || !config || !config->ttf_blob || !config->ttf_size || config->size <= 0.0f||
13512         !atlas->permanent.alloc || !atlas->permanent.free ||
13513         !atlas->temporary.alloc || !atlas->temporary.free)
13514         return 0;
13515
13516     /* allocate font config  */
```

```
13517     cfg = (struct nk_font_config*)
13518         atlas->permanent.alloc(atlas->permanent.userdata,0, sizeof(struct nk_font_config));
13519     NK_MEMCPY(cfg, config, sizeof(*config));
13520     cfg->n = cfg;
13521     cfg->p = cfg;
13522
13523     if (!config->merge_mode) {
13524         /* insert font config into list */
13525         if (!atlas->config) {
13526             atlas->config = cfg;
13527             cfg->next = 0;
13528         } else {
13529             struct nk_font_config *i = atlas->config;
13530             while (i->next) i = i->next;
13531             i->next = cfg;
13532             cfg->next = 0;
13533         }
13534         /* allocate new font */
13535         font = (struct nk_font*)
13536             atlas->permanent.alloc(atlas->permanent.userdata,0, sizeof(struct nk_font));
13537         NK_ASSERT(font);
13538         nk_zero(font, sizeof(*font));
13539         if (!font) return 0;
13540         font->config = cfg;
13541
13542         /* insert font into list */
13543         if (!atlas->fonts) {
13544             atlas->fonts = font;
13545             font->next = 0;
13546         } else {
13547             struct nk_font *i = atlas->fonts;
13548             while (i->next) i = i->next;
13549             i->next = font;
13550             font->next = 0;
13551         }
13552         cfg->font = &font->info;
13553     } else {
13554         /* extend previously added font */
13555         struct nk_font *f = 0;
13556         struct nk_font_config *c = 0;
13557         NK_ASSERT(atlas->font_num);
13558         f = atlas->fonts;
13559         c = f->config;
13560         cfg->font = &f->info;
13561
13562         cfg->n = c;
13563         cfg->p = c->p;
13564         c->p->n = cfg;
13565         c->p = cfg;
13566     }
13567     /* create own copy of .TTF font blob */
13568     if (!config->ttf_data_owned_by_atlas) {
13569         cfg->ttf_blob = atlas->permanent.alloc(atlas->permanent.userdata,0, cfg->ttf_size);
13570         NK_ASSERT(cfg->ttf_blob);
13571         if (!cfg->ttf_blob) {
13572             atlas->font_num++;
13573             return 0;
13574         }
13575         NK_MEMCPY(cfg->ttf_blob, config->ttf_blob, cfg->ttf_size);
13576         cfg->ttf_data_owned_by_atlas = 1;
13577     }
13578     atlas->font_num++;
13579     return font;
13580 }
13581 NK_API struct nk_font*
13582 nk_font_atlas_add_from_memory(struct nk_font_atlas *atlas, void *memory,
13583     nk_size size, float height, const struct nk_font_config *config)
13584 {
13585     struct nk_font_config cfg;
13586     NK_ASSERT(memory);
13587     NK_ASSERT(size);
13588
13589     NK_ASSERT(atlas);
13590     NK_ASSERT(atlas->temporary.alloc);
13591     NK_ASSERT(atlas->temporary.free);
13592     NK_ASSERT(atlas->permanent.alloc);
13593     NK_ASSERT(atlas->permanent.free);
13594     if (!atlas || !atlas->temporary.alloc || !atlas->temporary.free || !memory || !size ||
13595         !atlas->permanent.alloc || !atlas->permanent.free)
13596         return 0;
13597
13598     cfg = (config) ? *config: nk_font_config(height);
13599     cfg.ttf_blob = memory;
13600     cfg.ttf_size = size;
13601     cfg.size = height;
13602     cfg.ttf_data_owned_by_atlas = 0;
13603     return nk_font_atlas_add(atlas, &cfg);
```

```
13604 }
13605 #ifdef NK_INCLUDE_STANDARD_IO
13606 NK_API struct nk_font*
13607 nk_font_atlas_add_from_file(struct nk_font_atlas *atlas, const char *file_path,
13608     float height, const struct nk_font_config *config)
13609 {
13610     nk_size size;
13611     char *memory;
13612     struct nk_font_config cfg;
13613
13614     NK_ASSERT(atlas);
13615     NK_ASSERT(atlas->temporary.alloc);
13616     NK_ASSERT(atlas->temporary.free);
13617     NK_ASSERT(atlas->permanent.alloc);
13618     NK_ASSERT(atlas->permanent.free);
13619
13620     if (!atlas || !file_path) return 0;
13621     memory = nk_file_load(file_path, &size, &atlas->permanent);
13622     if (!memory) return 0;
13623
13624     cfg = (config) ? *config: nk_font_config(height);
13625     cfg.ttf_blob = memory;
13626     cfg.ttf_size = size;
13627     cfg.size = height;
13628     cfg.ttf_data_owned_by_atlas = 1;
13629     return nk_font_atlas_add(atlas, &cfg);
13630 }
13631 #endif
13632 NK_API struct nk_font*
13633 nk_font_atlas_add_compressed(struct nk_font_atlas *atlas,
13634     void *compressed_data, nk_size compressed_size, float height,
13635     const struct nk_font_config *config)
13636 {
13637     unsigned int decompressed_size;
13638     void *decompressed_data;
13639     struct nk_font_config cfg;
13640
13641     NK_ASSERT(atlas);
13642     NK_ASSERT(atlas->temporary.alloc);
13643     NK_ASSERT(atlas->temporary.free);
13644     NK_ASSERT(atlas->permanent.alloc);
13645     NK_ASSERT(atlas->permanent.free);
13646
13647     NK_ASSERT(compressed_data);
13648     NK_ASSERT(compressed_size);
13649     if (!atlas || !compressed_data || !atlas->temporary.alloc || !atlas->temporary.free ||
13650         !atlas->permanent.alloc || !atlas->permanent.free)
13651         return 0;
13652
13653     decompressed_size = nk_decompress_length((unsigned char*)compressed_data);
13654     decompressed_data = atlas->permanent.alloc(atlas->permanent.userdata,0,decompressed_size);
13655     NK_ASSERT(decompressed_data);
13656     if (!decompressed_data) return 0;
13657     nk_decompress((unsigned char*)decompressed_data, (unsigned char*)compressed_data,
13658         (unsigned int)compressed_size);
13659
13660     cfg = (config) ? *config: nk_font_config(height);
13661     cfg.ttf_blob = decompressed_data;
13662     cfg.ttf_size = decompressed_size;
13663     cfg.size = height;
13664     cfg.ttf_data_owned_by_atlas = 1;
13665     return nk_font_atlas_add(atlas, &cfg);
13666 }
13667 NK_API struct nk_font*
13668 nk_font_atlas_add_compressed_base85(struct nk_font_atlas *atlas,
13669     const char *data_base85, float height, const struct nk_font_config *config)
13670 {
13671     int compressed_size;
13672     void *compressed_data;
13673     struct nk_font *font;
13674
13675     NK_ASSERT(atlas);
13676     NK_ASSERT(atlas->temporary.alloc);
13677     NK_ASSERT(atlas->temporary.free);
13678     NK_ASSERT(atlas->permanent.alloc);
13679     NK_ASSERT(atlas->permanent.free);
13680
13681     NK_ASSERT(data_base85);
13682     if (!atlas || !data_base85 || !atlas->temporary.alloc || !atlas->temporary.free ||
13683         !atlas->permanent.alloc || !atlas->permanent.free)
13684         return 0;
13685
13686     compressed_size = (((int)nk_strlen(data_base85) + 4) / 5) * 4;
13687     compressed_data = atlas->temporary.alloc(atlas->temporary.userdata,0, (nk_size)compressed_size);
13688     NK_ASSERT(compressed_data);
13689     if (!compressed_data) return 0;
13690     nk_decode_85((unsigned char*)compressed_data, (const unsigned char*)data_base85);
```

```
13691     font = nk_font_atlas_add_compressed(atlas, compressed_data,
13692                     (nk_size)compressed_size, height, config);
13693     atlas->temporary.free(atlas->temporary.userdata, compressed_data);
13694     return font;
13695 }
13696
13697 #ifdef NK_INCLUDE_DEFAULT_FONT
13698 NK_API struct nk_font*
13699 nk_font_atlas_add_default(struct nk_font_atlas *atlas,
13700     float pixel_height, const struct nk_font_config *config)
13701 {
13702     NK_ASSERT(atlas);
13703     NK_ASSERT(atlas->temporary.alloc);
13704     NK_ASSERT(atlas->temporary.free);
13705     NK_ASSERT(atlas->permanent.alloc);
13706     NK_ASSERT(atlas->permanent.free);
13707     return nk_font_atlas_add_compressed_base85(atlas,
13708         nk_proggy_clean_ttf_compressed_data_base85, pixel_height, config);
13709 }
13710 #endif
13711 NK_API const void*
13712 nk_font_atlas_bake(struct nk_font_atlas *atlas, int *width, int *height,
13713     enum nk_font_atlas_format fmt)
13714 {
13715     int i = 0;
13716     void *tmp = 0;
13717     nk_size tmp_size, img_size;
13718     struct nk_font *font_iter;
13719     struct nk_font_baker *baker;
13720
13721     NK_ASSERT(atlas);
13722     NK_ASSERT(atlas->temporary.alloc);
13723     NK_ASSERT(atlas->temporary.free);
13724     NK_ASSERT(atlas->permanent.alloc);
13725     NK_ASSERT(atlas->permanent.free);
13726
13727     NK_ASSERT(width);
13728     NK_ASSERT(height);
13729     if (!atlas || !width || !height ||
13730         !atlas->temporary.alloc || !atlas->temporary.free ||
13731         !atlas->permanent.alloc || !atlas->permanent.free)
13732         return 0;
13733
13734 #ifdef NK_INCLUDE_DEFAULT_FONT
13735     /* no font added so just use default font */
13736     if (!atlas->font_num)
13737         atlas->default_font = nk_font_atlas_add_default(atlas, 13.0f, 0);
13738 #endif
13739     NK_ASSERT(atlas->font_num);
13740     if (!atlas->font_num) return 0;
13741
13742     /* allocate temporary baker memory required for the baking process */
13743     nk_font_baker_memory(&tmp_size, &atlas->glyph_count, atlas->config, atlas->font_num);
13744     tmp = atlas->temporary.alloc(atlas->temporary.userdata,0, tmp_size);
13745     NK_ASSERT(tmp);
13746     if (!tmp) goto failed;
13747
13748     /* allocate glyph memory for all fonts */
13749     baker = nk_font_baker(tmp, atlas->glyph_count, atlas->font_num, &atlas->temporary);
13750     atlas->glyphs = (struct nk_font_glyph*)atlas->permanent.alloc(
13751         atlas->permanent.userdata,0, sizeof(struct nk_font_glyph)*(nk_size)atlas->glyph_count);
13752     NK_ASSERT(atlas->glyphs);
13753     if (!atlas->glyphs)
13754         goto failed;
13755
13756     /* pack all glyphs into a tight fit space */
13757     atlas->custom.w = (NK_CURSOR_DATA_W*2)+1;
13758     atlas->custom.h = NK_CURSOR_DATA_H + 1;
13759     if (!nk_font_bake_pack(baker, &img_size, width, height, &atlas->custom,
13760         atlas->config, atlas->font_num, &atlas->temporary))
13761         goto failed;
13762
13763     /* allocate memory for the baked image font atlas */
13764     atlas->pixel = atlas->temporary.alloc(atlas->temporary.userdata,0, img_size);
13765     NK_ASSERT(atlas->pixel);
13766     if (!atlas->pixel)
13767         goto failed;
13768
13769     /* bake glyphs and custom white pixel into image */
13770     nk_font_bake(baker, atlas->pixel, *width, *height,
13771         atlas->glyphs, atlas->glyph_count, atlas->font_num);
13772     nk_font_bake_custom_data(atlas->pixel, *width, *height, atlas->custom,
13773             nk_custom_cursor_data, NK_CURSOR_DATA_W, NK_CURSOR_DATA_H, '.', 'X');
13774
13775     if (fmt == NK_FONT_ATLAS_RGBA32) {
13776         /* convert alpha8 image into rgba32 image */
13777         void *img_rgba = atlas->temporary.alloc(atlas->temporary.userdata,0,
```

```
13778                               (nk_size)(*width * *height * 4));
13779         NK_ASSERT(img_rgba);
13780         if (!img_rgba) goto failed;
13781         nk_font_bake_convert(img_rgba, *width, *height, atlas->pixel);
13782         atlas->temporary.free(atlas->temporary.userdata, atlas->pixel);
13783         atlas->pixel = img_rgba;
13784     }
13785     atlas->tex_width = *width;
13786     atlas->tex_height = *height;
13787
13788     /* initialize each font */
13789     for (font_iter = atlas->fonts; font_iter; font_iter = font_iter->next) {
13790         struct nk_font *font = font_iter;
13791         struct nk_font_config *config = font->config;
13792         nk_font_init(font, config->size, config->fallback_glyph, atlas->glyphs,
13793             config->font, nk_handle_ptr(0));
13794     }
13795
13796     /* initialize each cursor */
13797     {NK_STORAGE const struct nk_vec2 nk_cursor_data[NK_CURSOR_COUNT][3] = {
13798         /* Pos      Size        Offset */
13799         {{ 0, 3},   {12,19},    { 0, 0}},
13800         {{13, 0},   { 7,16},    { 4, 8}},
13801         {{31, 0},   {23,23},    {11,11}},
13802         {{21, 0},   { 9, 23},   { 5,11}},
13803         {{55,18},   {23, 9},    {11, 5}},
13804         {{73, 0},   {17,17},    { 9, 9}},
13805         {{55, 0},   {17,17},    { 9, 9}}
13806     };
13807     for (i = 0; i < NK_CURSOR_COUNT; ++i) {
13808         struct nk_cursor *cursor = &atlas->cursors[i];
13809         cursor->img.w = (unsigned short)*width;
13810         cursor->img.h = (unsigned short)*height;
13811         cursor->img.region[0] = (unsigned short)(atlas->custom.x + nk_cursor_data[i][0].x);
13812         cursor->img.region[1] = (unsigned short)(atlas->custom.y + nk_cursor_data[i][0].y);
13813         cursor->img.region[2] = (unsigned short)nk_cursor_data[i][1].x;
13814         cursor->img.region[3] = (unsigned short)nk_cursor_data[i][1].y;
13815         cursor->size = nk_cursor_data[i][1];
13816         cursor->offset = nk_cursor_data[i][2];
13817     }}
13818     /* free temporary memory */
13819     atlas->temporary.free(atlas->temporary.userdata, tmp);
13820     return atlas->pixel;
13821
13822 failed:
13823     /* error so cleanup all memory */
13824     if (tmp) atlas->temporary.free(atlas->temporary.userdata, tmp);
13825     if (atlas->glyphs) {
13826         atlas->permanent.free(atlas->permanent.userdata, atlas->glyphs);
13827         atlas->glyphs = 0;
13828     }
13829     if (atlas->pixel) {
13830         atlas->temporary.free(atlas->temporary.userdata, atlas->pixel);
13831         atlas->pixel = 0;
13832     }
13833     return 0;
13834 }
13835 NK_API void
13836 nk_font_atlas_end(struct nk_font_atlas *atlas, nk_handle texture,
13837     struct nk_draw_null_texture *null)
13838 {
13839     int i = 0;
13840     struct nk_font *font_iter;
13841     NK_ASSERT(atlas);
13842     if (!atlas) {
13843         if (!null) return;
13844         null->texture = texture;
13845         null->uv = nk_vec2(0.5f,0.5f);
13846     }
13847     if (null) {
13848         null->texture = texture;
13849         null->uv.x = (atlas->custom.x + 0.5f)/(float)atlas->tex_width;
13850         null->uv.y = (atlas->custom.y + 0.5f)/(float)atlas->tex_height;
13851     }
13852     for (font_iter = atlas->fonts; font_iter; font_iter = font_iter->next) {
13853         font_iter->texture = texture;
13854 #ifdef NK_INCLUDE_VERTEX_BUFFER_OUTPUT
13855         font_iter->handle.texture = texture;
13856 #endif
13857     }
13858     for (i = 0; i < NK_CURSOR_COUNT; ++i)
13859         atlas->cursors[i].img.handle = texture;
13860
13861     atlas->temporary.free(atlas->temporary.userdata, atlas->pixel);
13862     atlas->pixel = 0;
13863     atlas->tex_width = 0;
13864     atlas->tex_height = 0;
```

```
13865     atlas->custom.x = 0;
13866     atlas->custom.y = 0;
13867     atlas->custom.w = 0;
13868     atlas->custom.h = 0;
13869 }
13870 NK_API void
13871 nk_font_atlas_cleanup(struct nk_font_atlas *atlas)
13872 {
13873     NK_ASSERT(atlas);
13874     NK_ASSERT(atlas->temporary.alloc);
13875     NK_ASSERT(atlas->temporary.free);
13876     NK_ASSERT(atlas->permanent.alloc);
13877     NK_ASSERT(atlas->permanent.free);
13878     if (!atlas || !atlas->permanent.alloc || !atlas->permanent.free) return;
13879     if (atlas->config) {
13880         struct nk_font_config *iter;
13881         for (iter = atlas->config; iter; iter = iter->next) {
13882             struct nk_font_config *i;
13883             for (i = iter->n; i != iter; i = i->n) {
13884                 atlas->permanent.free(atlas->permanent.userdata, i->ttf_blob);
13885                 i->ttf_blob = 0;
13886             }
13887             atlas->permanent.free(atlas->permanent.userdata, iter->ttf_blob);
13888             iter->ttf_blob = 0;
13889         }
13890     }
13891 }
13892 NK_API void
13893 nk_font_atlas_clear(struct nk_font_atlas *atlas)
13894 {
13895     NK_ASSERT(atlas);
13896     NK_ASSERT(atlas->temporary.alloc);
13897     NK_ASSERT(atlas->temporary.free);
13898     NK_ASSERT(atlas->permanent.alloc);
13899     NK_ASSERT(atlas->permanent.free);
13900     if (!atlas || !atlas->permanent.alloc || !atlas->permanent.free) return;
13901
13902     if (atlas->config) {
13903         struct nk_font_config *iter, *next;
13904         for (iter = atlas->config; iter; iter = next) {
13905             struct nk_font_config *i, *n;
13906             for (i = iter->n; i != iter; i = n) {
13907                 n = i->n;
13908                 if (i->ttf_blob)
13909                     atlas->permanent.free(atlas->permanent.userdata, i->ttf_blob);
13910                 atlas->permanent.free(atlas->permanent.userdata, i);
13911             }
13912             next = iter->next;
13913             if (i->ttf_blob)
13914                 atlas->permanent.free(atlas->permanent.userdata, iter->ttf_blob);
13915             atlas->permanent.free(atlas->permanent.userdata, iter);
13916         }
13917         atlas->config = 0;
13918     }
13919     if (atlas->fonts) {
13920         struct nk_font *iter, *next;
13921         for (iter = atlas->fonts; iter; iter = next) {
13922             next = iter->next;
13923             atlas->permanent.free(atlas->permanent.userdata, iter);
13924         }
13925         atlas->fonts = 0;
13926     }
13927     if (atlas->glyphs)
13928         atlas->permanent.free(atlas->permanent.userdata, atlas->glyphs);
13929     nk_zero_struct(*atlas);
13930 }
13931 #endif
13932
13933
13934
13935
13936
13937 /* ===============================================================
13938  *
13939  *                          INPUT
13940  *
13941  * ===============================================================*/
13942 NK_API void
13943 nk_input_begin(struct nk_context *ctx)
13944 {
13945     int i;
13946     struct nk_input *in;
13947     NK_ASSERT(ctx);
13948     if (!ctx) return;
13949     in = &ctx->input;
13950     for (i = 0; i < NK_BUTTON_MAX; ++i)
13951         in->mouse.buttons[i].clicked = 0;
```

```
13952
13953     in->keyboard.text_len = 0;
13954     in->mouse.scroll_delta = nk_vec2(0,0);
13955     in->mouse.prev.x = in->mouse.pos.x;
13956     in->mouse.prev.y = in->mouse.pos.y;
13957     in->mouse.delta.x = 0;
13958     in->mouse.delta.y = 0;
13959     for (i = 0; i < NK_KEY_MAX; i++)
13960         in->keyboard.keys[i].clicked = 0;
13961 }
13962 NK_API void
13963 nk_input_end(struct nk_context *ctx)
13964 {
13965     struct nk_input *in;
13966     NK_ASSERT(ctx);
13967     if (!ctx) return;
13968     in = &ctx->input;
13969     if (in->mouse.grab)
13970         in->mouse.grab = 0;
13971     if (in->mouse.ungrab) {
13972         in->mouse.grabbed = 0;
13973         in->mouse.ungrab = 0;
13974         in->mouse.grab = 0;
13975     }
13976 }
13977 NK_API void
13978 nk_input_motion(struct nk_context *ctx, int x, int y)
13979 {
13980     struct nk_input *in;
13981     NK_ASSERT(ctx);
13982     if (!ctx) return;
13983     in = &ctx->input;
13984     in->mouse.pos.x = (float)x;
13985     in->mouse.pos.y = (float)y;
13986     in->mouse.delta.x = in->mouse.pos.x - in->mouse.prev.x;
13987     in->mouse.delta.y = in->mouse.pos.y - in->mouse.prev.y;
13988 }
13989 NK_API void
13990 nk_input_key(struct nk_context *ctx, enum nk_keys key, int down)
13991 {
13992     struct nk_input *in;
13993     NK_ASSERT(ctx);
13994     if (!ctx) return;
13995     in = &ctx->input;
13996 #ifdef NK_KEYSTATE_BASED_INPUT
13997     if (in->keyboard.keys[key].down != down)
13998         in->keyboard.keys[key].clicked++;
13999 #else
14000     in->keyboard.keys[key].clicked++;
14001 #endif
14002     in->keyboard.keys[key].down = down;
14003 }
14004 NK_API void
14005 nk_input_button(struct nk_context *ctx, enum nk_buttons id, int x, int y, int down)
14006 {
14007     struct nk_mouse_button *btn;
14008     struct nk_input *in;
14009     NK_ASSERT(ctx);
14010     if (!ctx) return;
14011     in = &ctx->input;
14012     if (in->mouse.buttons[id].down == down) return;
14013
14014     btn = &in->mouse.buttons[id];
14015     btn->clicked_pos.x = (float)x;
14016     btn->clicked_pos.y = (float)y;
14017     btn->down = down;
14018     btn->clicked++;
14019 }
14020 NK_API void
14021 nk_input_scroll(struct nk_context *ctx, struct nk_vec2 val)
14022 {
14023     NK_ASSERT(ctx);
14024     if (!ctx) return;
14025     ctx->input.mouse.scroll_delta.x += val.x;
14026     ctx->input.mouse.scroll_delta.y += val.y;
14027 }
14028 NK_API void
14029 nk_input_glyph(struct nk_context *ctx, const nk_glyph glyph)
14030 {
14031     int len = 0;
14032     nk_rune unicode;
14033     struct nk_input *in;
14034
14035     NK_ASSERT(ctx);
14036     if (!ctx) return;
14037     in = &ctx->input;
14038
```

```
14039     len = nk_utf_decode(glyph, &unicode, NK_UTF_SIZE);
14040     if (len && ((in->keyboard.text_len + len) < NK_INPUT_MAX)) {
14041         nk_utf_encode(unicode, &in->keyboard.text[in->keyboard.text_len],
14042             NK_INPUT_MAX - in->keyboard.text_len);
14043         in->keyboard.text_len += len;
14044     }
14045 }
14046 NK_API void
14047 nk_input_char(struct nk_context *ctx, char c)
14048 {
14049     nk_glyph glyph;
14050     NK_ASSERT(ctx);
14051     if (!ctx) return;
14052     glyph[0] = c;
14053     nk_input_glyph(ctx, glyph);
14054 }
14055 NK_API void
14056 nk_input_unicode(struct nk_context *ctx, nk_rune unicode)
14057 {
14058     nk_glyph rune;
14059     NK_ASSERT(ctx);
14060     if (!ctx) return;
14061     nk_utf_encode(unicode, rune, NK_UTF_SIZE);
14062     nk_input_glyph(ctx, rune);
14063 }
14064 NK_API int
14065 nk_input_has_mouse_click(const struct nk_input *i, enum nk_buttons id)
14066 {
14067     const struct nk_mouse_button *btn;
14068     if (!i) return nk_false;
14069     btn = &i->mouse.buttons[id];
14070     return (btn->clicked && btn->down == nk_false) ? nk_true : nk_false;
14071 }
14072 NK_API int
14073 nk_input_has_mouse_click_in_rect(const struct nk_input *i, enum nk_buttons id,
14074     struct nk_rect b)
14075 {
14076     const struct nk_mouse_button *btn;
14077     if (!i) return nk_false;
14078     btn = &i->mouse.buttons[id];
14079     if (!NK_INBOX(btn->clicked_pos.x,btn->clicked_pos.y,b.x,b.y,b.w,b.h))
14080         return nk_false;
14081     return nk_true;
14082 }
14083 NK_API int
14084 nk_input_has_mouse_click_down_in_rect(const struct nk_input *i, enum nk_buttons id,
14085     struct nk_rect b, int down)
14086 {
14087     const struct nk_mouse_button *btn;
14088     if (!i) return nk_false;
14089     btn = &i->mouse.buttons[id];
14090     return nk_input_has_mouse_click_in_rect(i, id, b) && (btn->down == down);
14091 }
14092 NK_API int
14093 nk_input_is_mouse_click_in_rect(const struct nk_input *i, enum nk_buttons id,
14094     struct nk_rect b)
14095 {
14096     const struct nk_mouse_button *btn;
14097     if (!i) return nk_false;
14098     btn = &i->mouse.buttons[id];
14099     return (nk_input_has_mouse_click_down_in_rect(i, id, b, nk_false) &&
14100             btn->clicked) ? nk_true : nk_false;
14101 }
14102 NK_API int
14103 nk_input_is_mouse_click_down_in_rect(const struct nk_input *i, enum nk_buttons id,
14104     struct nk_rect b, int down)
14105 {
14106     const struct nk_mouse_button *btn;
14107     if (!i) return nk_false;
14108     btn = &i->mouse.buttons[id];
14109     return (nk_input_has_mouse_click_down_in_rect(i, id, b, down) &&
14110             btn->clicked) ? nk_true : nk_false;
14111 }
14112 NK_API int
14113 nk_input_any_mouse_click_in_rect(const struct nk_input *in, struct nk_rect b)
14114 {
14115     int i, down = 0;
14116     for (i = 0; i < NK_BUTTON_MAX; ++i)
14117         down = down || nk_input_is_mouse_click_in_rect(in, (enum nk_buttons)i, b);
14118     return down;
14119 }
14120 NK_API int
14121 nk_input_is_mouse_hovering_rect(const struct nk_input *i, struct nk_rect rect)
14122 {
14123     if (!i) return nk_false;
14124     return NK_INBOX(i->mouse.pos.x, i->mouse.pos.y, rect.x, rect.y, rect.w, rect.h);
14125 }
```

```
14126 NK_API int
14127 nk_input_is_mouse_prev_hovering_rect(const struct nk_input *i, struct nk_rect rect)
14128 {
14129     if (!i) return nk_false;
14130     return NK_INBOX(i->mouse.prev.x, i->mouse.prev.y, rect.x, rect.y, rect.w, rect.h);
14131 }
14132 NK_API int
14133 nk_input_mouse_clicked(const struct nk_input *i, enum nk_buttons id, struct nk_rect rect)
14134 {
14135     if (!i) return nk_false;
14136     if (!nk_input_is_mouse_hovering_rect(i, rect)) return nk_false;
14137     return nk_input_is_mouse_click_in_rect(i, id, rect);
14138 }
14139 NK_API int
14140 nk_input_is_mouse_down(const struct nk_input *i, enum nk_buttons id)
14141 {
14142     if (!i) return nk_false;
14143     return i->mouse.buttons[id].down;
14144 }
14145 NK_API int
14146 nk_input_is_mouse_pressed(const struct nk_input *i, enum nk_buttons id)
14147 {
14148     const struct nk_mouse_button *b;
14149     if (!i) return nk_false;
14150     b = &i->mouse.buttons[id];
14151     if (b->down && b->clicked)
14152         return nk_true;
14153     return nk_false;
14154 }
14155 NK_API int
14156 nk_input_is_mouse_released(const struct nk_input *i, enum nk_buttons id)
14157 {
14158     if (!i) return nk_false;
14159     return (!i->mouse.buttons[id].down && i->mouse.buttons[id].clicked);
14160 }
14161 NK_API int
14162 nk_input_is_key_pressed(const struct nk_input *i, enum nk_keys key)
14163 {
14164     const struct nk_key *k;
14165     if (!i) return nk_false;
14166     k = &i->keyboard.keys[key];
14167     if ((k->down && k->clicked) || (!k->down && k->clicked >= 2))
14168         return nk_true;
14169     return nk_false;
14170 }
14171 NK_API int
14172 nk_input_is_key_released(const struct nk_input *i, enum nk_keys key)
14173 {
14174     const struct nk_key *k;
14175     if (!i) return nk_false;
14176     k = &i->keyboard.keys[key];
14177     if ((!k->down && k->clicked) || (k->down && k->clicked >= 2))
14178         return nk_true;
14179     return nk_false;
14180 }
14181 NK_API int
14182 nk_input_is_key_down(const struct nk_input *i, enum nk_keys key)
14183 {
14184     const struct nk_key *k;
14185     if (!i) return nk_false;
14186     k = &i->keyboard.keys[key];
14187     if (k->down) return nk_true;
14188     return nk_false;
14189 }
14190
14191
14192
14193
14194
14195 /* ===============================================================
14196  *
14197  *                              STYLE
14198  *
14199  * ===============================================================*/
14200 NK_API void nk_style_default(struct nk_context *ctx){nk_style_from_table(ctx, 0);}
14201 #define NK_COLOR_MAP(NK_COLOR)\
14202     NK_COLOR(NK_COLOR_TEXT,                     175,175,175,255) \
14203     NK_COLOR(NK_COLOR_WINDOW,                   45, 45, 45, 255) \
14204     NK_COLOR(NK_COLOR_HEADER,                   40, 40, 40, 255) \
14205     NK_COLOR(NK_COLOR_BORDER,                   65, 65, 65, 255) \
14206     NK_COLOR(NK_COLOR_BUTTON,                   50, 50, 50, 255) \
14207     NK_COLOR(NK_COLOR_BUTTON_HOVER,             40, 40, 40, 255) \
14208     NK_COLOR(NK_COLOR_BUTTON_ACTIVE,            35, 35, 35, 255) \
14209     NK_COLOR(NK_COLOR_TOGGLE,                   100,100,100,255) \
14210     NK_COLOR(NK_COLOR_TOGGLE_HOVER,             120,120,120,255) \
14211     NK_COLOR(NK_COLOR_TOGGLE_CURSOR,            45, 45, 45, 255) \
14212     NK_COLOR(NK_COLOR_SELECT,                   45, 45, 45, 255) \
```

```
14213     NK_COLOR(NK_COLOR_SELECT_ACTIVE,            35, 35, 35,255) \
14214     NK_COLOR(NK_COLOR_SLIDER,                   38, 38, 38, 255) \
14215     NK_COLOR(NK_COLOR_SLIDER_CURSOR,            100,100,100,255) \
14216     NK_COLOR(NK_COLOR_SLIDER_CURSOR_HOVER,      120,120,120,255) \
14217     NK_COLOR(NK_COLOR_SLIDER_CURSOR_ACTIVE,     150,150,150,255) \
14218     NK_COLOR(NK_COLOR_PROPERTY,                 38, 38, 38, 255) \
14219     NK_COLOR(NK_COLOR_EDIT,                     38, 38, 38, 255)  \
14220     NK_COLOR(NK_COLOR_EDIT_CURSOR,              175,175,175,255) \
14221     NK_COLOR(NK_COLOR_COMBO,                    45, 45, 45, 255) \
14222     NK_COLOR(NK_COLOR_CHART,                    120,120,120,255) \
14223     NK_COLOR(NK_COLOR_CHART_COLOR,              45, 45, 45, 255) \
14224     NK_COLOR(NK_COLOR_CHART_COLOR_HIGHLIGHT,    255, 0,  0, 255) \
14225     NK_COLOR(NK_COLOR_SCROLLBAR,                40, 40, 40, 255) \
14226     NK_COLOR(NK_COLOR_SCROLLBAR_CURSOR,         100,100,100,255) \
14227     NK_COLOR(NK_COLOR_SCROLLBAR_CURSOR_HOVER,   120,120,120,255) \
14228     NK_COLOR(NK_COLOR_SCROLLBAR_CURSOR_ACTIVE,  150,150,150,255) \
14229     NK_COLOR(NK_COLOR_TAB_HEADER,               40, 40, 40,255)
14230
14231 NK_GLOBAL const struct nk_color
14232 nk_default_color_style[NK_COLOR_COUNT] = {
14233 #define NK_COLOR(a,b,c,d,e) {b,c,d,e},
14234     NK_COLOR_MAP(NK_COLOR)
14235 #undef NK_COLOR
14236 };
14237 NK_GLOBAL const char *nk_color_names[NK_COLOR_COUNT] = {
14238 #define NK_COLOR(a,b,c,d,e) #a,
14239     NK_COLOR_MAP(NK_COLOR)
14240 #undef NK_COLOR
14241 };
14242
14243 NK_API const char*
14244 nk_style_get_color_by_name(enum nk_style_colors c)
14245 {
14246     return nk_color_names[c];
14247 }
14248 NK_API struct nk_style_item
14249 nk_style_item_image(struct nk_image img)
14250 {
14251     struct nk_style_item i;
14252     i.type = NK_STYLE_ITEM_IMAGE;
14253     i.data.image = img;
14254     return i;
14255 }
14256 NK_API struct nk_style_item
14257 nk_style_item_color(struct nk_color col)
14258 {
14259     struct nk_style_item i;
14260     i.type = NK_STYLE_ITEM_COLOR;
14261     i.data.color = col;
14262     return i;
14263 }
14264 NK_API struct nk_style_item
14265 nk_style_item_hide(void)
14266 {
14267     struct nk_style_item i;
14268     i.type = NK_STYLE_ITEM_COLOR;
14269     i.data.color = nk_rgba(0,0,0,0);
14270     return i;
14271 }
14272 NK_API void
14273 nk_style_from_table(struct nk_context *ctx, const struct nk_color *table)
14274 {
14275     struct nk_style *style;
14276     struct nk_style_text *text;
14277     struct nk_style_button *button;
14278     struct nk_style_toggle *toggle;
14279     struct nk_style_selectable *select;
14280     struct nk_style_slider *slider;
14281     struct nk_style_progress *prog;
14282     struct nk_style_scrollbar *scroll;
14283     struct nk_style_edit *edit;
14284     struct nk_style_property *property;
14285     struct nk_style_combo *combo;
14286     struct nk_style_chart *chart;
14287     struct nk_style_tab *tab;
14288     struct nk_style_window *win;
14289
14290     NK_ASSERT(ctx);
14291     if (!ctx) return;
14292     style = &ctx->style;
14293     table = (!table) ? nk_default_color_style: table;
14294
14295     /* default text */
14296     text = &style->text;
14297     text->color = table[NK_COLOR_TEXT];
14298     text->padding = nk_vec2(0,0);
14299
```

```
14300     /* default button */
14301     button = &style->button;
14302     nk_zero_struct(*button);
14303     button->normal          = nk_style_item_color(table[NK_COLOR_BUTTON]);
14304     button->hover           = nk_style_item_color(table[NK_COLOR_BUTTON_HOVER]);
14305     button->active          = nk_style_item_color(table[NK_COLOR_BUTTON_ACTIVE]);
14306     button->border_color    = table[NK_COLOR_BORDER];
14307     button->text_background = table[NK_COLOR_BUTTON];
14308     button->text_normal     = table[NK_COLOR_TEXT];
14309     button->text_hover      = table[NK_COLOR_TEXT];
14310     button->text_active     = table[NK_COLOR_TEXT];
14311     button->padding         = nk_vec2(2.0f,2.0f);
14312     button->image_padding   = nk_vec2(0.0f,0.0f);
14313     button->touch_padding   = nk_vec2(0.0f, 0.0f);
14314     button->userdata        = nk_handle_ptr(0);
14315     button->text_alignment  = NK_TEXT_CENTERED;
14316     button->border          = 1.0f;
14317     button->rounding        = 4.0f;
14318     button->draw_begin      = 0;
14319     button->draw_end        = 0;
14320
14321     /* contextual button */
14322     button = &style->contextual_button;
14323     nk_zero_struct(*button);
14324     button->normal          = nk_style_item_color(table[NK_COLOR_WINDOW]);
14325     button->hover           = nk_style_item_color(table[NK_COLOR_BUTTON_HOVER]);
14326     button->active          = nk_style_item_color(table[NK_COLOR_BUTTON_ACTIVE]);
14327     button->border_color    = table[NK_COLOR_WINDOW];
14328     button->text_background = table[NK_COLOR_WINDOW];
14329     button->text_normal     = table[NK_COLOR_TEXT];
14330     button->text_hover      = table[NK_COLOR_TEXT];
14331     button->text_active     = table[NK_COLOR_TEXT];
14332     button->padding         = nk_vec2(2.0f,2.0f);
14333     button->touch_padding   = nk_vec2(0.0f,0.0f);
14334     button->userdata        = nk_handle_ptr(0);
14335     button->text_alignment  = NK_TEXT_CENTERED;
14336     button->border          = 0.0f;
14337     button->rounding        = 0.0f;
14338     button->draw_begin      = 0;
14339     button->draw_end        = 0;
14340
14341     /* menu button */
14342     button = &style->menu_button;
14343     nk_zero_struct(*button);
14344     button->normal          = nk_style_item_color(table[NK_COLOR_WINDOW]);
14345     button->hover           = nk_style_item_color(table[NK_COLOR_WINDOW]);
14346     button->active          = nk_style_item_color(table[NK_COLOR_WINDOW]);
14347     button->border_color    = table[NK_COLOR_WINDOW];
14348     button->text_background = table[NK_COLOR_WINDOW];
14349     button->text_normal     = table[NK_COLOR_TEXT];
14350     button->text_hover      = table[NK_COLOR_TEXT];
14351     button->text_active     = table[NK_COLOR_TEXT];
14352     button->padding         = nk_vec2(2.0f,2.0f);
14353     button->touch_padding   = nk_vec2(0.0f,0.0f);
14354     button->userdata        = nk_handle_ptr(0);
14355     button->text_alignment  = NK_TEXT_CENTERED;
14356     button->border          = 0.0f;
14357     button->rounding        = 1.0f;
14358     button->draw_begin      = 0;
14359     button->draw_end        = 0;
14360
14361     /* checkbox toggle */
14362     toggle = &style->checkbox;
14363     nk_zero_struct(*toggle);
14364     toggle->normal          = nk_style_item_color(table[NK_COLOR_TOGGLE]);
14365     toggle->hover           = nk_style_item_color(table[NK_COLOR_TOGGLE_HOVER]);
14366     toggle->active          = nk_style_item_color(table[NK_COLOR_TOGGLE_HOVER]);
14367     toggle->cursor_normal   = nk_style_item_color(table[NK_COLOR_TOGGLE_CURSOR]);
14368     toggle->cursor_hover    = nk_style_item_color(table[NK_COLOR_TOGGLE_CURSOR]);
14369     toggle->userdata        = nk_handle_ptr(0);
14370     toggle->text_background = table[NK_COLOR_WINDOW];
14371     toggle->text_normal     = table[NK_COLOR_TEXT];
14372     toggle->text_hover      = table[NK_COLOR_TEXT];
14373     toggle->text_active     = table[NK_COLOR_TEXT];
14374     toggle->padding         = nk_vec2(2.0f, 2.0f);
14375     toggle->touch_padding   = nk_vec2(0,0);
14376     toggle->border_color    = nk_rgba(0,0,0,0);
14377     toggle->border          = 0.0f;
14378     toggle->spacing         = 4;
14379
14380     /* option toggle */
14381     toggle = &style->option;
14382     nk_zero_struct(*toggle);
14383     toggle->normal          = nk_style_item_color(table[NK_COLOR_TOGGLE]);
14384     toggle->hover           = nk_style_item_color(table[NK_COLOR_TOGGLE_HOVER]);
14385     toggle->active          = nk_style_item_color(table[NK_COLOR_TOGGLE_HOVER]);
14386     toggle->cursor_normal   = nk_style_item_color(table[NK_COLOR_TOGGLE_CURSOR]);
```

```
14387     toggle->cursor_hover    = nk_style_item_color(table[NK_COLOR_TOGGLE_CURSOR]);
14388     toggle->userdata        = nk_handle_ptr(0);
14389     toggle->text_background = table[NK_COLOR_WINDOW];
14390     toggle->text_normal     = table[NK_COLOR_TEXT];
14391     toggle->text_hover      = table[NK_COLOR_TEXT];
14392     toggle->text_active     = table[NK_COLOR_TEXT];
14393     toggle->padding         = nk_vec2(3.0f, 3.0f);
14394     toggle->touch_padding   = nk_vec2(0,0);
14395     toggle->border_color    = nk_rgba(0,0,0,0);
14396     toggle->border          = 0.0f;
14397     toggle->spacing         = 4;
14398
14399     /* selectable */
14400     select = &style->selectable;
14401     nk_zero_struct(*select);
14402     select->normal          = nk_style_item_color(table[NK_COLOR_SELECT]);
14403     select->hover           = nk_style_item_color(table[NK_COLOR_SELECT]);
14404     select->pressed         = nk_style_item_color(table[NK_COLOR_SELECT]);
14405     select->normal_active   = nk_style_item_color(table[NK_COLOR_SELECT_ACTIVE]);
14406     select->hover_active    = nk_style_item_color(table[NK_COLOR_SELECT_ACTIVE]);
14407     select->pressed_active  = nk_style_item_color(table[NK_COLOR_SELECT_ACTIVE]);
14408     select->text_normal     = table[NK_COLOR_TEXT];
14409     select->text_hover      = table[NK_COLOR_TEXT];
14410     select->text_pressed    = table[NK_COLOR_TEXT];
14411     select->text_normal_active  = table[NK_COLOR_TEXT];
14412     select->text_hover_active   = table[NK_COLOR_TEXT];
14413     select->text_pressed_active = table[NK_COLOR_TEXT];
14414     select->padding         = nk_vec2(2.0f,2.0f);
14415     select->image_padding   = nk_vec2(2.0f,2.0f);
14416     select->touch_padding   = nk_vec2(0,0);
14417     select->userdata        = nk_handle_ptr(0);
14418     select->rounding        = 0.0f;
14419     select->draw_begin      = 0;
14420     select->draw_end        = 0;
14421
14422     /* slider */
14423     slider = &style->slider;
14424     nk_zero_struct(*slider);
14425     slider->normal          = nk_style_item_hide();
14426     slider->hover           = nk_style_item_hide();
14427     slider->active          = nk_style_item_hide();
14428     slider->bar_normal      = table[NK_COLOR_SLIDER];
14429     slider->bar_hover       = table[NK_COLOR_SLIDER];
14430     slider->bar_active      = table[NK_COLOR_SLIDER];
14431     slider->bar_filled      = table[NK_COLOR_SLIDER_CURSOR];
14432     slider->cursor_normal   = nk_style_item_color(table[NK_COLOR_SLIDER_CURSOR]);
14433     slider->cursor_hover    = nk_style_item_color(table[NK_COLOR_SLIDER_CURSOR_HOVER]);
14434     slider->cursor_active   = nk_style_item_color(table[NK_COLOR_SLIDER_CURSOR_ACTIVE]);
14435     slider->inc_symbol      = NK_SYMBOL_TRIANGLE_RIGHT;
14436     slider->dec_symbol      = NK_SYMBOL_TRIANGLE_LEFT;
14437     slider->cursor_size     = nk_vec2(16,16);
14438     slider->padding         = nk_vec2(2,2);
14439     slider->spacing         = nk_vec2(2,2);
14440     slider->userdata        = nk_handle_ptr(0);
14441     slider->show_buttons    = nk_false;
14442     slider->bar_height      = 8;
14443     slider->rounding        = 0;
14444     slider->draw_begin      = 0;
14445     slider->draw_end        = 0;
14446
14447     /* slider buttons */
14448     button = &style->slider.inc_button;
14449     button->normal          = nk_style_item_color(nk_rgb(40,40,40));
14450     button->hover           = nk_style_item_color(nk_rgb(42,42,42));
14451     button->active          = nk_style_item_color(nk_rgb(44,44,44));
14452     button->border_color    = nk_rgb(65,65,65);
14453     button->text_background = nk_rgb(40,40,40);
14454     button->text_normal     = nk_rgb(175,175,175);
14455     button->text_hover      = nk_rgb(175,175,175);
14456     button->text_active     = nk_rgb(175,175,175);
14457     button->padding         = nk_vec2(8.0f,8.0f);
14458     button->touch_padding   = nk_vec2(0.0f,0.0f);
14459     button->userdata        = nk_handle_ptr(0);
14460     button->text_alignment  = NK_TEXT_CENTERED;
14461     button->border          = 1.0f;
14462     button->rounding        = 0.0f;
14463     button->draw_begin      = 0;
14464     button->draw_end        = 0;
14465     style->slider.dec_button = style->slider.inc_button;
14466
14467     /* progressbar */
14468     prog = &style->progress;
14469     nk_zero_struct(*prog);
14470     prog->normal            = nk_style_item_color(table[NK_COLOR_SLIDER]);
14471     prog->hover             = nk_style_item_color(table[NK_COLOR_SLIDER]);
14472     prog->active            = nk_style_item_color(table[NK_COLOR_SLIDER]);
14473     prog->cursor_normal     = nk_style_item_color(table[NK_COLOR_SLIDER_CURSOR]);
```

```
14474     prog->cursor_hover      = nk_style_item_color(table[NK_COLOR_SLIDER_CURSOR_HOVER]);
14475     prog->cursor_active     = nk_style_item_color(table[NK_COLOR_SLIDER_CURSOR_ACTIVE]);
14476     prog->border_color      = nk_rgba(0,0,0,0);
14477     prog->cursor_border_color = nk_rgba(0,0,0,0);
14478     prog->userdata          = nk_handle_ptr(0);
14479     prog->padding           = nk_vec2(4,4);
14480     prog->rounding          = 0;
14481     prog->border            = 0;
14482     prog->cursor_rounding   = 0;
14483     prog->cursor_border     = 0;
14484     prog->draw_begin        = 0;
14485     prog->draw_end          = 0;
14486
14487     /* scrollbars */
14488     scroll = &style->scrollh;
14489     nk_zero_struct(*scroll);
14490     scroll->normal          = nk_style_item_color(table[NK_COLOR_SCROLLBAR]);
14491     scroll->hover           = nk_style_item_color(table[NK_COLOR_SCROLLBAR]);
14492     scroll->active          = nk_style_item_color(table[NK_COLOR_SCROLLBAR]);
14493     scroll->cursor_normal   = nk_style_item_color(table[NK_COLOR_SCROLLBAR_CURSOR]);
14494     scroll->cursor_hover    = nk_style_item_color(table[NK_COLOR_SCROLLBAR_CURSOR_HOVER]);
14495     scroll->cursor_active   = nk_style_item_color(table[NK_COLOR_SCROLLBAR_CURSOR_ACTIVE]);
14496     scroll->dec_symbol      = NK_SYMBOL_CIRCLE_SOLID;
14497     scroll->inc_symbol      = NK_SYMBOL_CIRCLE_SOLID;
14498     scroll->userdata        = nk_handle_ptr(0);
14499     scroll->border_color    = table[NK_COLOR_SCROLLBAR];
14500     scroll->cursor_border_color = table[NK_COLOR_SCROLLBAR];
14501     scroll->padding         = nk_vec2(0,0);
14502     scroll->show_buttons    = nk_false;
14503     scroll->border          = 0;
14504     scroll->rounding        = 0;
14505     scroll->border_cursor   = 0;
14506     scroll->rounding_cursor = 0;
14507     scroll->draw_begin      = 0;
14508     scroll->draw_end        = 0;
14509     style->scrollv = style->scrollh;
14510
14511     /* scrollbars buttons */
14512     button = &style->scrollh.inc_button;
14513     button->normal          = nk_style_item_color(nk_rgb(40,40,40));
14514     button->hover           = nk_style_item_color(nk_rgb(42,42,42));
14515     button->active          = nk_style_item_color(nk_rgb(44,44,44));
14516     button->border_color    = nk_rgb(65,65,65);
14517     button->text_background = nk_rgb(40,40,40);
14518     button->text_normal     = nk_rgb(175,175,175);
14519     button->text_hover      = nk_rgb(175,175,175);
14520     button->text_active     = nk_rgb(175,175,175);
14521     button->padding         = nk_vec2(4.0f,4.0f);
14522     button->touch_padding   = nk_vec2(0.0f,0.0f);
14523     button->userdata        = nk_handle_ptr(0);
14524     button->text_alignment  = NK_TEXT_CENTERED;
14525     button->border          = 1.0f;
14526     button->rounding        = 0.0f;
14527     button->draw_begin      = 0;
14528     button->draw_end        = 0;
14529     style->scrollh.dec_button = style->scrollh.inc_button;
14530     style->scrollv.inc_button = style->scrollh.inc_button;
14531     style->scrollv.dec_button = style->scrollh.inc_button;
14532
14533     /* edit */
14534     edit = &style->edit;
14535     nk_zero_struct(*edit);
14536     edit->normal            = nk_style_item_color(table[NK_COLOR_EDIT]);
14537     edit->hover             = nk_style_item_color(table[NK_COLOR_EDIT]);
14538     edit->active            = nk_style_item_color(table[NK_COLOR_EDIT]);
14539     edit->cursor_normal     = table[NK_COLOR_TEXT];
14540     edit->cursor_hover      = table[NK_COLOR_TEXT];
14541     edit->cursor_text_normal= table[NK_COLOR_EDIT];
14542     edit->cursor_text_hover = table[NK_COLOR_EDIT];
14543     edit->border_color      = table[NK_COLOR_BORDER];
14544     edit->text_normal       = table[NK_COLOR_TEXT];
14545     edit->text_hover        = table[NK_COLOR_TEXT];
14546     edit->text_active       = table[NK_COLOR_TEXT];
14547     edit->selected_normal   = table[NK_COLOR_TEXT];
14548     edit->selected_hover    = table[NK_COLOR_TEXT];
14549     edit->selected_text_normal  = table[NK_COLOR_EDIT];
14550     edit->selected_text_hover   = table[NK_COLOR_EDIT];
14551     edit->scrollbar_size    = nk_vec2(10,10);
14552     edit->scrollbar         = style->scrollv;
14553     edit->padding           = nk_vec2(4,4);
14554     edit->row_padding       = 2;
14555     edit->cursor_size       = 4;
14556     edit->border            = 1;
14557     edit->rounding          = 0;
14558
14559     /* property */
14560     property = &style->property;
```

```
14561     nk_zero_struct(*property);
14562     property->normal        = nk_style_item_color(table[NK_COLOR_PROPERTY]);
14563     property->hover         = nk_style_item_color(table[NK_COLOR_PROPERTY]);
14564     property->active        = nk_style_item_color(table[NK_COLOR_PROPERTY]);
14565     property->border_color  = table[NK_COLOR_BORDER];
14566     property->label_normal  = table[NK_COLOR_TEXT];
14567     property->label_hover   = table[NK_COLOR_TEXT];
14568     property->label_active  = table[NK_COLOR_TEXT];
14569     property->sym_left      = NK_SYMBOL_TRIANGLE_LEFT;
14570     property->sym_right     = NK_SYMBOL_TRIANGLE_RIGHT;
14571     property->userdata      = nk_handle_ptr(0);
14572     property->padding       = nk_vec2(4,4);
14573     property->border        = 1;
14574     property->rounding      = 10;
14575     property->draw_begin    = 0;
14576     property->draw_end      = 0;
14577
14578     /* property buttons */
14579     button = &style->property.dec_button;
14580     nk_zero_struct(*button);
14581     button->normal          = nk_style_item_color(table[NK_COLOR_PROPERTY]);
14582     button->hover           = nk_style_item_color(table[NK_COLOR_PROPERTY]);
14583     button->active          = nk_style_item_color(table[NK_COLOR_PROPERTY]);
14584     button->border_color    = nk_rgba(0,0,0,0);
14585     button->text_background = table[NK_COLOR_PROPERTY];
14586     button->text_normal     = table[NK_COLOR_TEXT];
14587     button->text_hover      = table[NK_COLOR_TEXT];
14588     button->text_active     = table[NK_COLOR_TEXT];
14589     button->padding         = nk_vec2(0.0f,0.0f);
14590     button->touch_padding   = nk_vec2(0.0f,0.0f);
14591     button->userdata        = nk_handle_ptr(0);
14592     button->text_alignment  = NK_TEXT_CENTERED;
14593     button->border          = 0.0f;
14594     button->rounding        = 0.0f;
14595     button->draw_begin      = 0;
14596     button->draw_end        = 0;
14597     style->property.inc_button = style->property.dec_button;
14598
14599     /* property edit */
14600     edit = &style->property.edit;
14601     nk_zero_struct(*edit);
14602     edit->normal            = nk_style_item_color(table[NK_COLOR_PROPERTY]);
14603     edit->hover             = nk_style_item_color(table[NK_COLOR_PROPERTY]);
14604     edit->active            = nk_style_item_color(table[NK_COLOR_PROPERTY]);
14605     edit->border_color      = nk_rgba(0,0,0,0);
14606     edit->cursor_normal     = table[NK_COLOR_TEXT];
14607     edit->cursor_hover      = table[NK_COLOR_TEXT];
14608     edit->cursor_text_normal= table[NK_COLOR_EDIT];
14609     edit->cursor_text_hover = table[NK_COLOR_EDIT];
14610     edit->text_normal       = table[NK_COLOR_TEXT];
14611     edit->text_hover        = table[NK_COLOR_TEXT];
14612     edit->text_active       = table[NK_COLOR_TEXT];
14613     edit->selected_normal   = table[NK_COLOR_TEXT];
14614     edit->selected_hover    = table[NK_COLOR_TEXT];
14615     edit->selected_text_normal  = table[NK_COLOR_EDIT];
14616     edit->selected_text_hover   = table[NK_COLOR_EDIT];
14617     edit->padding           = nk_vec2(0,0);
14618     edit->cursor_size       = 8;
14619     edit->border            = 0;
14620     edit->rounding          = 0;
14621
14622     /* chart */
14623     chart = &style->chart;
14624     nk_zero_struct(*chart);
14625     chart->background       = nk_style_item_color(table[NK_COLOR_CHART]);
14626     chart->border_color     = table[NK_COLOR_BORDER];
14627     chart->selected_color   = table[NK_COLOR_CHART_COLOR_HIGHLIGHT];
14628     chart->color            = table[NK_COLOR_CHART_COLOR];
14629     chart->padding          = nk_vec2(4,4);
14630     chart->border           = 0;
14631     chart->rounding         = 0;
14632
14633     /* combo */
14634     combo = &style->combo;
14635     combo->normal           = nk_style_item_color(table[NK_COLOR_COMBO]);
14636     combo->hover            = nk_style_item_color(table[NK_COLOR_COMBO]);
14637     combo->active           = nk_style_item_color(table[NK_COLOR_COMBO]);
14638     combo->border_color     = table[NK_COLOR_BORDER];
14639     combo->label_normal     = table[NK_COLOR_TEXT];
14640     combo->label_hover      = table[NK_COLOR_TEXT];
14641     combo->label_active     = table[NK_COLOR_TEXT];
14642     combo->sym_normal       = NK_SYMBOL_TRIANGLE_DOWN;
14643     combo->sym_hover        = NK_SYMBOL_TRIANGLE_DOWN;
14644     combo->sym_active       = NK_SYMBOL_TRIANGLE_DOWN;
14645     combo->content_padding  = nk_vec2(4,4);
14646     combo->button_padding   = nk_vec2(0,4);
14647     combo->spacing          = nk_vec2(4,0);
```

```
14648     combo->border           = 1;
14649     combo->rounding         = 0;
14650
14651     /* combo button */
14652     button = &style->combo.button;
14653     nk_zero_struct(*button);
14654     button->normal          = nk_style_item_color(table[NK_COLOR_COMBO]);
14655     button->hover           = nk_style_item_color(table[NK_COLOR_COMBO]);
14656     button->active          = nk_style_item_color(table[NK_COLOR_COMBO]);
14657     button->border_color    = nk_rgba(0,0,0,0);
14658     button->text_background = table[NK_COLOR_COMBO];
14659     button->text_normal     = table[NK_COLOR_TEXT];
14660     button->text_hover      = table[NK_COLOR_TEXT];
14661     button->text_active     = table[NK_COLOR_TEXT];
14662     button->padding         = nk_vec2(2.0f,2.0f);
14663     button->touch_padding   = nk_vec2(0.0f,0.0f);
14664     button->userdata        = nk_handle_ptr(0);
14665     button->text_alignment  = NK_TEXT_CENTERED;
14666     button->border          = 0.0f;
14667     button->rounding        = 0.0f;
14668     button->draw_begin      = 0;
14669     button->draw_end        = 0;
14670
14671     /* tab */
14672     tab = &style->tab;
14673     tab->background         = nk_style_item_color(table[NK_COLOR_TAB_HEADER]);
14674     tab->border_color       = table[NK_COLOR_BORDER];
14675     tab->text               = table[NK_COLOR_TEXT];
14676     tab->sym_minimize       = NK_SYMBOL_TRIANGLE_RIGHT;
14677     tab->sym_maximize       = NK_SYMBOL_TRIANGLE_DOWN;
14678     tab->padding            = nk_vec2(4,4);
14679     tab->spacing            = nk_vec2(4,4);
14680     tab->indent             = 10.0f;
14681     tab->border             = 1;
14682     tab->rounding           = 0;
14683
14684     /* tab button */
14685     button = &style->tab.tab_minimize_button;
14686     nk_zero_struct(*button);
14687     button->normal          = nk_style_item_color(table[NK_COLOR_TAB_HEADER]);
14688     button->hover           = nk_style_item_color(table[NK_COLOR_TAB_HEADER]);
14689     button->active          = nk_style_item_color(table[NK_COLOR_TAB_HEADER]);
14690     button->border_color    = nk_rgba(0,0,0,0);
14691     button->text_background = table[NK_COLOR_TAB_HEADER];
14692     button->text_normal     = table[NK_COLOR_TEXT];
14693     button->text_hover      = table[NK_COLOR_TEXT];
14694     button->text_active     = table[NK_COLOR_TEXT];
14695     button->padding         = nk_vec2(2.0f,2.0f);
14696     button->touch_padding   = nk_vec2(0.0f,0.0f);
14697     button->userdata        = nk_handle_ptr(0);
14698     button->text_alignment  = NK_TEXT_CENTERED;
14699     button->border          = 0.0f;
14700     button->rounding        = 0.0f;
14701     button->draw_begin      = 0;
14702     button->draw_end        = 0;
14703     style->tab.tab_maximize_button =*button;
14704
14705     /* node button */
14706     button = &style->tab.node_minimize_button;
14707     nk_zero_struct(*button);
14708     button->normal          = nk_style_item_color(table[NK_COLOR_WINDOW]);
14709     button->hover           = nk_style_item_color(table[NK_COLOR_WINDOW]);
14710     button->active          = nk_style_item_color(table[NK_COLOR_WINDOW]);
14711     button->border_color    = nk_rgba(0,0,0,0);
14712     button->text_background = table[NK_COLOR_TAB_HEADER];
14713     button->text_normal     = table[NK_COLOR_TEXT];
14714     button->text_hover      = table[NK_COLOR_TEXT];
14715     button->text_active     = table[NK_COLOR_TEXT];
14716     button->padding         = nk_vec2(2.0f,2.0f);
14717     button->touch_padding   = nk_vec2(0.0f,0.0f);
14718     button->userdata        = nk_handle_ptr(0);
14719     button->text_alignment  = NK_TEXT_CENTERED;
14720     button->border          = 0.0f;
14721     button->rounding        = 0.0f;
14722     button->draw_begin      = 0;
14723     button->draw_end        = 0;
14724     style->tab.node_maximize_button =*button;
14725
14726     /* window header */
14727     win = &style->window;
14728     win->header.align = NK_HEADER_RIGHT;
14729     win->header.close_symbol = NK_SYMBOL_X;
14730     win->header.minimize_symbol = NK_SYMBOL_MINUS;
14731     win->header.maximize_symbol = NK_SYMBOL_PLUS;
14732     win->header.normal = nk_style_item_color(table[NK_COLOR_HEADER]);
14733     win->header.hover = nk_style_item_color(table[NK_COLOR_HEADER]);
14734     win->header.active = nk_style_item_color(table[NK_COLOR_HEADER]);
```

```
14735     win->header.label_normal = table[NK_COLOR_TEXT];
14736     win->header.label_hover = table[NK_COLOR_TEXT];
14737     win->header.label_active = table[NK_COLOR_TEXT];
14738     win->header.label_padding = nk_vec2(4,4);
14739     win->header.padding = nk_vec2(4,4);
14740     win->header.spacing = nk_vec2(0,0);
14741
14742     /* window header close button */
14743     button = &style->window.header.close_button;
14744     nk_zero_struct(*button);
14745     button->normal          = nk_style_item_color(table[NK_COLOR_HEADER]);
14746     button->hover           = nk_style_item_color(table[NK_COLOR_HEADER]);
14747     button->active          = nk_style_item_color(table[NK_COLOR_HEADER]);
14748     button->border_color    = nk_rgba(0,0,0,0);
14749     button->text_background = table[NK_COLOR_HEADER];
14750     button->text_normal     = table[NK_COLOR_TEXT];
14751     button->text_hover      = table[NK_COLOR_TEXT];
14752     button->text_active     = table[NK_COLOR_TEXT];
14753     button->padding         = nk_vec2(0.0f,0.0f);
14754     button->touch_padding   = nk_vec2(0.0f,0.0f);
14755     button->userdata        = nk_handle_ptr(0);
14756     button->text_alignment  = NK_TEXT_CENTERED;
14757     button->border          = 0.0f;
14758     button->rounding        = 0.0f;
14759     button->draw_begin      = 0;
14760     button->draw_end        = 0;
14761
14762     /* window header minimize button */
14763     button = &style->window.header.minimize_button;
14764     nk_zero_struct(*button);
14765     button->normal          = nk_style_item_color(table[NK_COLOR_HEADER]);
14766     button->hover           = nk_style_item_color(table[NK_COLOR_HEADER]);
14767     button->active          = nk_style_item_color(table[NK_COLOR_HEADER]);
14768     button->border_color    = nk_rgba(0,0,0,0);
14769     button->text_background = table[NK_COLOR_HEADER];
14770     button->text_normal     = table[NK_COLOR_TEXT];
14771     button->text_hover      = table[NK_COLOR_TEXT];
14772     button->text_active     = table[NK_COLOR_TEXT];
14773     button->padding         = nk_vec2(0.0f,0.0f);
14774     button->touch_padding   = nk_vec2(0.0f,0.0f);
14775     button->userdata        = nk_handle_ptr(0);
14776     button->text_alignment  = NK_TEXT_CENTERED;
14777     button->border          = 0.0f;
14778     button->rounding        = 0.0f;
14779     button->draw_begin      = 0;
14780     button->draw_end        = 0;
14781
14782     /* window */
14783     win->background = table[NK_COLOR_WINDOW];
14784     win->fixed_background = nk_style_item_color(table[NK_COLOR_WINDOW]);
14785     win->border_color = table[NK_COLOR_BORDER];
14786     win->popup_border_color = table[NK_COLOR_BORDER];
14787     win->combo_border_color = table[NK_COLOR_BORDER];
14788     win->contextual_border_color = table[NK_COLOR_BORDER];
14789     win->menu_border_color = table[NK_COLOR_BORDER];
14790     win->group_border_color = table[NK_COLOR_BORDER];
14791     win->tooltip_border_color = table[NK_COLOR_BORDER];
14792     win->scaler = nk_style_item_color(table[NK_COLOR_TEXT]);
14793
14794     win->rounding = 0.0f;
14795     win->spacing = nk_vec2(4,4);
14796     win->scrollbar_size = nk_vec2(10,10);
14797     win->min_size = nk_vec2(64,64);
14798
14799     win->combo_border = 1.0f;
14800     win->contextual_border = 1.0f;
14801     win->menu_border = 1.0f;
14802     win->group_border = 1.0f;
14803     win->tooltip_border = 1.0f;
14804     win->popup_border = 1.0f;
14805     win->border = 2.0f;
14806     win->min_row_height_padding = 8;
14807
14808     win->padding = nk_vec2(4,4);
14809     win->group_padding = nk_vec2(4,4);
14810     win->popup_padding = nk_vec2(4,4);
14811     win->combo_padding = nk_vec2(4,4);
14812     win->contextual_padding = nk_vec2(4,4);
14813     win->menu_padding = nk_vec2(4,4);
14814     win->tooltip_padding = nk_vec2(4,4);
14815 }
14816 NK_API void
14817 nk_style_set_font(struct nk_context *ctx, const struct nk_user_font *font)
14818 {
14819     struct nk_style *style;
14820     NK_ASSERT(ctx);
14821
```

```
14822     if (!ctx) return;
14823     style = &ctx->style;
14824     style->font = font;
14825     ctx->stacks.fonts.head = 0;
14826     if (ctx->current)
14827         nk_layout_reset_min_row_height(ctx);
14828 }
14829 NK_API int
14830 nk_style_push_font(struct nk_context *ctx, const struct nk_user_font *font)
14831 {
14832     struct nk_config_stack_user_font *font_stack;
14833     struct nk_config_stack_user_font_element *element;
14834
14835     NK_ASSERT(ctx);
14836     if (!ctx) return 0;
14837
14838     font_stack = &ctx->stacks.fonts;
14839     NK_ASSERT(font_stack->head < (int)NK_LEN(font_stack->elements));
14840     if (font_stack->head >= (int)NK_LEN(font_stack->elements))
14841         return 0;
14842
14843     element = &font_stack->elements[font_stack->head++];
14844     element->address = &ctx->style.font;
14845     element->old_value = ctx->style.font;
14846     ctx->style.font = font;
14847     return 1;
14848 }
14849 NK_API int
14850 nk_style_pop_font(struct nk_context *ctx)
14851 {
14852     struct nk_config_stack_user_font *font_stack;
14853     struct nk_config_stack_user_font_element *element;
14854
14855     NK_ASSERT(ctx);
14856     if (!ctx) return 0;
14857
14858     font_stack = &ctx->stacks.fonts;
14859     NK_ASSERT(font_stack->head > 0);
14860     if (font_stack->head < 1)
14861         return 0;
14862
14863     element = &font_stack->elements[--font_stack->head];
14864     *element->address = element->old_value;
14865     return 1;
14866 }
14867 #define NK_STYLE_PUSH_IMPLEMENATION(prefix, type, stack) \
14868 nk_style_push_##type(struct nk_context *ctx, prefix##_##type *address, prefix##_##type value)\
14869 {\
14870     struct nk_config_stack_##type * type_stack;\
14871     struct nk_config_stack_##type##_element *element;\
14872     NK_ASSERT(ctx);\
14873     if (!ctx) return 0;\
14874     type_stack = &ctx->stacks.stack;\
14875     NK_ASSERT(type_stack->head < (int)NK_LEN(type_stack->elements));\
14876     if (type_stack->head >= (int)NK_LEN(type_stack->elements))\
14877         return 0;\
14878     element = &type_stack->elements[type_stack->head++];\
14879     element->address = address;\
14880     element->old_value = *address;\
14881     *address = value;\
14882     return 1;\
14883 }
14884 #define NK_STYLE_POP_IMPLEMENATION(type, stack) \
14885 nk_style_pop_##type(struct nk_context *ctx)\
14886 {\
14887     struct nk_config_stack_##type *type_stack;\
14888     struct nk_config_stack_##type##_element *element;\
14889     NK_ASSERT(ctx);\
14890     if (!ctx) return 0;\
14891     type_stack = &ctx->stacks.stack;\
14892     NK_ASSERT(type_stack->head > 0);\
14893     if (type_stack->head < 1)\
14894         return 0;\
14895     element = &type_stack->elements[--type_stack->head];\
14896     *element->address = element->old_value;\
14897     return 1;\
14898 }
14899 NK_API int NK_STYLE_PUSH_IMPLEMENATION(struct nk, style_item, style_items)
14900 NK_API int NK_STYLE_PUSH_IMPLEMENATION(nk,float, floats)
14901 NK_API int NK_STYLE_PUSH_IMPLEMENATION(struct nk, vec2, vectors)
14902 NK_API int NK_STYLE_PUSH_IMPLEMENATION(nk,flags, flags)
14903 NK_API int NK_STYLE_PUSH_IMPLEMENATION(struct nk,color, colors)
14904
14905 NK_API int NK_STYLE_POP_IMPLEMENATION(style_item, style_items)
14906 NK_API int NK_STYLE_POP_IMPLEMENATION(float,floats)
14907 NK_API int NK_STYLE_POP_IMPLEMENATION(vec2, vectors)
14908 NK_API int NK_STYLE_POP_IMPLEMENATION(flags,flags)
```

```
14909 NK_API int NK_STYLE_POP_IMPLEMENATION(color,colors)
14910
14911 NK_API int
14912 nk_style_set_cursor(struct nk_context *ctx, enum nk_style_cursor c)
14913 {
14914     struct nk_style *style;
14915     NK_ASSERT(ctx);
14916     if (!ctx) return 0;
14917     style = &ctx->style;
14918     if (style->cursors[c]) {
14919         style->cursor_active = style->cursors[c];
14920         return 1;
14921     }
14922     return 0;
14923 }
14924 NK_API void
14925 nk_style_show_cursor(struct nk_context *ctx)
14926 {
14927     ctx->style.cursor_visible = nk_true;
14928 }
14929 NK_API void
14930 nk_style_hide_cursor(struct nk_context *ctx)
14931 {
14932     ctx->style.cursor_visible = nk_false;
14933 }
14934 NK_API void
14935 nk_style_load_cursor(struct nk_context *ctx, enum nk_style_cursor cursor,
14936     const struct nk_cursor *c)
14937 {
14938     struct nk_style *style;
14939     NK_ASSERT(ctx);
14940     if (!ctx) return;
14941     style = &ctx->style;
14942     style->cursors[cursor] = c;
14943 }
14944 NK_API void
14945 nk_style_load_all_cursors(struct nk_context *ctx, struct nk_cursor *cursors)
14946 {
14947     int i = 0;
14948     struct nk_style *style;
14949     NK_ASSERT(ctx);
14950     if (!ctx) return;
14951     style = &ctx->style;
14952     for (i = 0; i < NK_CURSOR_COUNT; ++i)
14953         style->cursors[i] = &cursors[i];
14954     style->cursor_visible = nk_true;
14955 }
14956
14957
14958
14959
14960
14961 /* ===============================================================
14962  *
14963  *                          CONTEXT
14964  *
14965  * ===============================================================*/
14966 NK_INTERN void
14967 nk_setup(struct nk_context *ctx, const struct nk_user_font *font)
14968 {
14969     NK_ASSERT(ctx);
14970     if (!ctx) return;
14971     nk_zero_struct(*ctx);
14972     nk_style_default(ctx);
14973     ctx->seq = 1;
14974     if (font) ctx->style.font = font;
14975 #ifdef NK_INCLUDE_VERTEX_BUFFER_OUTPUT
14976     nk_draw_list_init(&ctx->draw_list);
14977 #endif
14978 }
14979 #ifdef NK_INCLUDE_DEFAULT_ALLOCATOR
14980 NK_API int
14981 nk_init_default(struct nk_context *ctx, const struct nk_user_font *font)
14982 {
14983     struct nk_allocator alloc;
14984     alloc.userdata.ptr = 0;
14985     alloc.alloc = nk_malloc;
14986     alloc.free = nk_mfree;
14987     return nk_init(ctx, &alloc, font);
14988 }
14989 #endif
14990 NK_API int
14991 nk_init_fixed(struct nk_context *ctx, void *memory, nk_size size,
14992     const struct nk_user_font *font)
14993 {
14994     NK_ASSERT(memory);
14995     if (!memory) return 0;
```

```
14996     nk_setup(ctx, font);
14997     nk_buffer_init_fixed(&ctx->memory, memory, size);
14998     ctx->use_pool = nk_false;
14999     return 1;
15000 }
15001 NK_API int
15002 nk_init_custom(struct nk_context *ctx, struct nk_buffer *cmds,
15003     struct nk_buffer *pool, const struct nk_user_font *font)
15004 {
15005     NK_ASSERT(cmds);
15006     NK_ASSERT(pool);
15007     if (!cmds || !pool) return 0;
15008
15009     nk_setup(ctx, font);
15010     ctx->memory = *cmds;
15011     if (pool->type == NK_BUFFER_FIXED) {
15012         /* take memory from buffer and alloc fixed pool */
15013         nk_pool_init_fixed(&ctx->pool, pool->memory.ptr, pool->memory.size);
15014     } else {
15015         /* create dynamic pool from buffer allocator */
15016         struct nk_allocator *alloc = &pool->pool;
15017         nk_pool_init(&ctx->pool, alloc, NK_POOL_DEFAULT_CAPACITY);
15018     }
15019     ctx->use_pool = nk_true;
15020     return 1;
15021 }
15022 NK_API int
15023 nk_init(struct nk_context *ctx, struct nk_allocator *alloc,
15024     const struct nk_user_font *font)
15025 {
15026     NK_ASSERT(alloc);
15027     if (!alloc) return 0;
15028     nk_setup(ctx, font);
15029     nk_buffer_init(&ctx->memory, alloc, NK_DEFAULT_COMMAND_BUFFER_SIZE);
15030     nk_pool_init(&ctx->pool, alloc, NK_POOL_DEFAULT_CAPACITY);
15031     ctx->use_pool = nk_true;
15032     return 1;
15033 }
15034 #ifdef NK_INCLUDE_COMMAND_USERDATA
15035 NK_API void
15036 nk_set_user_data(struct nk_context *ctx, nk_handle handle)
15037 {
15038     if (!ctx) return;
15039     ctx->userdata = handle;
15040     if (ctx->current)
15041         ctx->current->buffer.userdata = handle;
15042 }
15043 #endif
15044 NK_API void
15045 nk_free(struct nk_context *ctx)
15046 {
15047     NK_ASSERT(ctx);
15048     if (!ctx) return;
15049     nk_buffer_free(&ctx->memory);
15050     if (ctx->use_pool)
15051         nk_pool_free(&ctx->pool);
15052
15053     nk_zero(&ctx->input, sizeof(ctx->input));
15054     nk_zero(&ctx->style, sizeof(ctx->style));
15055     nk_zero(&ctx->memory, sizeof(ctx->memory));
15056
15057     ctx->seq = 0;
15058     ctx->build = 0;
15059     ctx->begin = 0;
15060     ctx->end = 0;
15061     ctx->active = 0;
15062     ctx->current = 0;
15063     ctx->freelist = 0;
15064     ctx->count = 0;
15065 }
15066 NK_API void
15067 nk_clear(struct nk_context *ctx)
15068 {
15069     struct nk_window *iter;
15070     struct nk_window *next;
15071     NK_ASSERT(ctx);
15072
15073     if (!ctx) return;
15074     if (ctx->use_pool)
15075         nk_buffer_clear(&ctx->memory);
15076     else nk_buffer_reset(&ctx->memory, NK_BUFFER_FRONT);
15077
15078     ctx->build = 0;
15079     ctx->memory.calls = 0;
15080     ctx->last_widget_state = 0;
15081     ctx->style.cursor_active = ctx->style.cursors[NK_CURSOR_ARROW];
15082     NK_MEMSET(&ctx->overlay, 0, sizeof(ctx->overlay));
```

```
15083
15084     /* garbage collector */
15085     iter = ctx->begin;
15086     while (iter) {
15087         /* make sure valid minimized windows do not get removed */
15088         if ((iter->flags & NK_WINDOW_MINIMIZED) &&
15089             !(iter->flags & NK_WINDOW_CLOSED) &&
15090             iter->seq == ctx->seq) {
15091             iter = iter->next;
15092             continue;
15093         }
15094         /* remove hotness from hidden or closed windows*/
15095         if (((iter->flags & NK_WINDOW_HIDDEN) ||
15096             (iter->flags & NK_WINDOW_CLOSED)) &&
15097             iter == ctx->active) {
15098             ctx->active = iter->prev;
15099             ctx->end = iter->prev;
15100             if (!ctx->end)
15101                 ctx->begin = 0;
15102             if (ctx->active)
15103                 ctx->active->flags &= ~(unsigned)NK_WINDOW_ROM;
15104         }
15105         /* free unused popup windows */
15106         if (iter->popup.win && iter->popup.win->seq != ctx->seq) {
15107             nk_free_window(ctx, iter->popup.win);
15108             iter->popup.win = 0;
15109         }
15110         /* remove unused window state tables */
15111         {struct nk_table *n, *it = iter->tables;
15112         while (it) {
15113             n = it->next;
15114             if (it->seq != ctx->seq) {
15115                 nk_remove_table(iter, it);
15116                 nk_zero(it, sizeof(union nk_page_data));
15117                 nk_free_table(ctx, it);
15118                 if (it == iter->tables)
15119                     iter->tables = n;
15120             } it = n;
15121         }}
15122         /* window itself is not used anymore so free */
15123         if (iter->seq != ctx->seq || iter->flags & NK_WINDOW_CLOSED) {
15124             next = iter->next;
15125             nk_remove_window(ctx, iter);
15126             nk_free_window(ctx, iter);
15127             iter = next;
15128         } else iter = iter->next;
15129     }
15130     ctx->seq++;
15131 }
15132 NK_LIB void
15133 nk_start_buffer(struct nk_context *ctx, struct nk_command_buffer *buffer)
15134 {
15135     NK_ASSERT(ctx);
15136     NK_ASSERT(buffer);
15137     if (!ctx || !buffer) return;
15138     buffer->begin = ctx->memory.allocated;
15139     buffer->end = buffer->begin;
15140     buffer->last = buffer->begin;
15141     buffer->clip = nk_null_rect;
15142 }
15143 NK_LIB void
15144 nk_start(struct nk_context *ctx, struct nk_window *win)
15145 {
15146     NK_ASSERT(ctx);
15147     NK_ASSERT(win);
15148     nk_start_buffer(ctx, &win->buffer);
15149 }
15150 NK_LIB void
15151 nk_start_popup(struct nk_context *ctx, struct nk_window *win)
15152 {
15153     struct nk_popup_buffer *buf;
15154     NK_ASSERT(ctx);
15155     NK_ASSERT(win);
15156     if (!ctx || !win) return;
15157
15158     /* save buffer fill state for popup */
15159     buf = &win->popup.buf;
15160     buf->begin = win->buffer.end;
15161     buf->end = win->buffer.end;
15162     buf->parent = win->buffer.last;
15163     buf->last = buf->begin;
15164     buf->active = nk_true;
15165 }
15166 NK_LIB void
15167 nk_finish_popup(struct nk_context *ctx, struct nk_window *win)
15168 {
15169     struct nk_popup_buffer *buf;
```

```
15170     NK_ASSERT(ctx);
15171     NK_ASSERT(win);
15172     if (!ctx || !win) return;
15173
15174     buf = &win->popup.buf;
15175     buf->last = win->buffer.last;
15176     buf->end = win->buffer.end;
15177 }
15178 NK_LIB void
15179 nk_finish_buffer(struct nk_context *ctx, struct nk_command_buffer *buffer)
15180 {
15181     NK_ASSERT(ctx);
15182     NK_ASSERT(buffer);
15183     if (!ctx || !buffer) return;
15184     buffer->end = ctx->memory.allocated;
15185 }
15186 NK_LIB void
15187 nk_finish(struct nk_context *ctx, struct nk_window *win)
15188 {
15189     struct nk_popup_buffer *buf;
15190     struct nk_command *parent_last;
15191     void *memory;
15192
15193     NK_ASSERT(ctx);
15194     NK_ASSERT(win);
15195     if (!ctx || !win) return;
15196     nk_finish_buffer(ctx, &win->buffer);
15197     if (!win->popup.buf.active) return;
15198
15199     buf = &win->popup.buf;
15200     memory = ctx->memory.memory.ptr;
15201     parent_last = nk_ptr_add(struct nk_command, memory, buf->parent);
15202     parent_last->next = buf->end;
15203 }
15204 NK_LIB void
15205 nk_build(struct nk_context *ctx)
15206 {
15207     struct nk_window *it = 0;
15208     struct nk_command *cmd = 0;
15209     nk_byte *buffer = 0;
15210
15211     /* draw cursor overlay */
15212     if (!ctx->style.cursor_active)
15213         ctx->style.cursor_active = ctx->style.cursors[NK_CURSOR_ARROW];
15214     if (ctx->style.cursor_active && !ctx->input.mouse.grabbed && ctx->style.cursor_visible) {
15215         struct nk_rect mouse_bounds;
15216         const struct nk_cursor *cursor = ctx->style.cursor_active;
15217         nk_command_buffer_init(&ctx->overlay, &ctx->memory, NK_CLIPPING_OFF);
15218         nk_start_buffer(ctx, &ctx->overlay);
15219
15220         mouse_bounds.x = ctx->input.mouse.pos.x - cursor->offset.x;
15221         mouse_bounds.y = ctx->input.mouse.pos.y - cursor->offset.y;
15222         mouse_bounds.w = cursor->size.x;
15223         mouse_bounds.h = cursor->size.y;
15224
15225         nk_draw_image(&ctx->overlay, mouse_bounds, &cursor->img, nk_white);
15226         nk_finish_buffer(ctx, &ctx->overlay);
15227     }
15228     /* build one big draw command list out of all window buffers */
15229     it = ctx->begin;
15230     buffer = (nk_byte*)ctx->memory.memory.ptr;
15231     while (it != 0) {
15232         struct nk_window *next = it->next;
15233         if (it->buffer.last == it->buffer.begin || (it->flags & NK_WINDOW_HIDDEN)||
15234             it->seq != ctx->seq)
15235             goto cont;
15236
15237         cmd = nk_ptr_add(struct nk_command, buffer, it->buffer.last);
15238         while (next && ((next->buffer.last == next->buffer.begin) ||
15239             (next->flags & NK_WINDOW_HIDDEN) || next->seq != ctx->seq))
15240             next = next->next; /* skip empty command buffers */
15241
15242         if (next) cmd->next = next->buffer.begin;
15243         cont: it = next;
15244     }
15245     /* append all popup draw commands into lists */
15246     it = ctx->begin;
15247     while (it != 0) {
15248         struct nk_window *next = it->next;
15249         struct nk_popup_buffer *buf;
15250         if (!it->popup.buf.active)
15251             goto skip;
15252
15253         buf = &it->popup.buf;
15254         cmd->next = buf->begin;
15255         cmd = nk_ptr_add(struct nk_command, buffer, buf->last);
15256         buf->active = nk_false;
```

```
15257         skip: it = next;
15258     }
15259     if (cmd) {
15260         /* append overlay commands */
15261         if (ctx->overlay.end != ctx->overlay.begin)
15262             cmd->next = ctx->overlay.begin;
15263         else cmd->next = ctx->memory.allocated;
15264     }
15265 }
15266 NK_API const struct nk_command*
15267 nk__begin(struct nk_context *ctx)
15268 {
15269     struct nk_window *iter;
15270     nk_byte *buffer;
15271     NK_ASSERT(ctx);
15272     if (!ctx) return 0;
15273     if (!ctx->count) return 0;
15274
15275     buffer = (nk_byte*)ctx->memory.memory.ptr;
15276     if (!ctx->build) {
15277         nk_build(ctx);
15278         ctx->build = nk_true;
15279     }
15280     iter = ctx->begin;
15281     while (iter && ((iter->buffer.begin == iter->buffer.end) ||
15282         (iter->flags & NK_WINDOW_HIDDEN) || iter->seq != ctx->seq))
15283         iter = iter->next;
15284     if (!iter) return 0;
15285     return nk_ptr_add_const(struct nk_command, buffer, iter->buffer.begin);
15286 }
15287
15288 NK_API const struct nk_command*
15289 nk__next(struct nk_context *ctx, const struct nk_command *cmd)
15290 {
15291     nk_byte *buffer;
15292     const struct nk_command *next;
15293     NK_ASSERT(ctx);
15294     if (!ctx || !cmd || !ctx->count) return 0;
15295     if (cmd->next >= ctx->memory.allocated) return 0;
15296     buffer = (nk_byte*)ctx->memory.memory.ptr;
15297     next = nk_ptr_add_const(struct nk_command, buffer, cmd->next);
15298     return next;
15299 }
15300
15301
15302
15303
15304
15305
15306 /* ===============================================================
15307  *
15308  *                              POOL
15309  *
15310  * ===============================================================*/
15311 NK_LIB void
15312 nk_pool_init(struct nk_pool *pool, struct nk_allocator *alloc,
15313     unsigned int capacity)
15314 {
15315     nk_zero(pool, sizeof(*pool));
15316     pool->alloc = *alloc;
15317     pool->capacity = capacity;
15318     pool->type = NK_BUFFER_DYNAMIC;
15319     pool->pages = 0;
15320 }
15321 NK_LIB void
15322 nk_pool_free(struct nk_pool *pool)
15323 {
15324     struct nk_page *iter = pool->pages;
15325     if (!pool) return;
15326     if (pool->type == NK_BUFFER_FIXED) return;
15327     while (iter) {
15328         struct nk_page *next = iter->next;
15329         pool->alloc.free(pool->alloc.userdata, iter);
15330         iter = next;
15331     }
15332 }
15333 NK_LIB void
15334 nk_pool_init_fixed(struct nk_pool *pool, void *memory, nk_size size)
15335 {
15336     nk_zero(pool, sizeof(*pool));
15337     NK_ASSERT(size >= sizeof(struct nk_page));
15338     if (size < sizeof(struct nk_page)) return;
15339     pool->capacity = (unsigned)(size - sizeof(struct nk_page)) / sizeof(struct nk_page_element);
15340     pool->pages = (struct nk_page*)memory;
15341     pool->type = NK_BUFFER_FIXED;
15342     pool->size = size;
15343 }
```

```
15344 NK_LIB struct nk_page_element*
15345 nk_pool_alloc(struct nk_pool *pool)
15346 {
15347     if (!pool->pages || pool->pages->size >= pool->capacity) {
15348         /* allocate new page */
15349         struct nk_page *page;
15350         if (pool->type == NK_BUFFER_FIXED) {
15351             NK_ASSERT(pool->pages);
15352             if (!pool->pages) return 0;
15353             NK_ASSERT(pool->pages->size < pool->capacity);
15354             return 0;
15355         } else {
15356             nk_size size = sizeof(struct nk_page);
15357             size += NK_POOL_DEFAULT_CAPACITY * sizeof(union nk_page_data);
15358             page = (struct nk_page*)pool->alloc.alloc(pool->alloc.userdata,0, size);
15359             page->next = pool->pages;
15360             pool->pages = page;
15361             page->size = 0;
15362         }
15363     } return &pool->pages->win[pool->pages->size++];
15364 }
15365
15366
15367
15368
15369
15370 /* ===============================================================
15371  *
15372  *                          PAGE ELEMENT
15373  *
15374  * ===============================================================*/
15375 NK_LIB struct nk_page_element*
15376 nk_create_page_element(struct nk_context *ctx)
15377 {
15378     struct nk_page_element *elem;
15379     if (ctx->freelist) {
15380         /* unlink page element from free list */
15381         elem = ctx->freelist;
15382         ctx->freelist = elem->next;
15383     } else if (ctx->use_pool) {
15384         /* allocate page element from memory pool */
15385         elem = nk_pool_alloc(&ctx->pool);
15386         NK_ASSERT(elem);
15387         if (!elem) return 0;
15388     } else {
15389         /* allocate new page element from back of fixed size memory buffer */
15390         NK_STORAGE const nk_size size = sizeof(struct nk_page_element);
15391         NK_STORAGE const nk_size align = NK_ALIGNOF(struct nk_page_element);
15392         elem = (struct nk_page_element*)nk_buffer_alloc(&ctx->memory, NK_BUFFER_BACK, size, align);
15393         NK_ASSERT(elem);
15394         if (!elem) return 0;
15395     }
15396     nk_zero_struct(*elem);
15397     elem->next = 0;
15398     elem->prev = 0;
15399     return elem;
15400 }
15401 NK_LIB void
15402 nk_link_page_element_into_freelist(struct nk_context *ctx,
15403     struct nk_page_element *elem)
15404 {
15405     /* link table into freelist */
15406     if (!ctx->freelist) {
15407         ctx->freelist = elem;
15408     } else {
15409         elem->next = ctx->freelist;
15410         ctx->freelist = elem;
15411     }
15412 }
15413 NK_LIB void
15414 nk_free_page_element(struct nk_context *ctx, struct nk_page_element *elem)
15415 {
15416     /* we have a pool so just add to free list */
15417     if (ctx->use_pool) {
15418         nk_link_page_element_into_freelist(ctx, elem);
15419         return;
15420     }
15421     /* if possible remove last element from back of fixed memory buffer */
15422     {void *elem_end = (void*)(elem + 1);
15423     void *buffer_end = (nk_byte*)ctx->memory.memory.ptr + ctx->memory.size;
15424     if (elem_end == buffer_end)
15425         ctx->memory.size -= sizeof(struct nk_page_element);
15426     else nk_link_page_element_into_freelist(ctx, elem);}
15427 }
15428
15429
15430
```

```
15431
15432
15433 /* ===============================================================
15434  *
15435  *                              TABLE
15436  *
15437  * ===============================================================*/
15438 NK_LIB struct nk_table*
15439 nk_create_table(struct nk_context *ctx)
15440 {
15441     struct nk_page_element *elem;
15442     elem = nk_create_page_element(ctx);
15443     if (!elem) return 0;
15444     nk_zero_struct(*elem);
15445     return &elem->data.tbl;
15446 }
15447 NK_LIB void
15448 nk_free_table(struct nk_context *ctx, struct nk_table *tbl)
15449 {
15450     union nk_page_data *pd = NK_CONTAINER_OF(tbl, union nk_page_data, tbl);
15451     struct nk_page_element *pe = NK_CONTAINER_OF(pd, struct nk_page_element, data);
15452     nk_free_page_element(ctx, pe);
15453 }
15454 NK_LIB void
15455 nk_push_table(struct nk_window *win, struct nk_table *tbl)
15456 {
15457     if (!win->tables) {
15458         win->tables = tbl;
15459         tbl->next = 0;
15460         tbl->prev = 0;
15461         tbl->size = 0;
15462         win->table_count = 1;
15463         return;
15464     }
15465     win->tables->prev = tbl;
15466     tbl->next = win->tables;
15467     tbl->prev = 0;
15468     tbl->size = 0;
15469     win->tables = tbl;
15470     win->table_count++;
15471 }
15472 NK_LIB void
15473 nk_remove_table(struct nk_window *win, struct nk_table *tbl)
15474 {
15475     if (win->tables == tbl)
15476         win->tables = tbl->next;
15477     if (tbl->next)
15478         tbl->next->prev = tbl->prev;
15479     if (tbl->prev)
15480         tbl->prev->next = tbl->next;
15481     tbl->next = 0;
15482     tbl->prev = 0;
15483 }
15484 NK_LIB nk_uint*
15485 nk_add_value(struct nk_context *ctx, struct nk_window *win,
15486             nk_hash name, nk_uint value)
15487 {
15488     NK_ASSERT(ctx);
15489     NK_ASSERT(win);
15490     if (!win || !ctx) return 0;
15491     if (!win->tables || win->tables->size >= NK_VALUE_PAGE_CAPACITY) {
15492         struct nk_table *tbl = nk_create_table(ctx);
15493         NK_ASSERT(tbl);
15494         if (!tbl) return 0;
15495         nk_push_table(win, tbl);
15496     }
15497     win->tables->seq = win->seq;
15498     win->tables->keys[win->tables->size] = name;
15499     win->tables->values[win->tables->size] = value;
15500     return &win->tables->values[win->tables->size++];
15501 }
15502 NK_LIB nk_uint*
15503 nk_find_value(struct nk_window *win, nk_hash name)
15504 {
15505     struct nk_table *iter = win->tables;
15506     while (iter) {
15507         unsigned int i = 0;
15508         unsigned int size = iter->size;
15509         for (i = 0; i < size; ++i) {
15510             if (iter->keys[i] == name) {
15511                 iter->seq = win->seq;
15512                 return &iter->values[i];
15513             }
15514         } size = NK_VALUE_PAGE_CAPACITY;
15515         iter = iter->next;
15516     }
15517     return 0;
```

```
15518 }
15519
15520
15521
15522
15523
15524 /* ===============================================================
15525  *
15526  *                              PANEL
15527  *
15528  * ===============================================================*/
15529 NK_LIB void*
15530 nk_create_panel(struct nk_context *ctx)
15531 {
15532     struct nk_page_element *elem;
15533     elem = nk_create_page_element(ctx);
15534     if (!elem) return 0;
15535     nk_zero_struct(*elem);
15536     return &elem->data.pan;
15537 }
15538 NK_LIB void
15539 nk_free_panel(struct nk_context *ctx, struct nk_panel *pan)
15540 {
15541     union nk_page_data *pd = NK_CONTAINER_OF(pan, union nk_page_data, pan);
15542     struct nk_page_element *pe = NK_CONTAINER_OF(pd, struct nk_page_element, data);
15543     nk_free_page_element(ctx, pe);
15544 }
15545 NK_LIB int
15546 nk_panel_has_header(nk_flags flags, const char *title)
15547 {
15548     int active = 0;
15549     active = (flags & (NK_WINDOW_CLOSABLE|NK_WINDOW_MINIMIZABLE));
15550     active = active || (flags & NK_WINDOW_TITLE);
15551     active = active && !(flags & NK_WINDOW_HIDDEN) && title;
15552     return active;
15553 }
15554 NK_LIB struct nk_vec2
15555 nk_panel_get_padding(const struct nk_style *style, enum nk_panel_type type)
15556 {
15557     switch (type) {
15558     default:
15559     case NK_PANEL_WINDOW: return style->window.padding;
15560     case NK_PANEL_GROUP: return style->window.group_padding;
15561     case NK_PANEL_POPUP: return style->window.popup_padding;
15562     case NK_PANEL_CONTEXTUAL: return style->window.contextual_padding;
15563     case NK_PANEL_COMBO: return style->window.combo_padding;
15564     case NK_PANEL_MENU: return style->window.menu_padding;
15565     case NK_PANEL_TOOLTIP: return style->window.menu_padding;}
15566 }
15567 NK_LIB float
15568 nk_panel_get_border(const struct nk_style *style, nk_flags flags,
15569     enum nk_panel_type type)
15570 {
15571     if (flags & NK_WINDOW_BORDER) {
15572         switch (type) {
15573         default:
15574         case NK_PANEL_WINDOW: return style->window.border;
15575         case NK_PANEL_GROUP: return style->window.group_border;
15576         case NK_PANEL_POPUP: return style->window.popup_border;
15577         case NK_PANEL_CONTEXTUAL: return style->window.contextual_border;
15578         case NK_PANEL_COMBO: return style->window.combo_border;
15579         case NK_PANEL_MENU: return style->window.menu_border;
15580         case NK_PANEL_TOOLTIP: return style->window.menu_border;
15581     }} else return 0;
15582 }
15583 NK_LIB struct nk_color
15584 nk_panel_get_border_color(const struct nk_style *style, enum nk_panel_type type)
15585 {
15586     switch (type) {
15587     default:
15588     case NK_PANEL_WINDOW: return style->window.border_color;
15589     case NK_PANEL_GROUP: return style->window.group_border_color;
15590     case NK_PANEL_POPUP: return style->window.popup_border_color;
15591     case NK_PANEL_CONTEXTUAL: return style->window.contextual_border_color;
15592     case NK_PANEL_COMBO: return style->window.combo_border_color;
15593     case NK_PANEL_MENU: return style->window.menu_border_color;
15594     case NK_PANEL_TOOLTIP: return style->window.menu_border_color;}
15595 }
15596 NK_LIB int
15597 nk_panel_is_sub(enum nk_panel_type type)
15598 {
15599     return (type & NK_PANEL_SET_SUB)?1:0;
15600 }
15601 NK_LIB int
15602 nk_panel_is_nonblock(enum nk_panel_type type)
15603 {
15604     return (type & NK_PANEL_SET_NONBLOCK)?1:0;
```

```
15605 }
15606 NK_LIB int
15607 nk_panel_begin(struct nk_context *ctx, const char *title, enum nk_panel_type panel_type)
15608 {
15609     struct nk_input *in;
15610     struct nk_window *win;
15611     struct nk_panel *layout;
15612     struct nk_command_buffer *out;
15613     const struct nk_style *style;
15614     const struct nk_user_font *font;
15615
15616     struct nk_vec2 scrollbar_size;
15617     struct nk_vec2 panel_padding;
15618
15619     NK_ASSERT(ctx);
15620     NK_ASSERT(ctx->current);
15621     NK_ASSERT(ctx->current->layout);
15622     if (!ctx || !ctx->current || !ctx->current->layout) return 0;
15623     nk_zero(ctx->current->layout, sizeof(*ctx->current->layout));
15624     if ((ctx->current->flags & NK_WINDOW_HIDDEN) || (ctx->current->flags & NK_WINDOW_CLOSED)) {
15625         nk_zero(ctx->current->layout, sizeof(struct nk_panel));
15626         ctx->current->layout->type = panel_type;
15627         return 0;
15628     }
15629     /* pull state into local stack */
15630     style = &ctx->style;
15631     font = style->font;
15632     win = ctx->current;
15633     layout = win->layout;
15634     out = &win->buffer;
15635     in = (win->flags & NK_WINDOW_NO_INPUT) ? 0: &ctx->input;
15636 #ifdef NK_INCLUDE_COMMAND_USERDATA
15637     win->buffer.userdata = ctx->userdata;
15638 #endif
15639     /* pull style configuration into local stack */
15640     scrollbar_size = style->window.scrollbar_size;
15641     panel_padding = nk_panel_get_padding(style, panel_type);
15642
15643     /* window movement */
15644     if ((win->flags & NK_WINDOW_MOVABLE) && !(win->flags & NK_WINDOW_ROM)) {
15645         int left_mouse_down;
15646         int left_mouse_clicked;
15647         int left_mouse_click_in_cursor;
15648
15649         /* calculate draggable window space */
15650         struct nk_rect header;
15651         header.x = win->bounds.x;
15652         header.y = win->bounds.y;
15653         header.w = win->bounds.w;
15654         if (nk_panel_has_header(win->flags, title)) {
15655             header.h = font->height + 2.0f * style->window.header.padding.y;
15656             header.h += 2.0f * style->window.header.label_padding.y;
15657         } else header.h = panel_padding.y;
15658
15659         /* window movement by dragging */
15660         left_mouse_down = in->mouse.buttons[NK_BUTTON_LEFT].down;
15661         left_mouse_clicked = (int)in->mouse.buttons[NK_BUTTON_LEFT].clicked;
15662         left_mouse_click_in_cursor = nk_input_has_mouse_click_down_in_rect(in,
15663             NK_BUTTON_LEFT, header, nk_true);
15664         if (left_mouse_down && left_mouse_click_in_cursor && !left_mouse_clicked) {
15665             win->bounds.x = win->bounds.x + in->mouse.delta.x;
15666             win->bounds.y = win->bounds.y + in->mouse.delta.y;
15667             in->mouse.buttons[NK_BUTTON_LEFT].clicked_pos.x += in->mouse.delta.x;
15668             in->mouse.buttons[NK_BUTTON_LEFT].clicked_pos.y += in->mouse.delta.y;
15669             ctx->style.cursor_active = ctx->style.cursors[NK_CURSOR_MOVE];
15670         }
15671     }
15672
15673     /* setup panel */
15674     layout->type = panel_type;
15675     layout->flags = win->flags;
15676     layout->bounds = win->bounds;
15677     layout->bounds.x += panel_padding.x;
15678     layout->bounds.w -= 2*panel_padding.x;
15679     if (win->flags & NK_WINDOW_BORDER) {
15680         layout->border = nk_panel_get_border(style, win->flags, panel_type);
15681         layout->bounds = nk_shrink_rect(layout->bounds, layout->border);
15682     } else layout->border = 0;
15683     layout->at_y = layout->bounds.y;
15684     layout->at_x = layout->bounds.x;
15685     layout->max_x = 0;
15686     layout->header_height = 0;
15687     layout->footer_height = 0;
15688     nk_layout_reset_min_row_height(ctx);
15689     layout->row.index = 0;
15690     layout->row.columns = 0;
15691     layout->row.ratio = 0;
```

```
15692     layout->row.item_width = 0;
15693     layout->row.tree_depth = 0;
15694     layout->row.height = panel_padding.y;
15695     layout->has_scrolling = nk_true;
15696     if (!(win->flags & NK_WINDOW_NO_SCROLLBAR))
15697         layout->bounds.w -= scrollbar_size.x;
15698     if (!nk_panel_is_nonblock(panel_type)) {
15699         layout->footer_height = 0;
15700         if (!(win->flags & NK_WINDOW_NO_SCROLLBAR) || win->flags & NK_WINDOW_SCALABLE)
15701             layout->footer_height = scrollbar_size.y;
15702         layout->bounds.h -= layout->footer_height;
15703     }
15704
15705     /* panel header */
15706     if (nk_panel_has_header(win->flags, title))
15707     {
15708         struct nk_text text;
15709         struct nk_rect header;
15710         const struct nk_style_item *background = 0;
15711
15712         /* calculate header bounds */
15713         header.x = win->bounds.x;
15714         header.y = win->bounds.y;
15715         header.w = win->bounds.w;
15716         header.h = font->height + 2.0f * style->window.header.padding.y;
15717         header.h += (2.0f * style->window.header.label_padding.y);
15718
15719         /* shrink panel by header */
15720         layout->header_height = header.h;
15721         layout->bounds.y += header.h;
15722         layout->bounds.h -= header.h;
15723         layout->at_y += header.h;
15724
15725         /* select correct header background and text color */
15726         if (ctx->active == win) {
15727             background = &style->window.header.active;
15728             text.text = style->window.header.label_active;
15729         } else if (nk_input_is_mouse_hovering_rect(&ctx->input, header)) {
15730             background = &style->window.header.hover;
15731             text.text = style->window.header.label_hover;
15732         } else {
15733             background = &style->window.header.normal;
15734             text.text = style->window.header.label_normal;
15735         }
15736
15737         /* draw header background */
15738         header.h += 1.0f;
15739         if (background->type == NK_STYLE_ITEM_IMAGE) {
15740             text.background = nk_rgba(0,0,0,0);
15741             nk_draw_image(&win->buffer, header, &background->data.image, nk_white);
15742         } else {
15743             text.background = background->data.color;
15744             nk_fill_rect(out, header, 0, background->data.color);
15745         }
15746
15747         /* window close button */
15748         {struct nk_rect button;
15749         button.y = header.y + style->window.header.padding.y;
15750         button.h = header.h - 2 * style->window.header.padding.y;
15751         button.w = button.h;
15752         if (win->flags & NK_WINDOW_CLOSABLE) {
15753             nk_flags ws = 0;
15754             if (style->window.header.align == NK_HEADER_RIGHT) {
15755                 button.x = (header.w + header.x) - (button.w + style->window.header.padding.x);
15756                 header.w -= button.w + style->window.header.spacing.x +
      style->window.header.padding.x;
15757             } else {
15758                 button.x = header.x + style->window.header.padding.x;
15759                 header.x += button.w + style->window.header.spacing.x +
      style->window.header.padding.x;
15760             }
15761
15762             if (nk_do_button_symbol(&ws, &win->buffer, button,
15763                 style->window.header.close_symbol, NK_BUTTON_DEFAULT,
15764                 &style->window.header.close_button, in, style->font) && !(win->flags & NK_WINDOW_ROM))
15765             {
15766                 layout->flags |= NK_WINDOW_HIDDEN;
15767                 layout->flags &= (nk_flags)~NK_WINDOW_MINIMIZED;
15768             }
15769         }
15770
15771         /* window minimize button */
15772         if (win->flags & NK_WINDOW_MINIMIZABLE) {
15773             nk_flags ws = 0;
15774             if (style->window.header.align == NK_HEADER_RIGHT) {
15775                 button.x = (header.w + header.x) - button.w;
15776                 if (!(win->flags & NK_WINDOW_CLOSABLE)) {
```

```
15777                   button.x -= style->window.header.padding.x;
15778                   header.w -= style->window.header.padding.x;
15779               }
15780               header.w -= button.w + style->window.header.spacing.x;
15781           } else {
15782               button.x = header.x;
15783               header.x += button.w + style->window.header.spacing.x +
      style->window.header.padding.x;
15784           }
15785           if (nk_do_button_symbol(&ws, &win->buffer, button, (layout->flags & NK_WINDOW_MINIMIZED)?
15786               style->window.header.maximize_symbol: style->window.header.minimize_symbol,
15787               NK_BUTTON_DEFAULT, &style->window.header.minimize_button, in, style->font) &&
      !(win->flags & NK_WINDOW_ROM))
15788               layout->flags = (layout->flags & NK_WINDOW_MINIMIZED) ?
15789                   layout->flags & (nk_flags)~NK_WINDOW_MINIMIZED:
15790                   layout->flags | NK_WINDOW_MINIMIZED;
15791       }}
15792
15793       {/* window header title */
15794       int text_len = nk_strlen(title);
15795       struct nk_rect label = {0,0,0,0};
15796       float t = font->width(font->userdata, font->height, title, text_len);
15797       text.padding = nk_vec2(0,0);
15798
15799       label.x = header.x + style->window.header.padding.x;
15800       label.x += style->window.header.label_padding.x;
15801       label.y = header.y + style->window.header.label_padding.y;
15802       label.h = font->height + 2 * style->window.header.label_padding.y;
15803       label.w = t + 2 * style->window.header.spacing.x;
15804       label.w = NK_CLAMP(0, label.w, header.x + header.w - label.x);
15805       nk_widget_text(out, label,(const char*)title, text_len, &text, NK_TEXT_LEFT, font);}
15806   }
15807
15808   /* draw window background */
15809   if (!(layout->flags & NK_WINDOW_MINIMIZED) && !(layout->flags & NK_WINDOW_DYNAMIC)) {
15810       struct nk_rect body;
15811       body.x = win->bounds.x;
15812       body.w = win->bounds.w;
15813       body.y = (win->bounds.y + layout->header_height);
15814       body.h = (win->bounds.h - layout->header_height);
15815       if (style->window.fixed_background.type == NK_STYLE_ITEM_IMAGE)
15816           nk_draw_image(out, body, &style->window.fixed_background.data.image, nk_white);
15817       else nk_fill_rect(out, body, 0, style->window.fixed_background.data.color);
15818   }
15819
15820   /* set clipping rectangle */
15821   {struct nk_rect clip;
15822   layout->clip = layout->bounds;
15823   nk_unify(&clip, &win->buffer.clip, layout->clip.x, layout->clip.y,
15824       layout->clip.x + layout->clip.w, layout->clip.y + layout->clip.h);
15825   nk_push_scissor(out, clip);
15826   layout->clip = clip;}
15827   return !(layout->flags & NK_WINDOW_HIDDEN) && !(layout->flags & NK_WINDOW_MINIMIZED);
15828 }
15829 NK_LIB void
15830 nk_panel_end(struct nk_context *ctx)
15831 {
15832     struct nk_input *in;
15833     struct nk_window *window;
15834     struct nk_panel *layout;
15835     const struct nk_style *style;
15836     struct nk_command_buffer *out;
15837
15838     struct nk_vec2 scrollbar_size;
15839     struct nk_vec2 panel_padding;
15840
15841     NK_ASSERT(ctx);
15842     NK_ASSERT(ctx->current);
15843     NK_ASSERT(ctx->current->layout);
15844     if (!ctx || !ctx->current || !ctx->current->layout)
15845         return;
15846
15847     window = ctx->current;
15848     layout = window->layout;
15849     style = &ctx->style;
15850     out = &window->buffer;
15851     in = (layout->flags & NK_WINDOW_ROM || layout->flags & NK_WINDOW_NO_INPUT) ? 0 :&ctx->input;
15852     if (!nk_panel_is_sub(layout->type))
15853         nk_push_scissor(out, nk_null_rect);
15854
15855     /* cache configuration data */
15856     scrollbar_size = style->window.scrollbar_size;
15857     panel_padding = nk_panel_get_padding(style, layout->type);
15858
15859     /* update the current cursor Y-position to point over the last added widget */
15860     layout->at_y += layout->row.height;
15861
```

```
15862     /* dynamic panels */
15863     if (layout->flags & NK_WINDOW_DYNAMIC && !(layout->flags & NK_WINDOW_MINIMIZED))
15864     {
15865         /* update panel height to fit dynamic growth */
15866         struct nk_rect empty_space;
15867         if (layout->at_y < (layout->bounds.y + layout->bounds.h))
15868             layout->bounds.h = layout->at_y - layout->bounds.y;
15869
15870         /* fill top empty space */
15871         empty_space.x = window->bounds.x;
15872         empty_space.y = layout->bounds.y;
15873         empty_space.h = panel_padding.y;
15874         empty_space.w = window->bounds.w;
15875         nk_fill_rect(out, empty_space, 0, style->window.background);
15876
15877         /* fill left empty space */
15878         empty_space.x = window->bounds.x;
15879         empty_space.y = layout->bounds.y;
15880         empty_space.w = panel_padding.x + layout->border;
15881         empty_space.h = layout->bounds.h;
15882         nk_fill_rect(out, empty_space, 0, style->window.background);
15883
15884         /* fill right empty space */
15885         empty_space.x = layout->bounds.x + layout->bounds.w;
15886         empty_space.y = layout->bounds.y;
15887         empty_space.w = panel_padding.x + layout->border;
15888         empty_space.h = layout->bounds.h;
15889         if (*layout->offset_y == 0 && !(layout->flags & NK_WINDOW_NO_SCROLLBAR))
15890             empty_space.w += scrollbar_size.x;
15891         nk_fill_rect(out, empty_space, 0, style->window.background);
15892
15893         /* fill bottom empty space */
15894         if (layout->footer_height > 0) {
15895             empty_space.x = window->bounds.x;
15896             empty_space.y = layout->bounds.y + layout->bounds.h;
15897             empty_space.w = window->bounds.w;
15898             empty_space.h = layout->footer_height;
15899             nk_fill_rect(out, empty_space, 0, style->window.background);
15900         }
15901     }
15902
15903     /* scrollbars */
15904     if (!(layout->flags & NK_WINDOW_NO_SCROLLBAR) &&
15905         !(layout->flags & NK_WINDOW_MINIMIZED) &&
15906         window->scrollbar_hiding_timer < NK_SCROLLBAR_HIDING_TIMEOUT)
15907     {
15908         struct nk_rect scroll;
15909         int scroll_has_scrolling;
15910         float scroll_target;
15911         float scroll_offset;
15912         float scroll_step;
15913         float scroll_inc;
15914
15915         /* mouse wheel scrolling */
15916         if (nk_panel_is_sub(layout->type))
15917         {
15918             /* sub-window mouse wheel scrolling */
15919             struct nk_window *root_window = window;
15920             struct nk_panel *root_panel = window->layout;
15921             while (root_panel->parent)
15922                 root_panel = root_panel->parent;
15923             while (root_window->parent)
15924                 root_window = root_window->parent;
15925
15926             /* only allow scrolling if parent window is active */
15927             scroll_has_scrolling = 0;
15928             if ((root_window == ctx->active) && layout->has_scrolling) {
15929                 /* and panel is being hovered and inside clip rect*/
15930                 if (nk_input_is_mouse_hovering_rect(in, layout->bounds) &&
15931                     NK_INTERSECT(layout->bounds.x, layout->bounds.y, layout->bounds.w,
    layout->bounds.h,
15932                         root_panel->clip.x, root_panel->clip.y, root_panel->clip.w,
    root_panel->clip.h))
15933                 {
15934                     /* deactivate all parent scrolling */
15935                     root_panel = window->layout;
15936                     while (root_panel->parent) {
15937                         root_panel->has_scrolling = nk_false;
15938                         root_panel = root_panel->parent;
15939                     }
15940                     root_panel->has_scrolling = nk_false;
15941                     scroll_has_scrolling = nk_true;
15942                 }
15943             }
15944         } else if (!nk_panel_is_sub(layout->type)) {
15945             /* window mouse wheel scrolling */
15946             scroll_has_scrolling = (window == ctx->active) && layout->has_scrolling;
```

```
15947             if (in && (in->mouse.scroll_delta.y > 0 || in->mouse.scroll_delta.x > 0) &&
      scroll_has_scrolling)
15948                 window->scrolled = nk_true;
15949             else window->scrolled = nk_false;
15950         } else scroll_has_scrolling = nk_false;
15951
15952         {
15953             /* vertical scrollbar */
15954             nk_flags state = 0;
15955             scroll.x = layout->bounds.x + layout->bounds.w + panel_padding.x;
15956             scroll.y = layout->bounds.y;
15957             scroll.w = scrollbar_size.x;
15958             scroll.h = layout->bounds.h;
15959
15960             scroll_offset = (float)*layout->offset_y;
15961             scroll_step = scroll.h * 0.10f;
15962             scroll_inc = scroll.h * 0.01f;
15963             scroll_target = (float)(int)(layout->at_y - scroll.y);
15964             scroll_offset = nk_do_scrollbarv(&state, out, scroll, scroll_has_scrolling,
15965                 scroll_offset, scroll_target, scroll_step, scroll_inc,
15966                 &ctx->style.scrollv, in, style->font);
15967             *layout->offset_y = (nk_uint)scroll_offset;
15968             if (in && scroll_has_scrolling)
15969                 in->mouse.scroll_delta.y = 0;
15970         }
15971         {
15972             /* horizontal scrollbar */
15973             nk_flags state = 0;
15974             scroll.x = layout->bounds.x;
15975             scroll.y = layout->bounds.y + layout->bounds.h;
15976             scroll.w = layout->bounds.w;
15977             scroll.h = scrollbar_size.y;
15978
15979             scroll_offset = (float)*layout->offset_x;
15980             scroll_target = (float)(int)(layout->max_x - scroll.x);
15981             scroll_step = layout->max_x * 0.05f;
15982             scroll_inc = layout->max_x * 0.005f;
15983             scroll_offset = nk_do_scrollbarh(&state, out, scroll, scroll_has_scrolling,
15984                 scroll_offset, scroll_target, scroll_step, scroll_inc,
15985                 &ctx->style.scrollh, in, style->font);
15986             *layout->offset_x = (nk_uint)scroll_offset;
15987         }
15988     }
15989
15990     /* hide scroll if no user input */
15991     if (window->flags & NK_WINDOW_SCROLL_AUTO_HIDE) {
15992         int has_input = ctx->input.mouse.delta.x != 0 || ctx->input.mouse.delta.y != 0 ||
      ctx->input.mouse.scroll_delta.y != 0;
15993         int is_window_hovered = nk_window_is_hovered(ctx);
15994         int any_item_active = (ctx->last_widget_state & NK_WIDGET_STATE_MODIFIED);
15995         if ((!has_input && is_window_hovered) || (!is_window_hovered && !any_item_active))
15996             window->scrollbar_hiding_timer += ctx->delta_time_seconds;
15997         else window->scrollbar_hiding_timer = 0;
15998     } else window->scrollbar_hiding_timer = 0;
15999
16000     /* window border */
16001     if (layout->flags & NK_WINDOW_BORDER)
16002     {
16003         struct nk_color border_color = nk_panel_get_border_color(style, layout->type);
16004         const float padding_y = (layout->flags & NK_WINDOW_MINIMIZED)
16005             ? (style->window.border + window->bounds.y + layout->header_height)
16006             : ((layout->flags & NK_WINDOW_DYNAMIC)
16007                 ? (layout->bounds.y + layout->bounds.h + layout->footer_height)
16008                 : (window->bounds.y + window->bounds.h));
16009         struct nk_rect b = window->bounds;
16010         b.h = padding_y - window->bounds.y;
16011         nk_stroke_rect(out, b, 0, layout->border, border_color);
16012     }
16013
16014     /* scaler */
16015     if ((layout->flags & NK_WINDOW_SCALABLE) && in && !(layout->flags & NK_WINDOW_MINIMIZED))
16016     {
16017         /* calculate scaler bounds */
16018         struct nk_rect scaler;
16019         scaler.w = scrollbar_size.x;
16020         scaler.h = scrollbar_size.y;
16021         scaler.y = layout->bounds.y + layout->bounds.h;
16022         if (layout->flags & NK_WINDOW_SCALE_LEFT)
16023             scaler.x = layout->bounds.x - panel_padding.x * 0.5f;
16024         else scaler.x = layout->bounds.x + layout->bounds.w + panel_padding.x;
16025         if (layout->flags & NK_WINDOW_NO_SCROLLBAR)
16026             scaler.x -= scaler.w;
16027
16028         /* draw scaler */
16029         {const struct nk_style_item *item = &style->window.scaler;
16030         if (item->type == NK_STYLE_ITEM_IMAGE)
16031             nk_draw_image(out, scaler, &item->data.image, nk_white);
```

```
16032         else {
16033             if (layout->flags & NK_WINDOW_SCALE_LEFT) {
16034                 nk_fill_triangle(out, scaler.x, scaler.y, scaler.x,
16035                     scaler.y + scaler.h, scaler.x + scaler.w,
16036                     scaler.y + scaler.h, item->data.color);
16037             } else {
16038                 nk_fill_triangle(out, scaler.x + scaler.w, scaler.y, scaler.x + scaler.w,
16039                     scaler.y + scaler.h, scaler.x, scaler.y + scaler.h, item->data.color);
16040             }
16041         }}
16042
16043         /* do window scaling */
16044         if (!(window->flags & NK_WINDOW_ROM)) {
16045             struct nk_vec2 window_size = style->window.min_size;
16046             int left_mouse_down = in->mouse.buttons[NK_BUTTON_LEFT].down;
16047             int left_mouse_click_in_scaler = nk_input_has_mouse_click_down_in_rect(in,
16048                     NK_BUTTON_LEFT, scaler, nk_true);
16049
16050             if (left_mouse_down && left_mouse_click_in_scaler) {
16051                 float delta_x = in->mouse.delta.x;
16052                 if (layout->flags & NK_WINDOW_SCALE_LEFT) {
16053                     delta_x = -delta_x;
16054                     window->bounds.x += in->mouse.delta.x;
16055                 }
16056                 /* dragging in x-direction  */
16057                 if (window->bounds.w + delta_x >= window_size.x) {
16058                     if ((delta_x < 0) || (delta_x > 0 && in->mouse.pos.x >= scaler.x)) {
16059                         window->bounds.w = window->bounds.w + delta_x;
16060                         scaler.x += in->mouse.delta.x;
16061                     }
16062                 }
16063                 /* dragging in y-direction (only possible if static window) */
16064                 if (!(layout->flags & NK_WINDOW_DYNAMIC)) {
16065                     if (window_size.y < window->bounds.h + in->mouse.delta.y) {
16066                         if ((in->mouse.delta.y < 0) || (in->mouse.delta.y > 0 && in->mouse.pos.y >=
    scaler.y)) {
16067                             window->bounds.h = window->bounds.h + in->mouse.delta.y;
16068                             scaler.y += in->mouse.delta.y;
16069                         }
16070                     }
16071                 }
16072                 ctx->style.cursor_active = ctx->style.cursors[NK_CURSOR_RESIZE_TOP_RIGHT_DOWN_LEFT];
16073                 in->mouse.buttons[NK_BUTTON_LEFT].clicked_pos.x = scaler.x + scaler.w/2.0f;
16074                 in->mouse.buttons[NK_BUTTON_LEFT].clicked_pos.y = scaler.y + scaler.h/2.0f;
16075             }
16076         }
16077     }
16078     if (!nk_panel_is_sub(layout->type)) {
16079         /* window is hidden so clear command buffer  */
16080         if (layout->flags & NK_WINDOW_HIDDEN)
16081             nk_command_buffer_reset(&window->buffer);
16082         /* window is visible and not tab */
16083         else nk_finish(ctx, window);
16084     }
16085
16086     /* NK_WINDOW_REMOVE_ROM flag was set so remove NK_WINDOW_ROM */
16087     if (layout->flags & NK_WINDOW_REMOVE_ROM) {
16088         layout->flags &= ~(nk_flags)NK_WINDOW_ROM;
16089         layout->flags &= ~(nk_flags)NK_WINDOW_REMOVE_ROM;
16090     }
16091     window->flags = layout->flags;
16092
16093     /* property garbage collector */
16094     if (window->property.active && window->property.old != window->property.seq &&
16095         window->property.active == window->property.prev) {
16096         nk_zero(&window->property, sizeof(window->property));
16097     } else {
16098         window->property.old = window->property.seq;
16099         window->property.prev = window->property.active;
16100         window->property.seq = 0;
16101     }
16102     /* edit garbage collector */
16103     if (window->edit.active && window->edit.old != window->edit.seq &&
16104        window->edit.active == window->edit.prev) {
16105         nk_zero(&window->edit, sizeof(window->edit));
16106     } else {
16107         window->edit.old = window->edit.seq;
16108         window->edit.prev = window->edit.active;
16109         window->edit.seq = 0;
16110     }
16111     /* contextual garbage collector */
16112     if (window->popup.active_con && window->popup.con_old != window->popup.con_count) {
16113         window->popup.con_count = 0;
16114         window->popup.con_old = 0;
16115         window->popup.active_con = 0;
16116     } else {
16117         window->popup.con_old = window->popup.con_count;
```

```
16118         window->popup.con_count = 0;
16119     }
16120     window->popup.combo_count = 0;
16121     /* helper to make sure you have a 'nk_tree_push' for every 'nk_tree_pop' */
16122     NK_ASSERT(!layout->row.tree_depth);
16123 }
16124
16125
16126
16127
16128
16129 /* ===============================================================
16130  *
16131  *                              WINDOW
16132  *
16133  * ===============================================================*/
16134 NK_LIB void*
16135 nk_create_window(struct nk_context *ctx)
16136 {
16137     struct nk_page_element *elem;
16138     elem = nk_create_page_element(ctx);
16139     if (!elem) return 0;
16140     elem->data.win.seq = ctx->seq;
16141     return &elem->data.win;
16142 }
16143 NK_LIB void
16144 nk_free_window(struct nk_context *ctx, struct nk_window *win)
16145 {
16146     /* unlink windows from list */
16147     struct nk_table *it = win->tables;
16148     if (win->popup.win) {
16149         nk_free_window(ctx, win->popup.win);
16150         win->popup.win = 0;
16151     }
16152     win->next = 0;
16153     win->prev = 0;
16154
16155     while (it) {
16156         /*free window state tables */
16157         struct nk_table *n = it->next;
16158         nk_remove_table(win, it);
16159         nk_free_table(ctx, it);
16160         if (it == win->tables)
16161             win->tables = n;
16162         it = n;
16163     }
16164
16165     /* link windows into freelist */
16166     {union nk_page_data *pd = NK_CONTAINER_OF(win, union nk_page_data, win);
16167     struct nk_page_element *pe = NK_CONTAINER_OF(pd, struct nk_page_element, data);
16168     nk_free_page_element(ctx, pe);}
16169 }
16170 NK_LIB struct nk_window*
16171 nk_find_window(struct nk_context *ctx, nk_hash hash, const char *name)
16172 {
16173     struct nk_window *iter;
16174     iter = ctx->begin;
16175     while (iter) {
16176         NK_ASSERT(iter != iter->next);
16177         if (iter->name == hash) {
16178             int max_len = nk_strlen(iter->name_string);
16179             if (!nk_stricmpn(iter->name_string, name, max_len))
16180                 return iter;
16181         }
16182         iter = iter->next;
16183     }
16184     return 0;
16185 }
16186 NK_LIB void
16187 nk_insert_window(struct nk_context *ctx, struct nk_window *win,
16188     enum nk_window_insert_location loc)
16189 {
16190     const struct nk_window *iter;
16191     NK_ASSERT(ctx);
16192     NK_ASSERT(win);
16193     if (!win || !ctx) return;
16194
16195     iter = ctx->begin;
16196     while (iter) {
16197         NK_ASSERT(iter != iter->next);
16198         NK_ASSERT(iter != win);
16199         if (iter == win) return;
16200         iter = iter->next;
16201     }
16202
16203     if (!ctx->begin) {
16204         win->next = 0;
```

```
16205         win->prev = 0;
16206         ctx->begin = win;
16207         ctx->end = win;
16208         ctx->count = 1;
16209         return;
16210     }
16211     if (loc == NK_INSERT_BACK) {
16212         struct nk_window *end;
16213         end = ctx->end;
16214         end->flags |= NK_WINDOW_ROM;
16215         end->next = win;
16216         win->prev = ctx->end;
16217         win->next = 0;
16218         ctx->end = win;
16219         ctx->active = ctx->end;
16220         ctx->end->flags &= ~(nk_flags)NK_WINDOW_ROM;
16221     } else {
16222         /*ctx->end->flags |= NK_WINDOW_ROM;*/
16223         ctx->begin->prev = win;
16224         win->next = ctx->begin;
16225         win->prev = 0;
16226         ctx->begin = win;
16227         ctx->begin->flags &= ~(nk_flags)NK_WINDOW_ROM;
16228     }
16229     ctx->count++;
16230 }
16231 NK_LIB void
16232 nk_remove_window(struct nk_context *ctx, struct nk_window *win)
16233 {
16234     if (win == ctx->begin || win == ctx->end) {
16235         if (win == ctx->begin) {
16236             ctx->begin = win->next;
16237             if (win->next)
16238                 win->next->prev = 0;
16239         }
16240         if (win == ctx->end) {
16241             ctx->end = win->prev;
16242             if (win->prev)
16243                 win->prev->next = 0;
16244         }
16245     } else {
16246         if (win->next)
16247             win->next->prev = win->prev;
16248         if (win->prev)
16249             win->prev->next = win->next;
16250     }
16251     if (win == ctx->active || !ctx->active) {
16252         ctx->active = ctx->end;
16253         if (ctx->end)
16254             ctx->end->flags &= ~(nk_flags)NK_WINDOW_ROM;
16255     }
16256     win->next = 0;
16257     win->prev = 0;
16258     ctx->count--;
16259 }
16260 NK_API int
16261 nk_begin(struct nk_context *ctx, const char *title,
16262     struct nk_rect bounds, nk_flags flags)
16263 {
16264     return nk_begin_titled(ctx, title, title, bounds, flags);
16265 }
16266 NK_API int
16267 nk_begin_titled(struct nk_context *ctx, const char *name, const char *title,
16268     struct nk_rect bounds, nk_flags flags)
16269 {
16270     struct nk_window *win;
16271     struct nk_style *style;
16272     nk_hash name_hash;
16273     int name_len;
16274     int ret = 0;
16275
16276     NK_ASSERT(ctx);
16277     NK_ASSERT(name);
16278     NK_ASSERT(title);
16279     NK_ASSERT(ctx->style.font && ctx->style.font->width && "if this triggers you forgot to add a
      font");
16280     NK_ASSERT(!ctx->current && "if this triggers you missed a `nk_end` call");
16281     if (!ctx || ctx->current || !title || !name)
16282         return 0;
16283
16284     /* find or create window */
16285     style = &ctx->style;
16286     name_len = (int)nk_strlen(name);
16287     name_hash = nk_murmur_hash(name, (int)name_len, NK_WINDOW_TITLE);
16288     win = nk_find_window(ctx, name_hash, name);
16289     if (!win) {
16290         /* create new window */
```

```
16291         nk_size name_length = (nk_size)name_len;
16292         win = (struct nk_window*)nk_create_window(ctx);
16293         NK_ASSERT(win);
16294         if (!win) return 0;
16295
16296         if (flags & NK_WINDOW_BACKGROUND)
16297             nk_insert_window(ctx, win, NK_INSERT_FRONT);
16298         else nk_insert_window(ctx, win, NK_INSERT_BACK);
16299         nk_command_buffer_init(&win->buffer, &ctx->memory, NK_CLIPPING_ON);
16300
16301         win->flags = flags;
16302         win->bounds = bounds;
16303         win->name = name_hash;
16304         name_length = NK_MIN(name_length, NK_WINDOW_MAX_NAME-1);
16305         NK_MEMCPY(win->name_string, name, name_length);
16306         win->name_string[name_length] = 0;
16307         win->popup.win = 0;
16308         if (!ctx->active)
16309             ctx->active = win;
16310     } else {
16311         /* update window */
16312         win->flags &= ~(nk_flags)(NK_WINDOW_PRIVATE-1);
16313         win->flags |= flags;
16314         if (!(win->flags & (NK_WINDOW_MOVABLE | NK_WINDOW_SCALABLE)))
16315             win->bounds = bounds;
16316         /* If this assert triggers you either:
16317          *
16318          * I.) Have more than one window with the same name or
16319          * II.) You forgot to actually draw the window.
16320          *      More specific you did not call `nk_clear` (nk_clear will be
16321          *      automatically called for you if you are using one of the
16322          *      provided demo backends). */
16323         NK_ASSERT(win->seq != ctx->seq);
16324         win->seq = ctx->seq;
16325         if (!ctx->active && !(win->flags & NK_WINDOW_HIDDEN)) {
16326             ctx->active = win;
16327             ctx->end = win;
16328         }
16329     }
16330     if (win->flags & NK_WINDOW_HIDDEN) {
16331         ctx->current = win;
16332         win->layout = 0;
16333         return 0;
16334     } else nk_start(ctx, win);
16335
16336     /* window overlapping */
16337     if (!(win->flags & NK_WINDOW_HIDDEN) && !(win->flags & NK_WINDOW_NO_INPUT))
16338     {
16339         int inpanel, ishovered;
16340         struct nk_window *iter = win;
16341         float h = ctx->style.font->height + 2.0f * style->window.header.padding.y +
16342             (2.0f * style->window.header.label_padding.y);
16343         struct nk_rect win_bounds = (!(win->flags & NK_WINDOW_MINIMIZED))?
16344             win->bounds: nk_rect(win->bounds.x, win->bounds.y, win->bounds.w, h);
16345
16346         /* activate window if hovered and no other window is overlapping this window */
16347         inpanel = nk_input_has_mouse_click_down_in_rect(&ctx->input, NK_BUTTON_LEFT, win_bounds,
    nk_true);
16348         inpanel = inpanel && ctx->input.mouse.buttons[NK_BUTTON_LEFT].clicked;
16349         ishovered = nk_input_is_mouse_hovering_rect(&ctx->input, win_bounds);
16350         if ((win != ctx->active) && ishovered && !ctx->input.mouse.buttons[NK_BUTTON_LEFT].down) {
16351             iter = win->next;
16352             while (iter) {
16353                 struct nk_rect iter_bounds = (!(iter->flags & NK_WINDOW_MINIMIZED))?
16354                     iter->bounds: nk_rect(iter->bounds.x, iter->bounds.y, iter->bounds.w, h);
16355                 if (NK_INTERSECT(win_bounds.x, win_bounds.y, win_bounds.w, win_bounds.h,
16356                     iter_bounds.x, iter_bounds.y, iter_bounds.w, iter_bounds.h) &&
16357                     (!(iter->flags & NK_WINDOW_HIDDEN)))
16358                     break;
16359
16360                 if (iter->popup.win && iter->popup.active && !(iter->flags & NK_WINDOW_HIDDEN) &&
16361                     NK_INTERSECT(win->bounds.x, win_bounds.y, win_bounds.w, win_bounds.h,
16362                     iter->popup.win->bounds.x, iter->popup.win->bounds.y,
16363                     iter->popup.win->bounds.w, iter->popup.win->bounds.h))
16364                     break;
16365                 iter = iter->next;
16366             }
16367         }
16368
16369         /* activate window if clicked */
16370         if (iter && inpanel && (win != ctx->end)) {
16371             iter = win->next;
16372             while (iter) {
16373                 /* try to find a panel with higher priority in the same position */
16374                 struct nk_rect iter_bounds = (!(iter->flags & NK_WINDOW_MINIMIZED))?
16375                 iter->bounds: nk_rect(iter->bounds.x, iter->bounds.y, iter->bounds.w, h);
16376                 if (NK_INBOX(ctx->input.mouse.pos.x, ctx->input.mouse.pos.y,
```

```
16377                   iter_bounds.x, iter_bounds.y, iter_bounds.w, iter_bounds.h) &&
16378                   !(iter->flags & NK_WINDOW_HIDDEN))
16379                   break;
16380               if (iter->popup.win && iter->popup.active && !(iter->flags & NK_WINDOW_HIDDEN) &&
16381                   NK_INTERSECT(win_bounds.x, win_bounds.y, win_bounds.w, win_bounds.h,
16382                   iter->popup.win->bounds.x, iter->popup.win->bounds.y,
16383                   iter->popup.win->bounds.w, iter->popup.win->bounds.h))
16384                   break;
16385               iter = iter->next;
16386           }
16387       }
16388       if (iter && !(win->flags & NK_WINDOW_ROM) && (win->flags & NK_WINDOW_BACKGROUND)) {
16389           win->flags |= (nk_flags)NK_WINDOW_ROM;
16390           iter->flags &= ~(nk_flags)NK_WINDOW_ROM;
16391           ctx->active = iter;
16392           if (!(iter->flags & NK_WINDOW_BACKGROUND)) {
16393               /* current window is active in that position so transfer to top
16394                * at the highest priority in stack */
16395               nk_remove_window(ctx, iter);
16396               nk_insert_window(ctx, iter, NK_INSERT_BACK);
16397           }
16398       } else {
16399           if (!iter && ctx->end != win) {
16400               if (!(win->flags & NK_WINDOW_BACKGROUND)) {
16401                   /* current window is active in that position so transfer to top
16402                    * at the highest priority in stack */
16403                   nk_remove_window(ctx, win);
16404                   nk_insert_window(ctx, win, NK_INSERT_BACK);
16405               }
16406               win->flags &= ~(nk_flags)NK_WINDOW_ROM;
16407               ctx->active = win;
16408           }
16409           if (ctx->end != win && !(win->flags & NK_WINDOW_BACKGROUND))
16410               win->flags |= NK_WINDOW_ROM;
16411       }
16412   }
16413   win->layout = (struct nk_panel*)nk_create_panel(ctx);
16414   ctx->current = win;
16415   ret = nk_panel_begin(ctx, title, NK_PANEL_WINDOW);
16416   win->layout->offset_x = &win->scrollbar.x;
16417   win->layout->offset_y = &win->scrollbar.y;
16418   return ret;
16419 }
16420 NK_API void
16421 nk_end(struct nk_context *ctx)
16422 {
16423     struct nk_panel *layout;
16424     NK_ASSERT(ctx);
16425     NK_ASSERT(ctx->current && "if this triggers you forgot to call `nk_begin`");
16426     if (!ctx || !ctx->current)
16427         return;
16428
16429     layout = ctx->current->layout;
16430     if (!layout || (layout->type == NK_PANEL_WINDOW && (ctx->current->flags & NK_WINDOW_HIDDEN))) {
16431         ctx->current = 0;
16432         return;
16433     }
16434     nk_panel_end(ctx);
16435     nk_free_panel(ctx, ctx->current->layout);
16436     ctx->current = 0;
16437 }
16438 NK_API struct nk_rect
16439 nk_window_get_bounds(const struct nk_context *ctx)
16440 {
16441     NK_ASSERT(ctx);
16442     NK_ASSERT(ctx->current);
16443     if (!ctx || !ctx->current) return nk_rect(0,0,0,0);
16444     return ctx->current->bounds;
16445 }
16446 NK_API struct nk_vec2
16447 nk_window_get_position(const struct nk_context *ctx)
16448 {
16449     NK_ASSERT(ctx);
16450     NK_ASSERT(ctx->current);
16451     if (!ctx || !ctx->current) return nk_vec2(0,0);
16452     return nk_vec2(ctx->current->bounds.x, ctx->current->bounds.y);
16453 }
16454 NK_API struct nk_vec2
16455 nk_window_get_size(const struct nk_context *ctx)
16456 {
16457     NK_ASSERT(ctx);
16458     NK_ASSERT(ctx->current);
16459     if (!ctx || !ctx->current) return nk_vec2(0,0);
16460     return nk_vec2(ctx->current->bounds.w, ctx->current->bounds.h);
16461 }
16462 NK_API float
16463 nk_window_get_width(const struct nk_context *ctx)
```

```
16464 {
16465     NK_ASSERT(ctx);
16466     NK_ASSERT(ctx->current);
16467     if (!ctx || !ctx->current) return 0;
16468     return ctx->current->bounds.w;
16469 }
16470 NK_API float
16471 nk_window_get_height(const struct nk_context *ctx)
16472 {
16473     NK_ASSERT(ctx);
16474     NK_ASSERT(ctx->current);
16475     if (!ctx || !ctx->current) return 0;
16476     return ctx->current->bounds.h;
16477 }
16478 NK_API struct nk_rect
16479 nk_window_get_content_region(struct nk_context *ctx)
16480 {
16481     NK_ASSERT(ctx);
16482     NK_ASSERT(ctx->current);
16483     if (!ctx || !ctx->current) return nk_rect(0,0,0,0);
16484     return ctx->current->layout->clip;
16485 }
16486 NK_API struct nk_vec2
16487 nk_window_get_content_region_min(struct nk_context *ctx)
16488 {
16489     NK_ASSERT(ctx);
16490     NK_ASSERT(ctx->current);
16491     NK_ASSERT(ctx->current->layout);
16492     if (!ctx || !ctx->current) return nk_vec2(0,0);
16493     return nk_vec2(ctx->current->layout->clip.x, ctx->current->layout->clip.y);
16494 }
16495 NK_API struct nk_vec2
16496 nk_window_get_content_region_max(struct nk_context *ctx)
16497 {
16498     NK_ASSERT(ctx);
16499     NK_ASSERT(ctx->current);
16500     NK_ASSERT(ctx->current->layout);
16501     if (!ctx || !ctx->current) return nk_vec2(0,0);
16502     return nk_vec2(ctx->current->layout->clip.x + ctx->current->layout->clip.w,
16503         ctx->current->layout->clip.y + ctx->current->layout->clip.h);
16504 }
16505 NK_API struct nk_vec2
16506 nk_window_get_content_region_size(struct nk_context *ctx)
16507 {
16508     NK_ASSERT(ctx);
16509     NK_ASSERT(ctx->current);
16510     NK_ASSERT(ctx->current->layout);
16511     if (!ctx || !ctx->current) return nk_vec2(0,0);
16512     return nk_vec2(ctx->current->layout->clip.w, ctx->current->layout->clip.h);
16513 }
16514 NK_API struct nk_command_buffer*
16515 nk_window_get_canvas(struct nk_context *ctx)
16516 {
16517     NK_ASSERT(ctx);
16518     NK_ASSERT(ctx->current);
16519     NK_ASSERT(ctx->current->layout);
16520     if (!ctx || !ctx->current) return 0;
16521     return &ctx->current->buffer;
16522 }
16523 NK_API struct nk_panel*
16524 nk_window_get_panel(struct nk_context *ctx)
16525 {
16526     NK_ASSERT(ctx);
16527     NK_ASSERT(ctx->current);
16528     if (!ctx || !ctx->current) return 0;
16529     return ctx->current->layout;
16530 }
16531 NK_API void
16532 nk_window_get_scroll(struct nk_context *ctx, nk_uint *offset_x, nk_uint *offset_y)
16533 {
16534     struct nk_window *win;
16535     NK_ASSERT(ctx);
16536     NK_ASSERT(ctx->current);
16537     if (!ctx || !ctx->current)
16538         return ;
16539     win = ctx->current;
16540     if (offset_x)
16541       *offset_x = win->scrollbar.x;
16542     if (offset_y)
16543       *offset_y = win->scrollbar.y;
16544 }
16545 NK_API int
16546 nk_window_has_focus(const struct nk_context *ctx)
16547 {
16548     NK_ASSERT(ctx);
16549     NK_ASSERT(ctx->current);
16550     NK_ASSERT(ctx->current->layout);
```

```
16551     if (!ctx || !ctx->current) return 0;
16552     return ctx->current == ctx->active;
16553 }
16554 NK_API int
16555 nk_window_is_hovered(struct nk_context *ctx)
16556 {
16557     NK_ASSERT(ctx);
16558     NK_ASSERT(ctx->current);
16559     if (!ctx || !ctx->current) return 0;
16560     if(ctx->current->flags & NK_WINDOW_HIDDEN)
16561         return 0;
16562     return nk_input_is_mouse_hovering_rect(&ctx->input, ctx->current->bounds);
16563 }
16564 NK_API int
16565 nk_window_is_any_hovered(struct nk_context *ctx)
16566 {
16567     struct nk_window *iter;
16568     NK_ASSERT(ctx);
16569     if (!ctx) return 0;
16570     iter = ctx->begin;
16571     while (iter) {
16572         /* check if window is being hovered */
16573         if(!(iter->flags & NK_WINDOW_HIDDEN)) {
16574             /* check if window popup is being hovered */
16575             if (iter->popup.active && iter->popup.win && nk_input_is_mouse_hovering_rect(&ctx->input,
     iter->popup.win->bounds))
16576                 return 1;
16577
16578             if (iter->flags & NK_WINDOW_MINIMIZED) {
16579                 struct nk_rect header = iter->bounds;
16580                 header.h = ctx->style.font->height + 2 * ctx->style.window.header.padding.y;
16581                 if (nk_input_is_mouse_hovering_rect(&ctx->input, header))
16582                     return 1;
16583             } else if (nk_input_is_mouse_hovering_rect(&ctx->input, iter->bounds)) {
16584                 return 1;
16585             }
16586         }
16587         iter = iter->next;
16588     }
16589     return 0;
16590 }
16591 NK_API int
16592 nk_item_is_any_active(struct nk_context *ctx)
16593 {
16594     int any_hovered = nk_window_is_any_hovered(ctx);
16595     int any_active = (ctx->last_widget_state & NK_WIDGET_STATE_MODIFIED);
16596     return any_hovered || any_active;
16597 }
16598 NK_API int
16599 nk_window_is_collapsed(struct nk_context *ctx, const char *name)
16600 {
16601     int title_len;
16602     nk_hash title_hash;
16603     struct nk_window *win;
16604     NK_ASSERT(ctx);
16605     if (!ctx) return 0;
16606
16607     title_len = (int)nk_strlen(name);
16608     title_hash = nk_murmur_hash(name, (int)title_len, NK_WINDOW_TITLE);
16609     win = nk_find_window(ctx, title_hash, name);
16610     if (!win) return 0;
16611     return win->flags & NK_WINDOW_MINIMIZED;
16612 }
16613 NK_API int
16614 nk_window_is_closed(struct nk_context *ctx, const char *name)
16615 {
16616     int title_len;
16617     nk_hash title_hash;
16618     struct nk_window *win;
16619     NK_ASSERT(ctx);
16620     if (!ctx) return 1;
16621
16622     title_len = (int)nk_strlen(name);
16623     title_hash = nk_murmur_hash(name, (int)title_len, NK_WINDOW_TITLE);
16624     win = nk_find_window(ctx, title_hash, name);
16625     if (!win) return 1;
16626     return (win->flags & NK_WINDOW_CLOSED);
16627 }
16628 NK_API int
16629 nk_window_is_hidden(struct nk_context *ctx, const char *name)
16630 {
16631     int title_len;
16632     nk_hash title_hash;
16633     struct nk_window *win;
16634     NK_ASSERT(ctx);
16635     if (!ctx) return 1;
16636
```

```
16637     title_len = (int)nk_strlen(name);
16638     title_hash = nk_murmur_hash(name, (int)title_len, NK_WINDOW_TITLE);
16639     win = nk_find_window(ctx, title_hash, name);
16640     if (!win) return 1;
16641     return (win->flags & NK_WINDOW_HIDDEN);
16642 }
16643 NK_API int
16644 nk_window_is_active(struct nk_context *ctx, const char *name)
16645 {
16646     int title_len;
16647     nk_hash title_hash;
16648     struct nk_window *win;
16649     NK_ASSERT(ctx);
16650     if (!ctx) return 0;
16651
16652     title_len = (int)nk_strlen(name);
16653     title_hash = nk_murmur_hash(name, (int)title_len, NK_WINDOW_TITLE);
16654     win = nk_find_window(ctx, title_hash, name);
16655     if (!win) return 0;
16656     return win == ctx->active;
16657 }
16658 NK_API struct nk_window*
16659 nk_window_find(struct nk_context *ctx, const char *name)
16660 {
16661     int title_len;
16662     nk_hash title_hash;
16663     title_len = (int)nk_strlen(name);
16664     title_hash = nk_murmur_hash(name, (int)title_len, NK_WINDOW_TITLE);
16665     return nk_find_window(ctx, title_hash, name);
16666 }
16667 NK_API void
16668 nk_window_close(struct nk_context *ctx, const char *name)
16669 {
16670     struct nk_window *win;
16671     NK_ASSERT(ctx);
16672     if (!ctx) return;
16673     win = nk_window_find(ctx, name);
16674     if (!win) return;
16675     NK_ASSERT(ctx->current != win && "You cannot close a currently active window");
16676     if (ctx->current == win) return;
16677     win->flags |= NK_WINDOW_HIDDEN;
16678     win->flags |= NK_WINDOW_CLOSED;
16679 }
16680 NK_API void
16681 nk_window_set_bounds(struct nk_context *ctx,
16682     const char *name, struct nk_rect bounds)
16683 {
16684     struct nk_window *win;
16685     NK_ASSERT(ctx);
16686     if (!ctx) return;
16687     win = nk_window_find(ctx, name);
16688     if (!win) return;
16689     NK_ASSERT(ctx->current != win && "You cannot update a currently in procecss window");
16690     win->bounds = bounds;
16691 }
16692 NK_API void
16693 nk_window_set_position(struct nk_context *ctx,
16694     const char *name, struct nk_vec2 pos)
16695 {
16696     struct nk_window *win = nk_window_find(ctx, name);
16697     if (!win) return;
16698     win->bounds.x = pos.x;
16699     win->bounds.y = pos.y;
16700 }
16701 NK_API void
16702 nk_window_set_size(struct nk_context *ctx,
16703     const char *name, struct nk_vec2 size)
16704 {
16705     struct nk_window *win = nk_window_find(ctx, name);
16706     if (!win) return;
16707     win->bounds.w = size.x;
16708     win->bounds.h = size.y;
16709 }
16710 NK_API void
16711 nk_window_set_scroll(struct nk_context *ctx, nk_uint offset_x, nk_uint offset_y)
16712 {
16713     struct nk_window *win;
16714     NK_ASSERT(ctx);
16715     NK_ASSERT(ctx->current);
16716     if (!ctx || !ctx->current)
16717         return;
16718     win = ctx->current;
16719     win->scrollbar.x = offset_x;
16720     win->scrollbar.y = offset_y;
16721 }
16722 NK_API void
16723 nk_window_collapse(struct nk_context *ctx, const char *name,
```

```
16724                       enum nk_collapse_states c)
16725 {
16726     int title_len;
16727     nk_hash title_hash;
16728     struct nk_window *win;
16729     NK_ASSERT(ctx);
16730     if (!ctx) return;
16731
16732     title_len = (int)nk_strlen(name);
16733     title_hash = nk_murmur_hash(name, (int)title_len, NK_WINDOW_TITLE);
16734     win = nk_find_window(ctx, title_hash, name);
16735     if (!win) return;
16736     if (c == NK_MINIMIZED)
16737         win->flags |= NK_WINDOW_MINIMIZED;
16738     else win->flags &= ~(nk_flags)NK_WINDOW_MINIMIZED;
16739 }
16740 NK_API void
16741 nk_window_collapse_if(struct nk_context *ctx, const char *name,
16742     enum nk_collapse_states c, int cond)
16743 {
16744     NK_ASSERT(ctx);
16745     if (!ctx || !cond) return;
16746     nk_window_collapse(ctx, name, c);
16747 }
16748 NK_API void
16749 nk_window_show(struct nk_context *ctx, const char *name, enum nk_show_states s)
16750 {
16751     int title_len;
16752     nk_hash title_hash;
16753     struct nk_window *win;
16754     NK_ASSERT(ctx);
16755     if (!ctx) return;
16756
16757     title_len = (int)nk_strlen(name);
16758     title_hash = nk_murmur_hash(name, (int)title_len, NK_WINDOW_TITLE);
16759     win = nk_find_window(ctx, title_hash, name);
16760     if (!win) return;
16761     if (s == NK_HIDDEN) {
16762         win->flags |= NK_WINDOW_HIDDEN;
16763     } else win->flags &= ~(nk_flags)NK_WINDOW_HIDDEN;
16764 }
16765 NK_API void
16766 nk_window_show_if(struct nk_context *ctx, const char *name,
16767     enum nk_show_states s, int cond)
16768 {
16769     NK_ASSERT(ctx);
16770     if (!ctx || !cond) return;
16771     nk_window_show(ctx, name, s);
16772 }
16773
16774 NK_API void
16775 nk_window_set_focus(struct nk_context *ctx, const char *name)
16776 {
16777     int title_len;
16778     nk_hash title_hash;
16779     struct nk_window *win;
16780     NK_ASSERT(ctx);
16781     if (!ctx) return;
16782
16783     title_len = (int)nk_strlen(name);
16784     title_hash = nk_murmur_hash(name, (int)title_len, NK_WINDOW_TITLE);
16785     win = nk_find_window(ctx, title_hash, name);
16786     if (win && ctx->end != win) {
16787         nk_remove_window(ctx, win);
16788         nk_insert_window(ctx, win, NK_INSERT_BACK);
16789     }
16790     ctx->active = win;
16791 }
16792
16793
16794
16795
16796 /* ==============================================================
16797  *
16798  *                              POPUP
16799  *
16800  * ===============================================================*/
16801 NK_API int
16802 nk_popup_begin(struct nk_context *ctx, enum nk_popup_type type,
16803     const char *title, nk_flags flags, struct nk_rect rect)
16804 {
16805     struct nk_window *popup;
16806     struct nk_window *win;
16807     struct nk_panel *panel;
16808
16809     int title_len;
16810     nk_hash title_hash;
```

```
16811     nk_size allocated;
16812
16813     NK_ASSERT(ctx);
16814     NK_ASSERT(title);
16815     NK_ASSERT(ctx->current);
16816     NK_ASSERT(ctx->current->layout);
16817     if (!ctx || !ctx->current || !ctx->current->layout)
16818         return 0;
16819
16820     win = ctx->current;
16821     panel = win->layout;
16822     NK_ASSERT(!(panel->type & NK_PANEL_SET_POPUP) && "popups are not allowed to have popups");
16823     (void)panel;
16824     title_len = (int)nk_strlen(title);
16825     title_hash = nk_murmur_hash(title, (int)title_len, NK_PANEL_POPUP);
16826
16827     popup = win->popup.win;
16828     if (!popup) {
16829         popup = (struct nk_window*)nk_create_window(ctx);
16830         popup->parent = win;
16831         win->popup.win = popup;
16832         win->popup.active = 0;
16833         win->popup.type = NK_PANEL_POPUP;
16834     }
16835
16836     /* make sure we have correct popup */
16837     if (win->popup.name != title_hash) {
16838         if (!win->popup.active) {
16839             nk_zero(popup, sizeof(*popup));
16840             win->popup.name = title_hash;
16841             win->popup.active = 1;
16842             win->popup.type = NK_PANEL_POPUP;
16843         } else return 0;
16844     }
16845
16846     /* popup position is local to window */
16847     ctx->current = popup;
16848     rect.x += win->layout->clip.x;
16849     rect.y += win->layout->clip.y;
16850
16851     /* setup popup data */
16852     popup->parent = win;
16853     popup->bounds = rect;
16854     popup->seq = ctx->seq;
16855     popup->layout = (struct nk_panel*)nk_create_panel(ctx);
16856     popup->flags = flags;
16857     popup->flags |= NK_WINDOW_BORDER;
16858     if (type == NK_POPUP_DYNAMIC)
16859         popup->flags |= NK_WINDOW_DYNAMIC;
16860
16861     popup->buffer = win->buffer;
16862     nk_start_popup(ctx, win);
16863     allocated = ctx->memory.allocated;
16864     nk_push_scissor(&popup->buffer, nk_null_rect);
16865
16866     if (nk_panel_begin(ctx, title, NK_PANEL_POPUP)) {
16867         /* popup is running therefore invalidate parent panels */
16868         struct nk_panel *root;
16869         root = win->layout;
16870         while (root) {
16871             root->flags |= NK_WINDOW_ROM;
16872             root->flags &= ~(nk_flags)NK_WINDOW_REMOVE_ROM;
16873             root = root->parent;
16874         }
16875         win->popup.active = 1;
16876         popup->layout->offset_x = &popup->scrollbar.x;
16877         popup->layout->offset_y = &popup->scrollbar.y;
16878         popup->layout->parent = win->layout;
16879         return 1;
16880     } else {
16881         /* popup was closed/is invalid so cleanup */
16882         struct nk_panel *root;
16883         root = win->layout;
16884         while (root) {
16885             root->flags |= NK_WINDOW_REMOVE_ROM;
16886             root = root->parent;
16887         }
16888         win->popup.buf.active = 0;
16889         win->popup.active = 0;
16890         ctx->memory.allocated = allocated;
16891         ctx->current = win;
16892         nk_free_panel(ctx, popup->layout);
16893         popup->layout = 0;
16894         return 0;
16895     }
16896 }
16897 NK_LIB int
```

```
16898 nk_nonblock_begin(struct nk_context *ctx,
16899     nk_flags flags, struct nk_rect body, struct nk_rect header,
16900     enum nk_panel_type panel_type)
16901 {
16902     struct nk_window *popup;
16903     struct nk_window *win;
16904     struct nk_panel *panel;
16905     int is_active = nk_true;
16906
16907     NK_ASSERT(ctx);
16908     NK_ASSERT(ctx->current);
16909     NK_ASSERT(ctx->current->layout);
16910     if (!ctx || !ctx->current || !ctx->current->layout)
16911         return 0;
16912
16913     /* popups cannot have popups */
16914     win = ctx->current;
16915     panel = win->layout;
16916     NK_ASSERT(!(panel->type & NK_PANEL_SET_POPUP));
16917     (void)panel;
16918     popup = win->popup.win;
16919     if (!popup) {
16920         /* create window for nonblocking popup */
16921         popup = (struct nk_window*)nk_create_window(ctx);
16922         popup->parent = win;
16923         win->popup.win = popup;
16924         win->popup.type = panel_type;
16925         nk_command_buffer_init(&popup->buffer, &ctx->memory, NK_CLIPPING_ON);
16926     } else {
16927         /* close the popup if user pressed outside or in the header */
16928         int pressed, in_body, in_header;
16929 #ifdef NK_BUTTON_TRIGGER_ON_RELEASE
16930         pressed = nk_input_is_mouse_released(&ctx->input, NK_BUTTON_LEFT);
16931 #else
16932         pressed = nk_input_is_mouse_pressed(&ctx->input, NK_BUTTON_LEFT);
16933 #endif
16934         in_body = nk_input_is_mouse_hovering_rect(&ctx->input, body);
16935         in_header = nk_input_is_mouse_hovering_rect(&ctx->input, header);
16936         if (pressed && (!in_body || in_header))
16937             is_active = nk_false;
16938     }
16939     win->popup.header = header;
16940
16941     if (!is_active) {
16942         /* remove read only mode from all parent panels */
16943         struct nk_panel *root = win->layout;
16944         while (root) {
16945             root->flags |= NK_WINDOW_REMOVE_ROM;
16946             root = root->parent;
16947         }
16948         return is_active;
16949     }
16950     popup->bounds = body;
16951     popup->parent = win;
16952     popup->layout = (struct nk_panel*)nk_create_panel(ctx);
16953     popup->flags = flags;
16954     popup->flags |= NK_WINDOW_BORDER;
16955     popup->flags |= NK_WINDOW_DYNAMIC;
16956     popup->seq = ctx->seq;
16957     win->popup.active = 1;
16958     NK_ASSERT(popup->layout);
16959
16960     nk_start_popup(ctx, win);
16961     popup->buffer = win->buffer;
16962     nk_push_scissor(&popup->buffer, nk_null_rect);
16963     ctx->current = popup;
16964
16965     nk_panel_begin(ctx, 0, panel_type);
16966     win->buffer = popup->buffer;
16967     popup->layout->parent = win->layout;
16968     popup->layout->offset_x = &popup->scrollbar.x;
16969     popup->layout->offset_y = &popup->scrollbar.y;
16970
16971     /* set read only mode to all parent panels */
16972     {struct nk_panel *root;
16973     root = win->layout;
16974     while (root) {
16975         root->flags |= NK_WINDOW_ROM;
16976         root = root->parent;
16977     }}
16978     return is_active;
16979 }
16980 NK_API void
16981 nk_popup_close(struct nk_context *ctx)
16982 {
16983     struct nk_window *popup;
16984     NK_ASSERT(ctx);
```

```
16985     if (!ctx || !ctx->current) return;
16986
16987     popup = ctx->current;
16988     NK_ASSERT(popup->parent);
16989     NK_ASSERT(popup->layout->type & NK_PANEL_SET_POPUP);
16990     popup->flags |= NK_WINDOW_HIDDEN;
16991 }
16992 NK_API void
16993 nk_popup_end(struct nk_context *ctx)
16994 {
16995     struct nk_window *win;
16996     struct nk_window *popup;
16997
16998     NK_ASSERT(ctx);
16999     NK_ASSERT(ctx->current);
17000     NK_ASSERT(ctx->current->layout);
17001     if (!ctx || !ctx->current || !ctx->current->layout)
17002         return;
17003
17004     popup = ctx->current;
17005     if (!popup->parent) return;
17006     win = popup->parent;
17007     if (popup->flags & NK_WINDOW_HIDDEN) {
17008         struct nk_panel *root;
17009         root = win->layout;
17010         while (root) {
17011             root->flags |= NK_WINDOW_REMOVE_ROM;
17012             root = root->parent;
17013         }
17014         win->popup.active = 0;
17015     }
17016     nk_push_scissor(&popup->buffer, nk_null_rect);
17017     nk_end(ctx);
17018
17019     win->buffer = popup->buffer;
17020     nk_finish_popup(ctx, win);
17021     ctx->current = win;
17022     nk_push_scissor(&win->buffer, win->layout->clip);
17023 }
17024 NK_API void
17025 nk_popup_get_scroll(struct nk_context *ctx, nk_uint *offset_x, nk_uint *offset_y)
17026 {
17027     struct nk_window *popup;
17028
17029     NK_ASSERT(ctx);
17030     NK_ASSERT(ctx->current);
17031     NK_ASSERT(ctx->current->layout);
17032     if (!ctx || !ctx->current || !ctx->current->layout)
17033         return;
17034
17035     popup = ctx->current;
17036     if (offset_x)
17037       *offset_x = popup->scrollbar.x;
17038     if (offset_y)
17039       *offset_y = popup->scrollbar.y;
17040 }
17041 NK_API void
17042 nk_popup_set_scroll(struct nk_context *ctx, nk_uint offset_x, nk_uint offset_y)
17043 {
17044     struct nk_window *popup;
17045
17046     NK_ASSERT(ctx);
17047     NK_ASSERT(ctx->current);
17048     NK_ASSERT(ctx->current->layout);
17049     if (!ctx || !ctx->current || !ctx->current->layout)
17050         return;
17051
17052     popup = ctx->current;
17053     popup->scrollbar.x = offset_x;
17054     popup->scrollbar.y = offset_y;
17055 }
17056
17057
17058
17059
17060 /* ===============================================================
17061  *
17062  *                          CONTEXTUAL
17063  *
17064  * ===============================================================*/
17065 NK_API int
17066 nk_contextual_begin(struct nk_context *ctx, nk_flags flags, struct nk_vec2 size,
17067     struct nk_rect trigger_bounds)
17068 {
17069     struct nk_window *win;
17070     struct nk_window *popup;
17071     struct nk_rect body;
```

```
17072
17073     NK_STORAGE const struct nk_rect null_rect = {-1,-1,0,0};
17074     int is_clicked = 0;
17075     int is_open = 0;
17076     int ret = 0;
17077
17078     NK_ASSERT(ctx);
17079     NK_ASSERT(ctx->current);
17080     NK_ASSERT(ctx->current->layout);
17081     if (!ctx || !ctx->current || !ctx->current->layout)
17082         return 0;
17083
17084     win = ctx->current;
17085     ++win->popup.con_count;
17086     if (ctx->current != ctx->active)
17087         return 0;
17088
17089     /* check if currently active contextual is active */
17090     popup = win->popup.win;
17091     is_open = (popup && win->popup.type == NK_PANEL_CONTEXTUAL);
17092     is_clicked = nk_input_mouse_clicked(&ctx->input, NK_BUTTON_RIGHT, trigger_bounds);
17093     if (win->popup.active_con && win->popup.con_count != win->popup.active_con)
17094         return 0;
17095     if (!is_open && win->popup.active_con)
17096         win->popup.active_con = 0;
17097     if ((!is_open && !is_clicked))
17098         return 0;
17099
17100     /* calculate contextual position on click */
17101     win->popup.active_con = win->popup.con_count;
17102     if (is_clicked) {
17103         body.x = ctx->input.mouse.pos.x;
17104         body.y = ctx->input.mouse.pos.y;
17105     } else {
17106         body.x = popup->bounds.x;
17107         body.y = popup->bounds.y;
17108     }
17109     body.w = size.x;
17110     body.h = size.y;
17111
17112     /* start nonblocking contextual popup */
17113     ret = nk_nonblock_begin(ctx, flags|NK_WINDOW_NO_SCROLLBAR, body,
17114             null_rect, NK_PANEL_CONTEXTUAL);
17115     if (ret) win->popup.type = NK_PANEL_CONTEXTUAL;
17116     else {
17117         win->popup.active_con = 0;
17118         win->popup.type = NK_PANEL_NONE;
17119         if (win->popup.win)
17120             win->popup.win->flags = 0;
17121     }
17122     return ret;
17123 }
17124 NK_API int
17125 nk_contextual_item_text(struct nk_context *ctx, const char *text, int len,
17126     nk_flags alignment)
17127 {
17128     struct nk_window *win;
17129     const struct nk_input *in;
17130     const struct nk_style *style;
17131
17132     struct nk_rect bounds;
17133     enum nk_widget_layout_states state;
17134
17135     NK_ASSERT(ctx);
17136     NK_ASSERT(ctx->current);
17137     NK_ASSERT(ctx->current->layout);
17138     if (!ctx || !ctx->current || !ctx->current->layout)
17139         return 0;
17140
17141     win = ctx->current;
17142     style = &ctx->style;
17143     state = nk_widget_fitting(&bounds, ctx, style->contextual_button.padding);
17144     if (!state) return nk_false;
17145
17146     in = (state == NK_WIDGET_ROM || win->layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
17147     if (nk_do_button_text(&ctx->last_widget_state, &win->buffer, bounds,
17148         text, len, alignment, NK_BUTTON_DEFAULT, &style->contextual_button, in, style->font)) {
17149         nk_contextual_close(ctx);
17150         return nk_true;
17151     }
17152     return nk_false;
17153 }
17154 NK_API int
17155 nk_contextual_item_label(struct nk_context *ctx, const char *label, nk_flags align)
17156 {
17157     return nk_contextual_item_text(ctx, label, nk_strlen(label), align);
17158 }
```

```
17159 NK_API int
17160 nk_contextual_item_image_text(struct nk_context *ctx, struct nk_image img,
17161     const char *text, int len, nk_flags align)
17162 {
17163     struct nk_window *win;
17164     const struct nk_input *in;
17165     const struct nk_style *style;
17166
17167     struct nk_rect bounds;
17168     enum nk_widget_layout_states state;
17169
17170     NK_ASSERT(ctx);
17171     NK_ASSERT(ctx->current);
17172     NK_ASSERT(ctx->current->layout);
17173     if (!ctx || !ctx->current || !ctx->current->layout)
17174         return 0;
17175
17176     win = ctx->current;
17177     style = &ctx->style;
17178     state = nk_widget_fitting(&bounds, ctx, style->contextual_button.padding);
17179     if (!state) return nk_false;
17180
17181     in = (state == NK_WIDGET_ROM || win->layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
17182     if (nk_do_button_text_image(&ctx->last_widget_state, &win->buffer, bounds,
17183         img, text, len, align, NK_BUTTON_DEFAULT, &style->contextual_button, style->font, in)){
17184         nk_contextual_close(ctx);
17185         return nk_true;
17186     }
17187     return nk_false;
17188 }
17189 NK_API int
17190 nk_contextual_item_image_label(struct nk_context *ctx, struct nk_image img,
17191     const char *label, nk_flags align)
17192 {
17193     return nk_contextual_item_image_text(ctx, img, label, nk_strlen(label), align);
17194 }
17195 NK_API int
17196 nk_contextual_item_symbol_text(struct nk_context *ctx, enum nk_symbol_type symbol,
17197     const char *text, int len, nk_flags align)
17198 {
17199     struct nk_window *win;
17200     const struct nk_input *in;
17201     const struct nk_style *style;
17202
17203     struct nk_rect bounds;
17204     enum nk_widget_layout_states state;
17205
17206     NK_ASSERT(ctx);
17207     NK_ASSERT(ctx->current);
17208     NK_ASSERT(ctx->current->layout);
17209     if (!ctx || !ctx->current || !ctx->current->layout)
17210         return 0;
17211
17212     win = ctx->current;
17213     style = &ctx->style;
17214     state = nk_widget_fitting(&bounds, ctx, style->contextual_button.padding);
17215     if (!state) return nk_false;
17216
17217     in = (state == NK_WIDGET_ROM || win->layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
17218     if (nk_do_button_text_symbol(&ctx->last_widget_state, &win->buffer, bounds,
17219         symbol, text, len, align, NK_BUTTON_DEFAULT, &style->contextual_button, style->font, in)) {
17220         nk_contextual_close(ctx);
17221         return nk_true;
17222     }
17223     return nk_false;
17224 }
17225 NK_API int
17226 nk_contextual_item_symbol_label(struct nk_context *ctx, enum nk_symbol_type symbol,
17227     const char *text, nk_flags align)
17228 {
17229     return nk_contextual_item_symbol_text(ctx, symbol, text, nk_strlen(text), align);
17230 }
17231 NK_API void
17232 nk_contextual_close(struct nk_context *ctx)
17233 {
17234     NK_ASSERT(ctx);
17235     NK_ASSERT(ctx->current);
17236     NK_ASSERT(ctx->current->layout);
17237     if (!ctx || !ctx->current || !ctx->current->layout) return;
17238     nk_popup_close(ctx);
17239 }
17240 NK_API void
17241 nk_contextual_end(struct nk_context *ctx)
17242 {
17243     struct nk_window *popup;
17244     struct nk_panel *panel;
17245     NK_ASSERT(ctx);
```

```
17246     NK_ASSERT(ctx->current);
17247     if (!ctx || !ctx->current) return;
17248
17249     popup = ctx->current;
17250     panel = popup->layout;
17251     NK_ASSERT(popup->parent);
17252     NK_ASSERT(panel->type & NK_PANEL_SET_POPUP);
17253     if (panel->flags & NK_WINDOW_DYNAMIC) {
17254         /* Close behavior
17255         This is a bit of a hack solution since we do not know before we end our popup
17256         how big it will be. We therefore do not directly know when a
17257         click outside the non-blocking popup must close it at that direct frame.
17258         Instead it will be closed in the next frame.*/
17259         struct nk_rect body = {0,0,0,0};
17260         if (panel->at_y < (panel->bounds.y + panel->bounds.h)) {
17261             struct nk_vec2 padding = nk_panel_get_padding(&ctx->style, panel->type);
17262             body = panel->bounds;
17263             body.y = (panel->at_y + panel->footer_height + panel->border + padding.y +
      panel->row.height);
17264             body.h = (panel->bounds.y + panel->bounds.h) - body.y;
17265         }
17266         {int pressed = nk_input_is_mouse_pressed(&ctx->input, NK_BUTTON_LEFT);
17267         int in_body = nk_input_is_mouse_hovering_rect(&ctx->input, body);
17268         if (pressed && in_body)
17269             popup->flags |= NK_WINDOW_HIDDEN;
17270         }
17271     }
17272     if (popup->flags & NK_WINDOW_HIDDEN)
17273         popup->seq = 0;
17274     nk_popup_end(ctx);
17275     return;
17276 }
17277
17278
17279
17280
17281
17282 /* ===============================================================
17283  *
17284  *                              MENU
17285  *
17286  * ===============================================================*/
17287 NK_API void
17288 nk_menubar_begin(struct nk_context *ctx)
17289 {
17290     struct nk_panel *layout;
17291     NK_ASSERT(ctx);
17292     NK_ASSERT(ctx->current);
17293     NK_ASSERT(ctx->current->layout);
17294     if (!ctx || !ctx->current || !ctx->current->layout)
17295         return;
17296
17297     layout = ctx->current->layout;
17298     NK_ASSERT(layout->at_y == layout->bounds.y);
17299     /* if this assert triggers you allocated space between nk_begin and nk_menubar_begin.
17300     If you want a menubar the first nuklear function after `nk_begin` has to be a
17301     `nk_menubar_begin` call. Inside the menubar you then have to allocate space for
17302     widgets (also supports multiple rows).
17303     Example:
17304         if (nk_begin(...)) {
17305             nk_menubar_begin(...);
17306                 nk_layout_xxxx(...);
17307                 nk_button(...);
17308                 nk_layout_xxxx(...);
17309                 nk_button(...);
17310             nk_menubar_end(...);
17311         }
17312         nk_end(...);
17313     */
17314     if (layout->flags & NK_WINDOW_HIDDEN || layout->flags & NK_WINDOW_MINIMIZED)
17315         return;
17316
17317     layout->menu.x = layout->at_x;
17318     layout->menu.y = layout->at_y + layout->row.height;
17319     layout->menu.w = layout->bounds.w;
17320     layout->menu.offset.x = *layout->offset_x;
17321     layout->menu.offset.y = *layout->offset_y;
17322     *layout->offset_y = 0;
17323 }
17324 NK_API void
17325 nk_menubar_end(struct nk_context *ctx)
17326 {
17327     struct nk_window *win;
17328     struct nk_panel *layout;
17329     struct nk_command_buffer *out;
17330
17331     NK_ASSERT(ctx);
```

```
17332     NK_ASSERT(ctx->current);
17333     NK_ASSERT(ctx->current->layout);
17334     if (!ctx || !ctx->current || !ctx->current->layout)
17335         return;
17336
17337     win = ctx->current;
17338     out = &win->buffer;
17339     layout = win->layout;
17340     if (layout->flags & NK_WINDOW_HIDDEN || layout->flags & NK_WINDOW_MINIMIZED)
17341         return;
17342
17343     layout->menu.h = layout->at_y - layout->menu.y;
17344     layout->bounds.y += layout->menu.h + ctx->style.window.spacing.y + layout->row.height;
17345     layout->bounds.h -= layout->menu.h + ctx->style.window.spacing.y + layout->row.height;
17346
17347     *layout->offset_x = layout->menu.offset.x;
17348     *layout->offset_y = layout->menu.offset.y;
17349     layout->at_y = layout->bounds.y - layout->row.height;
17350
17351     layout->clip.y = layout->bounds.y;
17352     layout->clip.h = layout->bounds.h;
17353     nk_push_scissor(out, layout->clip);
17354 }
17355 NK_INTERN int
17356 nk_menu_begin(struct nk_context *ctx, struct nk_window *win,
17357     const char *id, int is_clicked, struct nk_rect header, struct nk_vec2 size)
17358 {
17359     int is_open = 0;
17360     int is_active = 0;
17361     struct nk_rect body;
17362     struct nk_window *popup;
17363     nk_hash hash = nk_murmur_hash(id, (int)nk_strlen(id), NK_PANEL_MENU);
17364
17365     NK_ASSERT(ctx);
17366     NK_ASSERT(ctx->current);
17367     NK_ASSERT(ctx->current->layout);
17368     if (!ctx || !ctx->current || !ctx->current->layout)
17369         return 0;
17370
17371     body.x = header.x;
17372     body.w = size.x;
17373     body.y = header.y + header.h;
17374     body.h = size.y;
17375
17376     popup = win->popup.win;
17377     is_open = popup ? nk_true : nk_false;
17378     is_active = (popup && (win->popup.name == hash) && win->popup.type == NK_PANEL_MENU);
17379     if ((is_clicked && is_open && !is_active) || (is_open && !is_active) ||
17380         (!is_open && !is_active && !is_clicked)) return 0;
17381     if (!nk_nonblock_begin(ctx, NK_WINDOW_NO_SCROLLBAR, body, header, NK_PANEL_MENU))
17382         return 0;
17383
17384     win->popup.type = NK_PANEL_MENU;
17385     win->popup.name = hash;
17386     return 1;
17387 }
17388 NK_API int
17389 nk_menu_begin_text(struct nk_context *ctx, const char *title, int len,
17390     nk_flags align, struct nk_vec2 size)
17391 {
17392     struct nk_window *win;
17393     const struct nk_input *in;
17394     struct nk_rect header;
17395     int is_clicked = nk_false;
17396     nk_flags state;
17397
17398     NK_ASSERT(ctx);
17399     NK_ASSERT(ctx->current);
17400     NK_ASSERT(ctx->current->layout);
17401     if (!ctx || !ctx->current || !ctx->current->layout)
17402         return 0;
17403
17404     win = ctx->current;
17405     state = nk_widget(&header, ctx);
17406     if (!state) return 0;
17407     in = (state == NK_WIDGET_ROM || win->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
17408     if (nk_do_button_text(&ctx->last_widget_state, &win->buffer, header,
17409         title, len, align, NK_BUTTON_DEFAULT, &ctx->style.menu_button, in, ctx->style.font))
17410         is_clicked = nk_true;
17411     return nk_menu_begin(ctx, win, title, is_clicked, header, size);
17412 }
17413 NK_API int nk_menu_begin_label(struct nk_context *ctx,
17414     const char *text, nk_flags align, struct nk_vec2 size)
17415 {
17416     return nk_menu_begin_text(ctx, text, nk_strlen(text), align, size);
17417 }
17418 NK_API int
```

```
17419 nk_menu_begin_image(struct nk_context *ctx, const char *id, struct nk_image img,
17420     struct nk_vec2 size)
17421 {
17422     struct nk_window *win;
17423     struct nk_rect header;
17424     const struct nk_input *in;
17425     int is_clicked = nk_false;
17426     nk_flags state;
17427
17428     NK_ASSERT(ctx);
17429     NK_ASSERT(ctx->current);
17430     NK_ASSERT(ctx->current->layout);
17431     if (!ctx || !ctx->current || !ctx->current->layout)
17432         return 0;
17433
17434     win = ctx->current;
17435     state = nk_widget(&header, ctx);
17436     if (!state) return 0;
17437     in = (state == NK_WIDGET_ROM || win->layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
17438     if (nk_do_button_image(&ctx->last_widget_state, &win->buffer, header,
17439         img, NK_BUTTON_DEFAULT, &ctx->style.menu_button, in))
17440         is_clicked = nk_true;
17441     return nk_menu_begin(ctx, win, id, is_clicked, header, size);
17442 }
17443 NK_API int
17444 nk_menu_begin_symbol(struct nk_context *ctx, const char *id,
17445     enum nk_symbol_type sym, struct nk_vec2 size)
17446 {
17447     struct nk_window *win;
17448     const struct nk_input *in;
17449     struct nk_rect header;
17450     int is_clicked = nk_false;
17451     nk_flags state;
17452
17453     NK_ASSERT(ctx);
17454     NK_ASSERT(ctx->current);
17455     NK_ASSERT(ctx->current->layout);
17456     if (!ctx || !ctx->current || !ctx->current->layout)
17457         return 0;
17458
17459     win = ctx->current;
17460     state = nk_widget(&header, ctx);
17461     if (!state) return 0;
17462     in = (state == NK_WIDGET_ROM || win->layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
17463     if (nk_do_button_symbol(&ctx->last_widget_state,  &win->buffer, header,
17464         sym, NK_BUTTON_DEFAULT, &ctx->style.menu_button, in, ctx->style.font))
17465         is_clicked = nk_true;
17466     return nk_menu_begin(ctx, win, id, is_clicked, header, size);
17467 }
17468 NK_API int
17469 nk_menu_begin_image_text(struct nk_context *ctx, const char *title, int len,
17470     nk_flags align, struct nk_image img, struct nk_vec2 size)
17471 {
17472     struct nk_window *win;
17473     struct nk_rect header;
17474     const struct nk_input *in;
17475     int is_clicked = nk_false;
17476     nk_flags state;
17477
17478     NK_ASSERT(ctx);
17479     NK_ASSERT(ctx->current);
17480     NK_ASSERT(ctx->current->layout);
17481     if (!ctx || !ctx->current || !ctx->current->layout)
17482         return 0;
17483
17484     win = ctx->current;
17485     state = nk_widget(&header, ctx);
17486     if (!state) return 0;
17487     in = (state == NK_WIDGET_ROM || win->layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
17488     if (nk_do_button_text_image(&ctx->last_widget_state, &win->buffer,
17489         header, img, title, len, align, NK_BUTTON_DEFAULT, &ctx->style.menu_button,
17490         ctx->style.font, in))
17491         is_clicked = nk_true;
17492     return nk_menu_begin(ctx, win, title, is_clicked, header, size);
17493 }
17494 NK_API int
17495 nk_menu_begin_image_label(struct nk_context *ctx,
17496     const char *title, nk_flags align, struct nk_image img, struct nk_vec2 size)
17497 {
17498     return nk_menu_begin_image_text(ctx, title, nk_strlen(title), align, img, size);
17499 }
17500 NK_API int
17501 nk_menu_begin_symbol_text(struct nk_context *ctx, const char *title, int len,
17502     nk_flags align, enum nk_symbol_type sym, struct nk_vec2 size)
17503 {
17504     struct nk_window *win;
17505     struct nk_rect header;
```

```
17506     const struct nk_input *in;
17507     int is_clicked = nk_false;
17508     nk_flags state;
17509
17510     NK_ASSERT(ctx);
17511     NK_ASSERT(ctx->current);
17512     NK_ASSERT(ctx->current->layout);
17513     if (!ctx || !ctx->current || !ctx->current->layout)
17514         return 0;
17515
17516     win = ctx->current;
17517     state = nk_widget(&header, ctx);
17518     if (!state) return 0;
17519
17520     in = (state == NK_WIDGET_ROM || win->layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
17521     if (nk_do_button_text_symbol(&ctx->last_widget_state, &win->buffer,
17522         header, sym, title, len, align, NK_BUTTON_DEFAULT, &ctx->style.menu_button,
17523         ctx->style.font, in)) is_clicked = nk_true;
17524     return nk_menu_begin(ctx, win, title, is_clicked, header, size);
17525 }
17526 NK_API int
17527 nk_menu_begin_symbol_label(struct nk_context *ctx,
17528     const char *title, nk_flags align, enum nk_symbol_type sym, struct nk_vec2 size )
17529 {
17530     return nk_menu_begin_symbol_text(ctx, title, nk_strlen(title), align,sym,size);
17531 }
17532 NK_API int
17533 nk_menu_item_text(struct nk_context *ctx, const char *title, int len, nk_flags align)
17534 {
17535     return nk_contextual_item_text(ctx, title, len, align);
17536 }
17537 NK_API int
17538 nk_menu_item_label(struct nk_context *ctx, const char *label, nk_flags align)
17539 {
17540     return nk_contextual_item_label(ctx, label, align);
17541 }
17542 NK_API int
17543 nk_menu_item_image_label(struct nk_context *ctx, struct nk_image img,
17544     const char *label, nk_flags align)
17545 {
17546     return nk_contextual_item_image_label(ctx, img, label, align);
17547 }
17548 NK_API int
17549 nk_menu_item_image_text(struct nk_context *ctx, struct nk_image img,
17550     const char *text, int len, nk_flags align)
17551 {
17552     return nk_contextual_item_image_text(ctx, img, text, len, align);
17553 }
17554 NK_API int nk_menu_item_symbol_text(struct nk_context *ctx, enum nk_symbol_type sym,
17555     const char *text, int len, nk_flags align)
17556 {
17557     return nk_contextual_item_symbol_text(ctx, sym, text, len, align);
17558 }
17559 NK_API int nk_menu_item_symbol_label(struct nk_context *ctx, enum nk_symbol_type sym,
17560     const char *label, nk_flags align)
17561 {
17562     return nk_contextual_item_symbol_label(ctx, sym, label, align);
17563 }
17564 NK_API void nk_menu_close(struct nk_context *ctx)
17565 {
17566     nk_contextual_close(ctx);
17567 }
17568 NK_API void
17569 nk_menu_end(struct nk_context *ctx)
17570 {
17571     nk_contextual_end(ctx);
17572 }
17573
17574
17575
17576
17577
17578 /* ===============================================================
17579  *
17580  *                          LAYOUT
17581  *
17582  * ===============================================================*/
17583 NK_API void
17584 nk_layout_set_min_row_height(struct nk_context *ctx, float height)
17585 {
17586     struct nk_window *win;
17587     struct nk_panel *layout;
17588
17589     NK_ASSERT(ctx);
17590     NK_ASSERT(ctx->current);
17591     NK_ASSERT(ctx->current->layout);
17592     if (!ctx || !ctx->current || !ctx->current->layout)
```

```
17593         return;
17594
17595     win = ctx->current;
17596     layout = win->layout;
17597     layout->row.min_height = height;
17598 }
17599 NK_API void
17600 nk_layout_reset_min_row_height(struct nk_context *ctx)
17601 {
17602     struct nk_window *win;
17603     struct nk_panel *layout;
17604
17605     NK_ASSERT(ctx);
17606     NK_ASSERT(ctx->current);
17607     NK_ASSERT(ctx->current->layout);
17608     if (!ctx || !ctx->current || !ctx->current->layout)
17609         return;
17610
17611     win = ctx->current;
17612     layout = win->layout;
17613     layout->row.min_height = ctx->style.font->height;
17614     layout->row.min_height += ctx->style.text.padding.y*2;
17615     layout->row.min_height += ctx->style.window.min_row_height_padding*2;
17616 }
17617 NK_LIB float
17618 nk_layout_row_calculate_usable_space(const struct nk_style *style, enum nk_panel_type type,
17619     float total_space, int columns)
17620 {
17621     float panel_padding;
17622     float panel_spacing;
17623     float panel_space;
17624
17625     struct nk_vec2 spacing;
17626     struct nk_vec2 padding;
17627
17628     spacing = style->window.spacing;
17629     padding = nk_panel_get_padding(style, type);
17630
17631     /* calculate the usable panel space */
17632     panel_padding = 2 * padding.x;
17633     panel_spacing = (float)NK_MAX(columns - 1, 0) * spacing.x;
17634     panel_space  = total_space - panel_padding - panel_spacing;
17635     return panel_space;
17636 }
17637 NK_LIB void
17638 nk_panel_layout(const struct nk_context *ctx, struct nk_window *win,
17639     float height, int cols)
17640 {
17641     struct nk_panel *layout;
17642     const struct nk_style *style;
17643     struct nk_command_buffer *out;
17644
17645     struct nk_vec2 item_spacing;
17646     struct nk_color color;
17647
17648     NK_ASSERT(ctx);
17649     NK_ASSERT(ctx->current);
17650     NK_ASSERT(ctx->current->layout);
17651     if (!ctx || !ctx->current || !ctx->current->layout)
17652         return;
17653
17654     /* prefetch some configuration data */
17655     layout = win->layout;
17656     style = &ctx->style;
17657     out = &win->buffer;
17658     color = style->window.background;
17659     item_spacing = style->window.spacing;
17660
17661     /*  if one of these triggers you forgot to add an `if` condition around either
17662         a window, group, popup, combobox or contextual menu `begin` and `end` block.
17663         Example:
17664             if (nk_begin(...) {...} nk_end(...); or
17665             if (nk_group_begin(...) { nk_group_end(...);} */
17666     NK_ASSERT(!(layout->flags & NK_WINDOW_MINIMIZED));
17667     NK_ASSERT(!(layout->flags & NK_WINDOW_HIDDEN));
17668     NK_ASSERT(!(layout->flags & NK_WINDOW_CLOSED));
17669
17670     /* update the current row and set the current row layout */
17671     layout->row.index = 0;
17672     layout->at_y += layout->row.height;
17673     layout->row.columns = cols;
17674     if (height == 0.0f)
17675         layout->row.height = NK_MAX(height, layout->row.min_height) + item_spacing.y;
17676     else layout->row.height = height + item_spacing.y;
17677
17678     layout->row.item_offset = 0;
17679     if (layout->flags & NK_WINDOW_DYNAMIC) {
```

```
17680         /* draw background for dynamic panels */
17681         struct nk_rect background;
17682         background.x = win->bounds.x;
17683         background.w = win->bounds.w;
17684         background.y = layout->at_y - 1.0f;
17685         background.h = layout->row.height + 1.0f;
17686         nk_fill_rect(out, background, 0, color);
17687     }
17688 }
17689 NK_LIB void
17690 nk_row_layout(struct nk_context *ctx, enum nk_layout_format fmt,
17691     float height, int cols, int width)
17692 {
17693     /* update the current row and set the current row layout */
17694     struct nk_window *win;
17695     NK_ASSERT(ctx);
17696     NK_ASSERT(ctx->current);
17697     NK_ASSERT(ctx->current->layout);
17698     if (!ctx || !ctx->current || !ctx->current->layout)
17699         return;
17700
17701     win = ctx->current;
17702     nk_panel_layout(ctx, win, height, cols);
17703     if (fmt == NK_DYNAMIC)
17704         win->layout->row.type = NK_LAYOUT_DYNAMIC_FIXED;
17705     else win->layout->row.type = NK_LAYOUT_STATIC_FIXED;
17706
17707     win->layout->row.ratio = 0;
17708     win->layout->row.filled = 0;
17709     win->layout->row.item_offset = 0;
17710     win->layout->row.item_width = (float)width;
17711 }
17712 NK_API float
17713 nk_layout_ratio_from_pixel(struct nk_context *ctx, float pixel_width)
17714 {
17715     struct nk_window *win;
17716     NK_ASSERT(ctx);
17717     NK_ASSERT(pixel_width);
17718     if (!ctx || !ctx->current || !ctx->current->layout) return 0;
17719     win = ctx->current;
17720     return NK_CLAMP(0.0f, pixel_width/win->bounds.x, 1.0f);
17721 }
17722 NK_API void
17723 nk_layout_row_dynamic(struct nk_context *ctx, float height, int cols)
17724 {
17725     nk_row_layout(ctx, NK_DYNAMIC, height, cols, 0);
17726 }
17727 NK_API void
17728 nk_layout_row_static(struct nk_context *ctx, float height, int item_width, int cols)
17729 {
17730     nk_row_layout(ctx, NK_STATIC, height, cols, item_width);
17731 }
17732 NK_API void
17733 nk_layout_row_begin(struct nk_context *ctx, enum nk_layout_format fmt,
17734     float row_height, int cols)
17735 {
17736     struct nk_window *win;
17737     struct nk_panel *layout;
17738
17739     NK_ASSERT(ctx);
17740     NK_ASSERT(ctx->current);
17741     NK_ASSERT(ctx->current->layout);
17742     if (!ctx || !ctx->current || !ctx->current->layout)
17743         return;
17744
17745     win = ctx->current;
17746     layout = win->layout;
17747     nk_panel_layout(ctx, win, row_height, cols);
17748     if (fmt == NK_DYNAMIC)
17749         layout->row.type = NK_LAYOUT_DYNAMIC_ROW;
17750     else layout->row.type = NK_LAYOUT_STATIC_ROW;
17751
17752     layout->row.ratio = 0;
17753     layout->row.filled = 0;
17754     layout->row.item_width = 0;
17755     layout->row.item_offset = 0;
17756     layout->row.columns = cols;
17757 }
17758 NK_API void
17759 nk_layout_row_push(struct nk_context *ctx, float ratio_or_width)
17760 {
17761     struct nk_window *win;
17762     struct nk_panel *layout;
17763
17764     NK_ASSERT(ctx);
17765     NK_ASSERT(ctx->current);
17766     NK_ASSERT(ctx->current->layout);
```

```
17767     if (!ctx || !ctx->current || !ctx->current->layout)
17768         return;
17769
17770     win = ctx->current;
17771     layout = win->layout;
17772     NK_ASSERT(layout->row.type == NK_LAYOUT_STATIC_ROW || layout->row.type == NK_LAYOUT_DYNAMIC_ROW);
17773     if (layout->row.type != NK_LAYOUT_STATIC_ROW && layout->row.type != NK_LAYOUT_DYNAMIC_ROW)
17774         return;
17775
17776     if (layout->row.type == NK_LAYOUT_DYNAMIC_ROW) {
17777         float ratio = ratio_or_width;
17778         if ((ratio + layout->row.filled) > 1.0f) return;
17779         if (ratio > 0.0f)
17780             layout->row.item_width = NK_SATURATE(ratio);
17781         else layout->row.item_width = 1.0f - layout->row.filled;
17782     } else layout->row.item_width = ratio_or_width;
17783 }
17784 NK_API void
17785 nk_layout_row_end(struct nk_context *ctx)
17786 {
17787     struct nk_window *win;
17788     struct nk_panel *layout;
17789
17790     NK_ASSERT(ctx);
17791     NK_ASSERT(ctx->current);
17792     NK_ASSERT(ctx->current->layout);
17793     if (!ctx || !ctx->current || !ctx->current->layout)
17794         return;
17795
17796     win = ctx->current;
17797     layout = win->layout;
17798     NK_ASSERT(layout->row.type == NK_LAYOUT_STATIC_ROW || layout->row.type == NK_LAYOUT_DYNAMIC_ROW);
17799     if (layout->row.type != NK_LAYOUT_STATIC_ROW && layout->row.type != NK_LAYOUT_DYNAMIC_ROW)
17800         return;
17801     layout->row.item_width = 0;
17802     layout->row.item_offset = 0;
17803 }
17804 NK_API void
17805 nk_layout_row(struct nk_context *ctx, enum nk_layout_format fmt,
17806     float height, int cols, const float *ratio)
17807 {
17808     int i;
17809     int n_undef = 0;
17810     struct nk_window *win;
17811     struct nk_panel *layout;
17812
17813     NK_ASSERT(ctx);
17814     NK_ASSERT(ctx->current);
17815     NK_ASSERT(ctx->current->layout);
17816     if (!ctx || !ctx->current || !ctx->current->layout)
17817         return;
17818
17819     win = ctx->current;
17820     layout = win->layout;
17821     nk_panel_layout(ctx, win, height, cols);
17822     if (fmt == NK_DYNAMIC) {
17823         /* calculate width of undefined widget ratios */
17824         float r = 0;
17825         layout->row.ratio = ratio;
17826         for (i = 0; i < cols; ++i) {
17827             if (ratio[i] < 0.0f)
17828                 n_undef++;
17829             else r += ratio[i];
17830         }
17831         r = NK_SATURATE(1.0f - r);
17832         layout->row.type = NK_LAYOUT_DYNAMIC;
17833         layout->row.item_width = (r > 0 && n_undef > 0) ? (r / (float)n_undef):0;
17834     } else {
17835         layout->row.ratio = ratio;
17836         layout->row.type = NK_LAYOUT_STATIC;
17837         layout->row.item_width = 0;
17838         layout->row.item_offset = 0;
17839     }
17840     layout->row.item_offset = 0;
17841     layout->row.filled = 0;
17842 }
17843 NK_API void
17844 nk_layout_row_template_begin(struct nk_context *ctx, float height)
17845 {
17846     struct nk_window *win;
17847     struct nk_panel *layout;
17848
17849     NK_ASSERT(ctx);
17850     NK_ASSERT(ctx->current);
17851     NK_ASSERT(ctx->current->layout);
17852     if (!ctx || !ctx->current || !ctx->current->layout)
17853         return;
```

```
17854
17855     win = ctx->current;
17856     layout = win->layout;
17857     nk_panel_layout(ctx, win, height, 1);
17858     layout->row.type = NK_LAYOUT_TEMPLATE;
17859     layout->row.columns = 0;
17860     layout->row.ratio = 0;
17861     layout->row.item_width = 0;
17862     layout->row.item_height = 0;
17863     layout->row.item_offset = 0;
17864     layout->row.filled = 0;
17865     layout->row.item.x = 0;
17866     layout->row.item.y = 0;
17867     layout->row.item.w = 0;
17868     layout->row.item.h = 0;
17869 }
17870 NK_API void
17871 nk_layout_row_template_push_dynamic(struct nk_context *ctx)
17872 {
17873     struct nk_window *win;
17874     struct nk_panel *layout;
17875
17876     NK_ASSERT(ctx);
17877     NK_ASSERT(ctx->current);
17878     NK_ASSERT(ctx->current->layout);
17879     if (!ctx || !ctx->current || !ctx->current->layout)
17880         return;
17881
17882     win = ctx->current;
17883     layout = win->layout;
17884     NK_ASSERT(layout->row.type == NK_LAYOUT_TEMPLATE);
17885     NK_ASSERT(layout->row.columns < NK_MAX_LAYOUT_ROW_TEMPLATE_COLUMNS);
17886     if (layout->row.type != NK_LAYOUT_TEMPLATE) return;
17887     if (layout->row.columns >= NK_MAX_LAYOUT_ROW_TEMPLATE_COLUMNS) return;
17888     layout->row.templates[layout->row.columns++] = -1.0f;
17889 }
17890 NK_API void
17891 nk_layout_row_template_push_variable(struct nk_context *ctx, float min_width)
17892 {
17893     struct nk_window *win;
17894     struct nk_panel *layout;
17895
17896     NK_ASSERT(ctx);
17897     NK_ASSERT(ctx->current);
17898     NK_ASSERT(ctx->current->layout);
17899     if (!ctx || !ctx->current || !ctx->current->layout)
17900         return;
17901
17902     win = ctx->current;
17903     layout = win->layout;
17904     NK_ASSERT(layout->row.type == NK_LAYOUT_TEMPLATE);
17905     NK_ASSERT(layout->row.columns < NK_MAX_LAYOUT_ROW_TEMPLATE_COLUMNS);
17906     if (layout->row.type != NK_LAYOUT_TEMPLATE) return;
17907     if (layout->row.columns >= NK_MAX_LAYOUT_ROW_TEMPLATE_COLUMNS) return;
17908     layout->row.templates[layout->row.columns++] = -min_width;
17909 }
17910 NK_API void
17911 nk_layout_row_template_push_static(struct nk_context *ctx, float width)
17912 {
17913     struct nk_window *win;
17914     struct nk_panel *layout;
17915
17916     NK_ASSERT(ctx);
17917     NK_ASSERT(ctx->current);
17918     NK_ASSERT(ctx->current->layout);
17919     if (!ctx || !ctx->current || !ctx->current->layout)
17920         return;
17921
17922     win = ctx->current;
17923     layout = win->layout;
17924     NK_ASSERT(layout->row.type == NK_LAYOUT_TEMPLATE);
17925     NK_ASSERT(layout->row.columns < NK_MAX_LAYOUT_ROW_TEMPLATE_COLUMNS);
17926     if (layout->row.type != NK_LAYOUT_TEMPLATE) return;
17927     if (layout->row.columns >= NK_MAX_LAYOUT_ROW_TEMPLATE_COLUMNS) return;
17928     layout->row.templates[layout->row.columns++] = width;
17929 }
17930 NK_API void
17931 nk_layout_row_template_end(struct nk_context *ctx)
17932 {
17933     struct nk_window *win;
17934     struct nk_panel *layout;
17935
17936     int i = 0;
17937     int variable_count = 0;
17938     int min_variable_count = 0;
17939     float min_fixed_width = 0.0f;
17940     float total_fixed_width = 0.0f;
```

```
17941     float max_variable_width = 0.0f;
17942
17943     NK_ASSERT(ctx);
17944     NK_ASSERT(ctx->current);
17945     NK_ASSERT(ctx->current->layout);
17946     if (!ctx || !ctx->current || !ctx->current->layout)
17947         return;
17948
17949     win = ctx->current;
17950     layout = win->layout;
17951     NK_ASSERT(layout->row.type == NK_LAYOUT_TEMPLATE);
17952     if (layout->row.type != NK_LAYOUT_TEMPLATE) return;
17953     for (i = 0; i < layout->row.columns; ++i) {
17954         float width = layout->row.templates[i];
17955         if (width >= 0.0f) {
17956             total_fixed_width += width;
17957             min_fixed_width += width;
17958         } else if (width < -1.0f) {
17959             width = -width;
17960             total_fixed_width += width;
17961             max_variable_width = NK_MAX(max_variable_width, width);
17962             variable_count++;
17963         } else {
17964             min_variable_count++;
17965             variable_count++;
17966         }
17967     }
17968     if (variable_count) {
17969         float space = nk_layout_row_calculate_usable_space(&ctx->style, layout->type,
17970                                 layout->bounds.w, layout->row.columns);
17971         float var_width = (NK_MAX(space-min_fixed_width,0.0f)) / (float)variable_count;
17972         int enough_space = var_width >= max_variable_width;
17973         if (!enough_space)
17974             var_width = (NK_MAX(space-total_fixed_width,0)) / (float)min_variable_count;
17975         for (i = 0; i < layout->row.columns; ++i) {
17976             float *width = &layout->row.templates[i];
17977             *width = (*width >= 0.0f)? *width: (*width < -1.0f && !enough_space)? -(*width):
      var_width;
17978         }
17979     }
17980 }
17981 NK_API void
17982 nk_layout_space_begin(struct nk_context *ctx, enum nk_layout_format fmt,
17983     float height, int widget_count)
17984 {
17985     struct nk_window *win;
17986     struct nk_panel *layout;
17987
17988     NK_ASSERT(ctx);
17989     NK_ASSERT(ctx->current);
17990     NK_ASSERT(ctx->current->layout);
17991     if (!ctx || !ctx->current || !ctx->current->layout)
17992         return;
17993
17994     win = ctx->current;
17995     layout = win->layout;
17996     nk_panel_layout(ctx, win, height, widget_count);
17997     if (fmt == NK_STATIC)
17998         layout->row.type = NK_LAYOUT_STATIC_FREE;
17999     else layout->row.type = NK_LAYOUT_DYNAMIC_FREE;
18000
18001     layout->row.ratio = 0;
18002     layout->row.filled = 0;
18003     layout->row.item_width = 0;
18004     layout->row.item_offset = 0;
18005 }
18006 NK_API void
18007 nk_layout_space_end(struct nk_context *ctx)
18008 {
18009     struct nk_window *win;
18010     struct nk_panel *layout;
18011
18012     NK_ASSERT(ctx);
18013     NK_ASSERT(ctx->current);
18014     NK_ASSERT(ctx->current->layout);
18015     if (!ctx || !ctx->current || !ctx->current->layout)
18016         return;
18017
18018     win = ctx->current;
18019     layout = win->layout;
18020     layout->row.item_width = 0;
18021     layout->row.item_height = 0;
18022     layout->row.item_offset = 0;
18023     nk_zero(&layout->row.item, sizeof(layout->row.item));
18024 }
18025 NK_API void
18026 nk_layout_space_push(struct nk_context *ctx, struct nk_rect rect)
```

```
18027 {
18028     struct nk_window *win;
18029     struct nk_panel *layout;
18030
18031     NK_ASSERT(ctx);
18032     NK_ASSERT(ctx->current);
18033     NK_ASSERT(ctx->current->layout);
18034     if (!ctx || !ctx->current || !ctx->current->layout)
18035         return;
18036
18037     win = ctx->current;
18038     layout = win->layout;
18039     layout->row.item = rect;
18040 }
18041 NK_API struct nk_rect
18042 nk_layout_space_bounds(struct nk_context *ctx)
18043 {
18044     struct nk_rect ret;
18045     struct nk_window *win;
18046     struct nk_panel *layout;
18047
18048     NK_ASSERT(ctx);
18049     NK_ASSERT(ctx->current);
18050     NK_ASSERT(ctx->current->layout);
18051     win = ctx->current;
18052     layout = win->layout;
18053
18054     ret.x = layout->clip.x;
18055     ret.y = layout->clip.y;
18056     ret.w = layout->clip.w;
18057     ret.h = layout->row.height;
18058     return ret;
18059 }
18060 NK_API struct nk_rect
18061 nk_layout_widget_bounds(struct nk_context *ctx)
18062 {
18063     struct nk_rect ret;
18064     struct nk_window *win;
18065     struct nk_panel *layout;
18066
18067     NK_ASSERT(ctx);
18068     NK_ASSERT(ctx->current);
18069     NK_ASSERT(ctx->current->layout);
18070     win = ctx->current;
18071     layout = win->layout;
18072
18073     ret.x = layout->at_x;
18074     ret.y = layout->at_y;
18075     ret.w = layout->bounds.w - NK_MAX(layout->at_x - layout->bounds.x,0);
18076     ret.h = layout->row.height;
18077     return ret;
18078 }
18079 NK_API struct nk_vec2
18080 nk_layout_space_to_screen(struct nk_context *ctx, struct nk_vec2 ret)
18081 {
18082     struct nk_window *win;
18083     struct nk_panel *layout;
18084
18085     NK_ASSERT(ctx);
18086     NK_ASSERT(ctx->current);
18087     NK_ASSERT(ctx->current->layout);
18088     win = ctx->current;
18089     layout = win->layout;
18090
18091     ret.x += layout->at_x - (float)*layout->offset_x;
18092     ret.y += layout->at_y - (float)*layout->offset_y;
18093     return ret;
18094 }
18095 NK_API struct nk_vec2
18096 nk_layout_space_to_local(struct nk_context *ctx, struct nk_vec2 ret)
18097 {
18098     struct nk_window *win;
18099     struct nk_panel *layout;
18100
18101     NK_ASSERT(ctx);
18102     NK_ASSERT(ctx->current);
18103     NK_ASSERT(ctx->current->layout);
18104     win = ctx->current;
18105     layout = win->layout;
18106
18107     ret.x += -layout->at_x + (float)*layout->offset_x;
18108     ret.y += -layout->at_y + (float)*layout->offset_y;
18109     return ret;
18110 }
18111 NK_API struct nk_rect
18112 nk_layout_space_rect_to_screen(struct nk_context *ctx, struct nk_rect ret)
18113 {
```

```
18114     struct nk_window *win;
18115     struct nk_panel *layout;
18116
18117     NK_ASSERT(ctx);
18118     NK_ASSERT(ctx->current);
18119     NK_ASSERT(ctx->current->layout);
18120     win = ctx->current;
18121     layout = win->layout;
18122
18123     ret.x += layout->at_x - (float)*layout->offset_x;
18124     ret.y += layout->at_y - (float)*layout->offset_y;
18125     return ret;
18126 }
18127 NK_API struct nk_rect
18128 nk_layout_space_rect_to_local(struct nk_context *ctx, struct nk_rect ret)
18129 {
18130     struct nk_window *win;
18131     struct nk_panel *layout;
18132
18133     NK_ASSERT(ctx);
18134     NK_ASSERT(ctx->current);
18135     NK_ASSERT(ctx->current->layout);
18136     win = ctx->current;
18137     layout = win->layout;
18138
18139     ret.x += -layout->at_x + (float)*layout->offset_x;
18140     ret.y += -layout->at_y + (float)*layout->offset_y;
18141     return ret;
18142 }
18143 NK_LIB void
18144 nk_panel_alloc_row(const struct nk_context *ctx, struct nk_window *win)
18145 {
18146     struct nk_panel *layout = win->layout;
18147     struct nk_vec2 spacing = ctx->style.window.spacing;
18148     const float row_height = layout->row.height - spacing.y;
18149     nk_panel_layout(ctx, win, row_height, layout->row.columns);
18150 }
18151 NK_LIB void
18152 nk_layout_widget_space(struct nk_rect *bounds, const struct nk_context *ctx,
18153     struct nk_window *win, int modify)
18154 {
18155     struct nk_panel *layout;
18156     const struct nk_style *style;
18157
18158     struct nk_vec2 spacing;
18159     struct nk_vec2 padding;
18160
18161     float item_offset = 0;
18162     float item_width = 0;
18163     float item_spacing = 0;
18164     float panel_space = 0;
18165
18166     NK_ASSERT(ctx);
18167     NK_ASSERT(ctx->current);
18168     NK_ASSERT(ctx->current->layout);
18169     if (!ctx || !ctx->current || !ctx->current->layout)
18170         return;
18171
18172     win = ctx->current;
18173     layout = win->layout;
18174     style = &ctx->style;
18175     NK_ASSERT(bounds);
18176
18177     spacing = style->window.spacing;
18178     padding = nk_panel_get_padding(style, layout->type);
18179     panel_space = nk_layout_row_calculate_usable_space(&ctx->style, layout->type,
18180                                             layout->bounds.w, layout->row.columns);
18181
18182     #define NK_FRAC(x) (x - (int)x) /* will be used to remove fookin gaps */
18183     /* calculate the width of one item inside the current layout space */
18184     switch (layout->row.type) {
18185     case NK_LAYOUT_DYNAMIC_FIXED: {
18186         /* scaling fixed size widgets item width */
18187         float w = NK_MAX(1.0f,panel_space) / (float)layout->row.columns;
18188         item_offset = (float)layout->row.index * w;
18189         item_width = w + NK_FRAC(item_offset);
18190         item_spacing = (float)layout->row.index * spacing.x;
18191     } break;
18192     case NK_LAYOUT_DYNAMIC_ROW: {
18193         /* scaling single ratio widget width */
18194         float w = layout->row.item_width * panel_space;
18195         item_offset = layout->row.item_offset;
18196         item_width = w + NK_FRAC(item_offset);
18197         item_spacing = 0;
18198
18199         if (modify) {
18200             layout->row.item_offset += w + spacing.x;
```

```
18201             layout->row.filled += layout->row.item_width;
18202             layout->row.index = 0;
18203         }
18204     } break;
18205     case NK_LAYOUT_DYNAMIC_FREE: {
18206         /* panel width depended free widget placing */
18207         bounds->x = layout->at_x + (layout->bounds.w * layout->row.item.x);
18208         bounds->x -= (float)*layout->offset_x;
18209         bounds->y = layout->at_y + (layout->row.height * layout->row.item.y);
18210         bounds->y -= (float)*layout->offset_y;
18211         bounds->w = layout->bounds.w  * layout->row.item.w + NK_FRAC(bounds->x);
18212         bounds->h = layout->row.height * layout->row.item.h + NK_FRAC(bounds->y);
18213         return;
18214     }
18215     case NK_LAYOUT_DYNAMIC: {
18216         /* scaling arrays of panel width ratios for every widget */
18217         float ratio, w;
18218         NK_ASSERT(layout->row.ratio);
18219         ratio = (layout->row.ratio[layout->row.index] < 0) ?
18220             layout->row.item_width : layout->row.ratio[layout->row.index];
18221
18222         w = (ratio * panel_space);
18223         item_spacing = (float)layout->row.index * spacing.x;
18224         item_offset = layout->row.item_offset;
18225         item_width = w + NK_FRAC(item_offset);
18226
18227         if (modify) {
18228             layout->row.item_offset += w;
18229             layout->row.filled += ratio;
18230         }
18231     } break;
18232     case NK_LAYOUT_STATIC_FIXED: {
18233         /* non-scaling fixed widgets item width */
18234         item_width = layout->row.item_width;
18235         item_offset = (float)layout->row.index * item_width;
18236         item_spacing = (float)layout->row.index * spacing.x;
18237     } break;
18238     case NK_LAYOUT_STATIC_ROW: {
18239         /* scaling single ratio widget width */
18240         item_width = layout->row.item_width;
18241         item_offset = layout->row.item_offset;
18242         item_spacing = (float)layout->row.index * spacing.x;
18243         if (modify) layout->row.item_offset += item_width;
18244     } break;
18245     case NK_LAYOUT_STATIC_FREE: {
18246         /* free widget placing */
18247         bounds->x = layout->at_x + layout->row.item.x;
18248         bounds->w = layout->row.item.w;
18249         if (((bounds->x + bounds->w) > layout->max_x) && modify)
18250             layout->max_x = (bounds->x + bounds->w);
18251         bounds->x -= (float)*layout->offset_x;
18252         bounds->y = layout->at_y + layout->row.item.y;
18253         bounds->y -= (float)*layout->offset_y;
18254         bounds->h = layout->row.item.h;
18255         return;
18256     }
18257     case NK_LAYOUT_STATIC: {
18258         /* non-scaling array of panel pixel width for every widget */
18259         item_spacing = (float)layout->row.index * spacing.x;
18260         item_width = layout->row.ratio[layout->row.index];
18261         item_offset = layout->row.item_offset;
18262         if (modify) layout->row.item_offset += item_width;
18263     } break;
18264     case NK_LAYOUT_TEMPLATE: {
18265         /* stretchy row layout with combined dynamic/static widget width*/
18266         float w;
18267         NK_ASSERT(layout->row.index < layout->row.columns);
18268         NK_ASSERT(layout->row.index < NK_MAX_LAYOUT_ROW_TEMPLATE_COLUMNS);
18269         w = layout->row.templates[layout->row.index];
18270         item_offset = layout->row.item_offset;
18271         item_width = w + NK_FRAC(item_offset);
18272         item_spacing = (float)layout->row.index * spacing.x;
18273         if (modify) layout->row.item_offset += w;
18274     } break;
18275     #undef NK_FRAC
18276     default: NK_ASSERT(0); break;
18277     };
18278
18279     /* set the bounds of the newly allocated widget */
18280     bounds->w = item_width;
18281     bounds->h = layout->row.height - spacing.y;
18282     bounds->y = layout->at_y - (float)*layout->offset_y;
18283     bounds->x = layout->at_x + item_offset + item_spacing + padding.x;
18284     if (((bounds->x + bounds->w) > layout->max_x) && modify)
18285         layout->max_x = bounds->x + bounds->w;
18286     bounds->x -= (float)*layout->offset_x;
18287 }
```

```
18288 NK_LIB void
18289 nk_panel_alloc_space(struct nk_rect *bounds, const struct nk_context *ctx)
18290 {
18291     struct nk_window *win;
18292     struct nk_panel *layout;
18293
18294     NK_ASSERT(ctx);
18295     NK_ASSERT(ctx->current);
18296     NK_ASSERT(ctx->current->layout);
18297     if (!ctx || !ctx->current || !ctx->current->layout)
18298         return;
18299
18300     /* check if the end of the row has been hit and begin new row if so */
18301     win = ctx->current;
18302     layout = win->layout;
18303     if (layout->row.index >= layout->row.columns)
18304         nk_panel_alloc_row(ctx, win);
18305
18306     /* calculate widget position and size */
18307     nk_layout_widget_space(bounds, ctx, win, nk_true);
18308     layout->row.index++;
18309 }
18310 NK_LIB void
18311 nk_layout_peek(struct nk_rect *bounds, struct nk_context *ctx)
18312 {
18313     float y;
18314     int index;
18315     struct nk_window *win;
18316     struct nk_panel *layout;
18317
18318     NK_ASSERT(ctx);
18319     NK_ASSERT(ctx->current);
18320     NK_ASSERT(ctx->current->layout);
18321     if (!ctx || !ctx->current || !ctx->current->layout)
18322         return;
18323
18324     win = ctx->current;
18325     layout = win->layout;
18326     y = layout->at_y;
18327     index = layout->row.index;
18328     if (layout->row.index >= layout->row.columns) {
18329         layout->at_y += layout->row.height;
18330         layout->row.index = 0;
18331     }
18332     nk_layout_widget_space(bounds, ctx, win, nk_false);
18333     if (!layout->row.index) {
18334         bounds->x -= layout->row.item_offset;
18335     }
18336     layout->at_y = y;
18337     layout->row.index = index;
18338 }
18339
18340
18341
18342
18343
18344 /* ===============================================================
18345  *
18346  *                              TREE
18347  *
18348  * ===============================================================*/
18349 NK_INTERN int
18350 nk_tree_state_base(struct nk_context *ctx, enum nk_tree_type type,
18351     struct nk_image *img, const char *title, enum nk_collapse_states *state)
18352 {
18353     struct nk_window *win;
18354     struct nk_panel *layout;
18355     const struct nk_style *style;
18356     struct nk_command_buffer *out;
18357     const struct nk_input *in;
18358     const struct nk_style_button *button;
18359     enum nk_symbol_type symbol;
18360     float row_height;
18361
18362     struct nk_vec2 item_spacing;
18363     struct nk_rect header = {0,0,0,0};
18364     struct nk_rect sym = {0,0,0,0};
18365     struct nk_text text;
18366
18367     nk_flags ws = 0;
18368     enum nk_widget_layout_states widget_state;
18369
18370     NK_ASSERT(ctx);
18371     NK_ASSERT(ctx->current);
18372     NK_ASSERT(ctx->current->layout);
18373     if (!ctx || !ctx->current || !ctx->current->layout)
18374         return 0;
```

```
18375
18376     /* cache some data */
18377     win = ctx->current;
18378     layout = win->layout;
18379     out = &win->buffer;
18380     style = &ctx->style;
18381     item_spacing = style->window.spacing;
18382
18383     /* calculate header bounds and draw background */
18384     row_height = style->font->height + 2 * style->tab.padding.y;
18385     nk_layout_set_min_row_height(ctx, row_height);
18386     nk_layout_row_dynamic(ctx, row_height, 1);
18387     nk_layout_reset_min_row_height(ctx);
18388
18389     widget_state = nk_widget(&header, ctx);
18390     if (type == NK_TREE_TAB) {
18391         const struct nk_style_item *background = &style->tab.background;
18392         if (background->type == NK_STYLE_ITEM_IMAGE) {
18393             nk_draw_image(out, header, &background->data.image, nk_white);
18394             text.background = nk_rgba(0,0,0,0);
18395         } else {
18396             text.background = background->data.color;
18397             nk_fill_rect(out, header, 0, style->tab.border_color);
18398             nk_fill_rect(out, nk_shrink_rect(header, style->tab.border),
18399                 style->tab.rounding, background->data.color);
18400         }
18401     } else text.background = style->window.background;
18402
18403     /* update node state */
18404     in = (!(layout->flags & NK_WINDOW_ROM)) ? &ctx->input: 0;
18405     in = (in && widget_state == NK_WIDGET_VALID) ? &ctx->input : 0;
18406     if (nk_button_behavior(&ws, header, in, NK_BUTTON_DEFAULT))
18407         *state = (*state == NK_MAXIMIZED) ? NK_MINIMIZED : NK_MAXIMIZED;
18408
18409     /* select correct button style */
18410     if (*state == NK_MAXIMIZED) {
18411         symbol = style->tab.sym_maximize;
18412         if (type == NK_TREE_TAB)
18413             button = &style->tab.tab_maximize_button;
18414         else button = &style->tab.node_maximize_button;
18415     } else {
18416         symbol = style->tab.sym_minimize;
18417         if (type == NK_TREE_TAB)
18418             button = &style->tab.tab_minimize_button;
18419         else button = &style->tab.node_minimize_button;
18420     }
18421
18422     {/* draw triangle button */
18423     sym.w = sym.h = style->font->height;
18424     sym.y = header.y + style->tab.padding.y;
18425     sym.x = header.x + style->tab.padding.x;
18426     nk_do_button_symbol(&ws, &win->buffer, sym, symbol, NK_BUTTON_DEFAULT,
18427         button, 0, style->font);
18428
18429     if (img) {
18430         /* draw optional image icon */
18431         sym.x = sym.x + sym.w + 4 * item_spacing.x;
18432         nk_draw_image(&win->buffer, sym, img, nk_white);
18433         sym.w = style->font->height + style->tab.spacing.x;}
18434     }
18435
18436     {/* draw label */
18437     struct nk_rect label;
18438     header.w = NK_MAX(header.w, sym.w + item_spacing.x);
18439     label.x = sym.x + sym.w + item_spacing.x;
18440     label.y = sym.y;
18441     label.w = header.w - (sym.w + item_spacing.y + style->tab.indent);
18442     label.h = style->font->height;
18443     text.text = style->tab.text;
18444     text.padding = nk_vec2(0,0);
18445     nk_widget_text(out, label, title, nk_strlen(title), &text,
18446         NK_TEXT_LEFT, style->font);}
18447
18448     /* increase x-axis cursor widget position pointer */
18449     if (*state == NK_MAXIMIZED) {
18450         layout->at_x = header.x + (float)*layout->offset_x + style->tab.indent;
18451         layout->bounds.w = NK_MAX(layout->bounds.w, style->tab.indent);
18452         layout->bounds.w -= (style->tab.indent + style->window.padding.x);
18453         layout->row.tree_depth++;
18454         return nk_true;
18455     } else return nk_false;
18456 }
18457 NK_INTERN int
18458 nk_tree_base(struct nk_context *ctx, enum nk_tree_type type,
18459     struct nk_image *img, const char *title, enum nk_collapse_states initial_state,
18460     const char *hash, int len, int line)
18461 {
```

```
18462     struct nk_window *win = ctx->current;
18463     int title_len = 0;
18464     nk_hash tree_hash = 0;
18465     nk_uint *state = 0;
18466
18467     /* retrieve tree state from internal widget state tables */
18468     if (!hash) {
18469         title_len = (int)nk_strlen(title);
18470         tree_hash = nk_murmur_hash(title, (int)title_len, (nk_hash)line);
18471     } else tree_hash = nk_murmur_hash(hash, len, (nk_hash)line);
18472     state = nk_find_value(win, tree_hash);
18473     if (!state) {
18474         state = nk_add_value(ctx, win, tree_hash, 0);
18475         *state = initial_state;
18476     }
18477     return nk_tree_state_base(ctx, type, img, title, (enum nk_collapse_states*)state);
18478 }
18479 NK_API int
18480 nk_tree_state_push(struct nk_context *ctx, enum nk_tree_type type,
18481     const char *title, enum nk_collapse_states *state)
18482 {
18483     return nk_tree_state_base(ctx, type, 0, title, state);
18484 }
18485 NK_API int
18486 nk_tree_state_image_push(struct nk_context *ctx, enum nk_tree_type type,
18487     struct nk_image img, const char *title, enum nk_collapse_states *state)
18488 {
18489     return nk_tree_state_base(ctx, type, &img, title, state);
18490 }
18491 NK_API void
18492 nk_tree_state_pop(struct nk_context *ctx)
18493 {
18494     struct nk_window *win = 0;
18495     struct nk_panel *layout = 0;
18496
18497     NK_ASSERT(ctx);
18498     NK_ASSERT(ctx->current);
18499     NK_ASSERT(ctx->current->layout);
18500     if (!ctx || !ctx->current || !ctx->current->layout)
18501         return;
18502
18503     win = ctx->current;
18504     layout = win->layout;
18505     layout->at_x -= ctx->style.tab.indent + ctx->style.window.padding.x;
18506     layout->bounds.w += ctx->style.tab.indent + ctx->style.window.padding.x;
18507     NK_ASSERT(layout->row.tree_depth);
18508     layout->row.tree_depth--;
18509 }
18510 NK_API int
18511 nk_tree_push_hashed(struct nk_context *ctx, enum nk_tree_type type,
18512     const char *title, enum nk_collapse_states initial_state,
18513     const char *hash, int len, int line)
18514 {
18515     return nk_tree_base(ctx, type, 0, title, initial_state, hash, len, line);
18516 }
18517 NK_API int
18518 nk_tree_image_push_hashed(struct nk_context *ctx, enum nk_tree_type type,
18519     struct nk_image img, const char *title, enum nk_collapse_states initial_state,
18520     const char *hash, int len,int seed)
18521 {
18522     return nk_tree_base(ctx, type, &img, title, initial_state, hash, len, seed);
18523 }
18524 NK_API void
18525 nk_tree_pop(struct nk_context *ctx)
18526 {
18527     nk_tree_state_pop(ctx);
18528 }
18529 NK_INTERN int
18530 nk_tree_element_image_push_hashed_base(struct nk_context *ctx, enum nk_tree_type type,
18531     struct nk_image *img, const char *title, int title_len,
18532     enum nk_collapse_states *state, int *selected)
18533 {
18534     struct nk_window *win;
18535     struct nk_panel *layout;
18536     const struct nk_style *style;
18537     struct nk_command_buffer *out;
18538     const struct nk_input *in;
18539     const struct nk_style_button *button;
18540     enum nk_symbol_type symbol;
18541     float row_height;
18542     struct nk_vec2 padding;
18543
18544     int text_len;
18545     float text_width;
18546
18547     struct nk_vec2 item_spacing;
18548     struct nk_rect header = {0,0,0,0};
```

```
18549     struct nk_rect sym = {0,0,0,0};
18550     struct nk_text text;
18551
18552     nk_flags ws = 0;
18553     enum nk_widget_layout_states widget_state;
18554
18555     NK_ASSERT(ctx);
18556     NK_ASSERT(ctx->current);
18557     NK_ASSERT(ctx->current->layout);
18558     if (!ctx || !ctx->current || !ctx->current->layout)
18559         return 0;
18560
18561     /* cache some data */
18562     win = ctx->current;
18563     layout = win->layout;
18564     out = &win->buffer;
18565     style = &ctx->style;
18566     item_spacing = style->window.spacing;
18567     padding = style->selectable.padding;
18568
18569     /* calculate header bounds and draw background */
18570     row_height = style->font->height + 2 * style->tab.padding.y;
18571     nk_layout_set_min_row_height(ctx, row_height);
18572     nk_layout_row_dynamic(ctx, row_height, 1);
18573     nk_layout_reset_min_row_height(ctx);
18574
18575     widget_state = nk_widget(&header, ctx);
18576     if (type == NK_TREE_TAB) {
18577         const struct nk_style_item *background = &style->tab.background;
18578         if (background->type == NK_STYLE_ITEM_IMAGE) {
18579             nk_draw_image(out, header, &background->data.image, nk_white);
18580             text.background = nk_rgba(0,0,0,0);
18581         } else {
18582             text.background = background->data.color;
18583             nk_fill_rect(out, header, 0, style->tab.border_color);
18584             nk_fill_rect(out, nk_shrink_rect(header, style->tab.border),
18585                 style->tab.rounding, background->data.color);
18586         }
18587     } else text.background = style->window.background;
18588
18589     in = (!(layout->flags & NK_WINDOW_ROM)) ? &ctx->input: 0;
18590     in = (in && widget_state == NK_WIDGET_VALID) ? &ctx->input : 0;
18591
18592     /* select correct button style */
18593     if (*state == NK_MAXIMIZED) {
18594         symbol = style->tab.sym_maximize;
18595         if (type == NK_TREE_TAB)
18596             button = &style->tab.tab_maximize_button;
18597         else button = &style->tab.node_maximize_button;
18598     } else {
18599         symbol = style->tab.sym_minimize;
18600         if (type == NK_TREE_TAB)
18601             button = &style->tab.tab_minimize_button;
18602         else button = &style->tab.node_minimize_button;
18603     }
18604     {/* draw triangle button */
18605     sym.w = sym.h = style->font->height;
18606     sym.y = header.y + style->tab.padding.y;
18607     sym.x = header.x + style->tab.padding.x;
18608     if (nk_do_button_symbol(&ws, &win->buffer, sym, symbol, NK_BUTTON_DEFAULT, button, in,
    style->font))
18609         *state = (*state == NK_MAXIMIZED) ? NK_MINIMIZED : NK_MAXIMIZED;}
18610
18611     /* draw label */
18612     {nk_flags dummy = 0;
18613     struct nk_rect label;
18614     /* calculate size of the text and tooltip */
18615     text_len = nk_strlen(title);
18616     text_width = style->font->width(style->font->userdata, style->font->height, title, text_len);
18617     text_width += (4 * padding.x);
18618
18619     header.w = NK_MAX(header.w, sym.w + item_spacing.x);
18620     label.x = sym.x + sym.w + item_spacing.x;
18621     label.y = sym.y;
18622     label.w = NK_MIN(header.w - (sym.w + item_spacing.y + style->tab.indent), text_width);
18623     label.h = style->font->height;
18624
18625     if (img) {
18626         nk_do_selectable_image(&dummy, &win->buffer, label, title, title_len, NK_TEXT_LEFT,
18627             selected, img, &style->selectable, in, style->font);
18628     } else nk_do_selectable(&dummy, &win->buffer, label, title, title_len, NK_TEXT_LEFT,
18629             selected, &style->selectable, in, style->font);
18630     }
18631     /* increase x-axis cursor widget position pointer */
18632     if (*state == NK_MAXIMIZED) {
18633         layout->at_x = header.x + (float)*layout->offset_x + style->tab.indent;
18634         layout->bounds.w = NK_MAX(layout->bounds.w, style->tab.indent);
```

```
18635         layout->bounds.w -= (style->tab.indent + style->window.padding.x);
18636         layout->row.tree_depth++;
18637         return nk_true;
18638     } else return nk_false;
18639 }
18640 NK_INTERN int
18641 nk_tree_element_base(struct nk_context *ctx, enum nk_tree_type type,
18642     struct nk_image *img, const char *title, enum nk_collapse_states initial_state,
18643     int *selected, const char *hash, int len, int line)
18644 {
18645     struct nk_window *win = ctx->current;
18646     int title_len = 0;
18647     nk_hash tree_hash = 0;
18648     nk_uint *state = 0;
18649
18650     /* retrieve tree state from internal widget state tables */
18651     if (!hash) {
18652         title_len = (int)nk_strlen(title);
18653         tree_hash = nk_murmur_hash(title, (int)title_len, (nk_hash)line);
18654     } else tree_hash = nk_murmur_hash(hash, len, (nk_hash)line);
18655     state = nk_find_value(win, tree_hash);
18656     if (!state) {
18657         state = nk_add_value(ctx, win, tree_hash, 0);
18658         *state = initial_state;
18659     } return nk_tree_element_image_push_hashed_base(ctx, type, img, title,
18660         nk_strlen(title), (enum nk_collapse_states*)state, selected);
18661 }
18662 NK_API int
18663 nk_tree_element_push_hashed(struct nk_context *ctx, enum nk_tree_type type,
18664     const char *title, enum nk_collapse_states initial_state,
18665     int *selected, const char *hash, int len, int seed)
18666 {
18667     return nk_tree_element_base(ctx, type, 0, title, initial_state, selected, hash, len, seed);
18668 }
18669 NK_API int
18670 nk_tree_element_image_push_hashed(struct nk_context *ctx, enum nk_tree_type type,
18671     struct nk_image img, const char *title, enum nk_collapse_states initial_state,
18672     int *selected, const char *hash, int len,int seed)
18673 {
18674     return nk_tree_element_base(ctx, type, &img, title, initial_state, selected, hash, len, seed);
18675 }
18676 NK_API void
18677 nk_tree_element_pop(struct nk_context *ctx)
18678 {
18679     nk_tree_state_pop(ctx);
18680 }
18681
18682
18683
18684
18685
18686 /* ===============================================================
18687  *
18688  *                          GROUP
18689  *
18690  * ===============================================================*/
18691 NK_API int
18692 nk_group_scrolled_offset_begin(struct nk_context *ctx,
18693     nk_uint *x_offset, nk_uint *y_offset, const char *title, nk_flags flags)
18694 {
18695     struct nk_rect bounds;
18696     struct nk_window panel;
18697     struct nk_window *win;
18698
18699     win = ctx->current;
18700     nk_panel_alloc_space(&bounds, ctx);
18701     {const struct nk_rect *c = &win->layout->clip;
18702     if (!NK_INTERSECT(c->x, c->y, c->w, c->h, bounds.x, bounds.y, bounds.w, bounds.h) &&
18703         !(flags & NK_WINDOW_MOVABLE)) {
18704         return 0;
18705     }}
18706     if (win->flags & NK_WINDOW_ROM)
18707         flags |= NK_WINDOW_ROM;
18708
18709     /* initialize a fake window to create the panel from */
18710     nk_zero(&panel, sizeof(panel));
18711     panel.bounds = bounds;
18712     panel.flags = flags;
18713     panel.scrollbar.x = *x_offset;
18714     panel.scrollbar.y = *y_offset;
18715     panel.buffer = win->buffer;
18716     panel.layout = (struct nk_panel*)nk_create_panel(ctx);
18717     ctx->current = &panel;
18718     nk_panel_begin(ctx, (flags & NK_WINDOW_TITLE) ? title: 0, NK_PANEL_GROUP);
18719
18720     win->buffer = panel.buffer;
18721     win->buffer.clip = panel.layout->clip;
```

```
18722     panel.layout->offset_x = x_offset;
18723     panel.layout->offset_y = y_offset;
18724     panel.layout->parent = win->layout;
18725     win->layout = panel.layout;
18726
18727     ctx->current = win;
18728     if ((panel.layout->flags & NK_WINDOW_CLOSED) ||
18729         (panel.layout->flags & NK_WINDOW_MINIMIZED))
18730     {
18731         nk_flags f = panel.layout->flags;
18732         nk_group_scrolled_end(ctx);
18733         if (f & NK_WINDOW_CLOSED)
18734             return NK_WINDOW_CLOSED;
18735         if (f & NK_WINDOW_MINIMIZED)
18736             return NK_WINDOW_MINIMIZED;
18737     }
18738     return 1;
18739 }
18740 NK_API void
18741 nk_group_scrolled_end(struct nk_context *ctx)
18742 {
18743     struct nk_window *win;
18744     struct nk_panel *parent;
18745     struct nk_panel *g;
18746
18747     struct nk_rect clip;
18748     struct nk_window pan;
18749     struct nk_vec2 panel_padding;
18750
18751     NK_ASSERT(ctx);
18752     NK_ASSERT(ctx->current);
18753     if (!ctx || !ctx->current)
18754         return;
18755
18756     /* make sure nk_group_begin was called correctly */
18757     NK_ASSERT(ctx->current);
18758     win = ctx->current;
18759     NK_ASSERT(win->layout);
18760     g = win->layout;
18761     NK_ASSERT(g->parent);
18762     parent = g->parent;
18763
18764     /* dummy window */
18765     nk_zero_struct(pan);
18766     panel_padding = nk_panel_get_padding(&ctx->style, NK_PANEL_GROUP);
18767     pan.bounds.y = g->bounds.y - (g->header_height + g->menu.h);
18768     pan.bounds.x = g->bounds.x - panel_padding.x;
18769     pan.bounds.w = g->bounds.w + 2 * panel_padding.x;
18770     pan.bounds.h = g->bounds.h + g->header_height + g->menu.h;
18771     if (g->flags & NK_WINDOW_BORDER) {
18772         pan.bounds.x -= g->border;
18773         pan.bounds.y -= g->border;
18774         pan.bounds.w += 2*g->border;
18775         pan.bounds.h += 2*g->border;
18776     }
18777     if (!(g->flags & NK_WINDOW_NO_SCROLLBAR)) {
18778         pan.bounds.w += ctx->style.window.scrollbar_size.x;
18779         pan.bounds.h += ctx->style.window.scrollbar_size.y;
18780     }
18781     pan.scrollbar.x = *g->offset_x;
18782     pan.scrollbar.y = *g->offset_y;
18783     pan.flags = g->flags;
18784     pan.buffer = win->buffer;
18785     pan.layout = g;
18786     pan.parent = win;
18787     ctx->current = &pan;
18788
18789     /* make sure group has correct clipping rectangle */
18790     nk_unify(&clip, &parent->clip, pan.bounds.x, pan.bounds.y,
18791         pan.bounds.x + pan.bounds.w, pan.bounds.y + pan.bounds.h + panel_padding.x);
18792     nk_push_scissor(&pan.buffer, clip);
18793     nk_end(ctx);
18794
18795     win->buffer = pan.buffer;
18796     nk_push_scissor(&win->buffer, parent->clip);
18797     ctx->current = win;
18798     win->layout = parent;
18799     g->bounds = pan.bounds;
18800     return;
18801 }
18802 NK_API int
18803 nk_group_scrolled_begin(struct nk_context *ctx,
18804     struct nk_scroll *scroll, const char *title, nk_flags flags)
18805 {
18806     return nk_group_scrolled_offset_begin(ctx, &scroll->x, &scroll->y, title, flags);
18807 }
18808 NK_API int
```

```
18809 nk_group_begin_titled(struct nk_context *ctx, const char *id,
18810     const char *title, nk_flags flags)
18811 {
18812     int id_len;
18813     nk_hash id_hash;
18814     struct nk_window *win;
18815     nk_uint *x_offset;
18816     nk_uint *y_offset;
18817
18818     NK_ASSERT(ctx);
18819     NK_ASSERT(id);
18820     NK_ASSERT(ctx->current);
18821     NK_ASSERT(ctx->current->layout);
18822     if (!ctx || !ctx->current || !ctx->current->layout || !id)
18823         return 0;
18824
18825     /* find persistent group scrollbar value */
18826     win = ctx->current;
18827     id_len = (int)nk_strlen(id);
18828     id_hash = nk_murmur_hash(id, (int)id_len, NK_PANEL_GROUP);
18829     x_offset = nk_find_value(win, id_hash);
18830     if (!x_offset) {
18831         x_offset = nk_add_value(ctx, win, id_hash, 0);
18832         y_offset = nk_add_value(ctx, win, id_hash+1, 0);
18833
18834         NK_ASSERT(x_offset);
18835         NK_ASSERT(y_offset);
18836         if (!x_offset || !y_offset) return 0;
18837         *x_offset = *y_offset = 0;
18838     } else y_offset = nk_find_value(win, id_hash+1);
18839     return nk_group_scrolled_offset_begin(ctx, x_offset, y_offset, title, flags);
18840 }
18841 NK_API int
18842 nk_group_begin(struct nk_context *ctx, const char *title, nk_flags flags)
18843 {
18844     return nk_group_begin_titled(ctx, title, title, flags);
18845 }
18846 NK_API void
18847 nk_group_end(struct nk_context *ctx)
18848 {
18849     nk_group_scrolled_end(ctx);
18850 }
18851 NK_API void
18852 nk_group_get_scroll(struct nk_context *ctx, const char *id, nk_uint *x_offset, nk_uint *y_offset)
18853 {
18854     int id_len;
18855     nk_hash id_hash;
18856     struct nk_window *win;
18857     nk_uint *x_offset_ptr;
18858     nk_uint *y_offset_ptr;
18859
18860     NK_ASSERT(ctx);
18861     NK_ASSERT(id);
18862     NK_ASSERT(ctx->current);
18863     NK_ASSERT(ctx->current->layout);
18864     if (!ctx || !ctx->current || !ctx->current->layout || !id)
18865         return;
18866
18867     /* find persistent group scrollbar value */
18868     win = ctx->current;
18869     id_len = (int)nk_strlen(id);
18870     id_hash = nk_murmur_hash(id, (int)id_len, NK_PANEL_GROUP);
18871     x_offset_ptr = nk_find_value(win, id_hash);
18872     if (!x_offset_ptr) {
18873         x_offset_ptr = nk_add_value(ctx, win, id_hash, 0);
18874         y_offset_ptr = nk_add_value(ctx, win, id_hash+1, 0);
18875
18876         NK_ASSERT(x_offset_ptr);
18877         NK_ASSERT(y_offset_ptr);
18878         if (!x_offset_ptr || !y_offset_ptr) return;
18879         *x_offset_ptr = *y_offset_ptr = 0;
18880     } else y_offset_ptr = nk_find_value(win, id_hash+1);
18881     if (x_offset)
18882       *x_offset = *x_offset_ptr;
18883     if (y_offset)
18884       *y_offset = *y_offset_ptr;
18885 }
18886 NK_API void
18887 nk_group_set_scroll(struct nk_context *ctx, const char *id, nk_uint x_offset, nk_uint y_offset)
18888 {
18889     int id_len;
18890     nk_hash id_hash;
18891     struct nk_window *win;
18892     nk_uint *x_offset_ptr;
18893     nk_uint *y_offset_ptr;
18894
18895     NK_ASSERT(ctx);
```

```
18896     NK_ASSERT(id);
18897     NK_ASSERT(ctx->current);
18898     NK_ASSERT(ctx->current->layout);
18899     if (!ctx || !ctx->current || !ctx->current->layout || !id)
18900         return;
18901
18902     /* find persistent group scrollbar value */
18903     win = ctx->current;
18904     id_len = (int)nk_strlen(id);
18905     id_hash = nk_murmur_hash(id, (int)id_len, NK_PANEL_GROUP);
18906     x_offset_ptr = nk_find_value(win, id_hash);
18907     if (!x_offset_ptr) {
18908         x_offset_ptr = nk_add_value(ctx, win, id_hash, 0);
18909         y_offset_ptr = nk_add_value(ctx, win, id_hash+1, 0);
18910
18911         NK_ASSERT(x_offset_ptr);
18912         NK_ASSERT(y_offset_ptr);
18913         if (!x_offset_ptr || !y_offset_ptr) return;
18914         *x_offset_ptr = *y_offset_ptr = 0;
18915     } else y_offset_ptr = nk_find_value(win, id_hash+1);
18916     *x_offset_ptr = x_offset;
18917     *y_offset_ptr = y_offset;
18918 }
18919
18920
18921
18922
18923 /* ===============================================================
18924  *
18925  *                          LIST VIEW
18926  *
18927  * ===============================================================*/
18928 NK_API int
18929 nk_list_view_begin(struct nk_context *ctx, struct nk_list_view *view,
18930     const char *title, nk_flags flags, int row_height, int row_count)
18931 {
18932     int title_len;
18933     nk_hash title_hash;
18934     nk_uint *x_offset;
18935     nk_uint *y_offset;
18936
18937     int result;
18938     struct nk_window *win;
18939     struct nk_panel *layout;
18940     const struct nk_style *style;
18941     struct nk_vec2 item_spacing;
18942
18943     NK_ASSERT(ctx);
18944     NK_ASSERT(view);
18945     NK_ASSERT(title);
18946     if (!ctx || !view || !title) return 0;
18947
18948     win = ctx->current;
18949     style = &ctx->style;
18950     item_spacing = style->window.spacing;
18951     row_height += NK_MAX(0, (int)item_spacing.y);
18952
18953     /* find persistent list view scrollbar offset */
18954     title_len = (int)nk_strlen(title);
18955     title_hash = nk_murmur_hash(title, (int)title_len, NK_PANEL_GROUP);
18956     x_offset = nk_find_value(win, title_hash);
18957     if (!x_offset) {
18958         x_offset = nk_add_value(ctx, win, title_hash, 0);
18959         y_offset = nk_add_value(ctx, win, title_hash+1, 0);
18960
18961         NK_ASSERT(x_offset);
18962         NK_ASSERT(y_offset);
18963         if (!x_offset || !y_offset) return 0;
18964         *x_offset = *y_offset = 0;
18965     } else y_offset = nk_find_value(win, title_hash+1);
18966     view->scroll_value = *y_offset;
18967     view->scroll_pointer = y_offset;
18968
18969     *y_offset = 0;
18970     result = nk_group_scrolled_offset_begin(ctx, x_offset, y_offset, title, flags);
18971     win = ctx->current;
18972     layout = win->layout;
18973
18974     view->total_height = row_height * NK_MAX(row_count,1);
18975     view->begin = (int)NK_MAX(((float)view->scroll_value / (float)row_height), 0.0f);
18976     view->count = (int)NK_MAX(nk_iceilf((layout->clip.h)/(float)row_height),0);
18977     view->count = NK_MIN(view->count, row_count - view->begin);
18978     view->end = view->begin + view->count;
18979     view->ctx = ctx;
18980     return result;
18981 }
18982 NK_API void
```

```
18983 nk_list_view_end(struct nk_list_view *view)
18984 {
18985     struct nk_context *ctx;
18986     struct nk_window *win;
18987     struct nk_panel *layout;
18988
18989     NK_ASSERT(view);
18990     NK_ASSERT(view->ctx);
18991     NK_ASSERT(view->scroll_pointer);
18992     if (!view || !view->ctx) return;
18993
18994     ctx = view->ctx;
18995     win = ctx->current;
18996     layout = win->layout;
18997     layout->at_y = layout->bounds.y + (float)view->total_height;
18998     *view->scroll_pointer = *view->scroll_pointer + view->scroll_value;
18999     nk_group_end(view->ctx);
19000 }
19001
19002
19003
19004
19005
19006 /* ===============================================================
19007  *
19008  *                              WIDGET
19009  *
19010  * ===============================================================*/
19011 NK_API struct nk_rect
19012 nk_widget_bounds(struct nk_context *ctx)
19013 {
19014     struct nk_rect bounds;
19015     NK_ASSERT(ctx);
19016     NK_ASSERT(ctx->current);
19017     if (!ctx || !ctx->current)
19018         return nk_rect(0,0,0,0);
19019     nk_layout_peek(&bounds, ctx);
19020     return bounds;
19021 }
19022 NK_API struct nk_vec2
19023 nk_widget_position(struct nk_context *ctx)
19024 {
19025     struct nk_rect bounds;
19026     NK_ASSERT(ctx);
19027     NK_ASSERT(ctx->current);
19028     if (!ctx || !ctx->current)
19029         return nk_vec2(0,0);
19030
19031     nk_layout_peek(&bounds, ctx);
19032     return nk_vec2(bounds.x, bounds.y);
19033 }
19034 NK_API struct nk_vec2
19035 nk_widget_size(struct nk_context *ctx)
19036 {
19037     struct nk_rect bounds;
19038     NK_ASSERT(ctx);
19039     NK_ASSERT(ctx->current);
19040     if (!ctx || !ctx->current)
19041         return nk_vec2(0,0);
19042
19043     nk_layout_peek(&bounds, ctx);
19044     return nk_vec2(bounds.w, bounds.h);
19045 }
19046 NK_API float
19047 nk_widget_width(struct nk_context *ctx)
19048 {
19049     struct nk_rect bounds;
19050     NK_ASSERT(ctx);
19051     NK_ASSERT(ctx->current);
19052     if (!ctx || !ctx->current)
19053         return 0;
19054
19055     nk_layout_peek(&bounds, ctx);
19056     return bounds.w;
19057 }
19058 NK_API float
19059 nk_widget_height(struct nk_context *ctx)
19060 {
19061     struct nk_rect bounds;
19062     NK_ASSERT(ctx);
19063     NK_ASSERT(ctx->current);
19064     if (!ctx || !ctx->current)
19065         return 0;
19066
19067     nk_layout_peek(&bounds, ctx);
19068     return bounds.h;
19069 }
```

```
19070 NK_API int
19071 nk_widget_is_hovered(struct nk_context *ctx)
19072 {
19073     struct nk_rect c, v;
19074     struct nk_rect bounds;
19075     NK_ASSERT(ctx);
19076     NK_ASSERT(ctx->current);
19077     if (!ctx || !ctx->current || ctx->active != ctx->current)
19078         return 0;
19079
19080     c = ctx->current->layout->clip;
19081     c.x = (float)((int)c.x);
19082     c.y = (float)((int)c.y);
19083     c.w = (float)((int)c.w);
19084     c.h = (float)((int)c.h);
19085
19086     nk_layout_peek(&bounds, ctx);
19087     nk_unify(&v, &c, bounds.x, bounds.y, bounds.x + bounds.w, bounds.y + bounds.h);
19088     if (!NK_INTERSECT(c.x, c.y, c.w, c.h, bounds.x, bounds.y, bounds.w, bounds.h))
19089         return 0;
19090     return nk_input_is_mouse_hovering_rect(&ctx->input, bounds);
19091 }
19092 NK_API int
19093 nk_widget_is_mouse_clicked(struct nk_context *ctx, enum nk_buttons btn)
19094 {
19095     struct nk_rect c, v;
19096     struct nk_rect bounds;
19097     NK_ASSERT(ctx);
19098     NK_ASSERT(ctx->current);
19099     if (!ctx || !ctx->current || ctx->active != ctx->current)
19100         return 0;
19101
19102     c = ctx->current->layout->clip;
19103     c.x = (float)((int)c.x);
19104     c.y = (float)((int)c.y);
19105     c.w = (float)((int)c.w);
19106     c.h = (float)((int)c.h);
19107
19108     nk_layout_peek(&bounds, ctx);
19109     nk_unify(&v, &c, bounds.x, bounds.y, bounds.x + bounds.w, bounds.y + bounds.h);
19110     if (!NK_INTERSECT(c.x, c.y, c.w, c.h, bounds.x, bounds.y, bounds.w, bounds.h))
19111         return 0;
19112     return nk_input_mouse_clicked(&ctx->input, btn, bounds);
19113 }
19114 NK_API int
19115 nk_widget_has_mouse_click_down(struct nk_context *ctx, enum nk_buttons btn, int down)
19116 {
19117     struct nk_rect c, v;
19118     struct nk_rect bounds;
19119     NK_ASSERT(ctx);
19120     NK_ASSERT(ctx->current);
19121     if (!ctx || !ctx->current || ctx->active != ctx->current)
19122         return 0;
19123
19124     c = ctx->current->layout->clip;
19125     c.x = (float)((int)c.x);
19126     c.y = (float)((int)c.y);
19127     c.w = (float)((int)c.w);
19128     c.h = (float)((int)c.h);
19129
19130     nk_layout_peek(&bounds, ctx);
19131     nk_unify(&v, &c, bounds.x, bounds.y, bounds.x + bounds.w, bounds.y + bounds.h);
19132     if (!NK_INTERSECT(c.x, c.y, c.w, c.h, bounds.x, bounds.y, bounds.w, bounds.h))
19133         return 0;
19134     return nk_input_has_mouse_click_down_in_rect(&ctx->input, btn, bounds, down);
19135 }
19136 NK_API enum nk_widget_layout_states
19137 nk_widget(struct nk_rect *bounds, const struct nk_context *ctx)
19138 {
19139     struct nk_rect c, v;
19140     struct nk_window *win;
19141     struct nk_panel *layout;
19142     const struct nk_input *in;
19143
19144     NK_ASSERT(ctx);
19145     NK_ASSERT(ctx->current);
19146     NK_ASSERT(ctx->current->layout);
19147     if (!ctx || !ctx->current || !ctx->current->layout)
19148         return NK_WIDGET_INVALID;
19149
19150     /* allocate space and check if the widget needs to be updated and drawn */
19151     nk_panel_alloc_space(bounds, ctx);
19152     win = ctx->current;
19153     layout = win->layout;
19154     in = &ctx->input;
19155     c = layout->clip;
19156
```

```
19157     /*  if one of these triggers you forgot to add an `if` condition around either
19158         a window, group, popup, combobox or contextual menu `begin` and `end` block.
19159         Example:
19160             if (nk_begin(...) {...} nk_end(...); or
19161             if (nk_group_begin(...) { nk_group_end(...);} */
19162     NK_ASSERT(!(layout->flags & NK_WINDOW_MINIMIZED));
19163     NK_ASSERT(!(layout->flags & NK_WINDOW_HIDDEN));
19164     NK_ASSERT(!(layout->flags & NK_WINDOW_CLOSED));
19165
19166     /* need to convert to int here to remove floating point errors */
19167     bounds->x = (float)((int)bounds->x);
19168     bounds->y = (float)((int)bounds->y);
19169     bounds->w = (float)((int)bounds->w);
19170     bounds->h = (float)((int)bounds->h);
19171
19172     c.x = (float)((int)c.x);
19173     c.y = (float)((int)c.y);
19174     c.w = (float)((int)c.w);
19175     c.h = (float)((int)c.h);
19176
19177     nk_unify(&v, &c, bounds->x, bounds->y, bounds->x + bounds->w, bounds->y + bounds->h);
19178     if (!NK_INTERSECT(c.x, c.y, c.w, c.h, bounds->x, bounds->y, bounds->w, bounds->h))
19179         return NK_WIDGET_INVALID;
19180     if (!NK_INBOX(in->mouse.pos.x, in->mouse.pos.y, v.x, v.y, v.w, v.h))
19181         return NK_WIDGET_ROM;
19182     return NK_WIDGET_VALID;
19183 }
19184 NK_API enum nk_widget_layout_states
19185 nk_widget_fitting(struct nk_rect *bounds, struct nk_context *ctx,
19186     struct nk_vec2 item_padding)
19187 {
19188     /* update the bounds to stand without padding  */
19189     struct nk_window *win;
19190     struct nk_style *style;
19191     struct nk_panel *layout;
19192     enum nk_widget_layout_states state;
19193     struct nk_vec2 panel_padding;
19194
19195     NK_ASSERT(ctx);
19196     NK_ASSERT(ctx->current);
19197     NK_ASSERT(ctx->current->layout);
19198     if (!ctx || !ctx->current || !ctx->current->layout)
19199         return NK_WIDGET_INVALID;
19200
19201     win = ctx->current;
19202     style = &ctx->style;
19203     layout = win->layout;
19204     state = nk_widget(bounds, ctx);
19205
19206     panel_padding = nk_panel_get_padding(style, layout->type);
19207     if (layout->row.index == 1) {
19208         bounds->w += panel_padding.x;
19209         bounds->x -= panel_padding.x;
19210     } else bounds->x -= item_padding.x;
19211
19212     if (layout->row.index == layout->row.columns)
19213         bounds->w += panel_padding.x;
19214     else bounds->w += item_padding.x;
19215     return state;
19216 }
19217 NK_API void
19218 nk_spacing(struct nk_context *ctx, int cols)
19219 {
19220     struct nk_window *win;
19221     struct nk_panel *layout;
19222     struct nk_rect none;
19223     int i, index, rows;
19224
19225     NK_ASSERT(ctx);
19226     NK_ASSERT(ctx->current);
19227     NK_ASSERT(ctx->current->layout);
19228     if (!ctx || !ctx->current || !ctx->current->layout)
19229         return;
19230
19231     /* spacing over row boundaries */
19232     win = ctx->current;
19233     layout = win->layout;
19234     index = (layout->row.index + cols) % layout->row.columns;
19235     rows = (layout->row.index + cols) / layout->row.columns;
19236     if (rows) {
19237         for (i = 0; i < rows; ++i)
19238             nk_panel_alloc_row(ctx, win);
19239         cols = index;
19240     }
19241     /* non table layout need to allocate space */
19242     if (layout->row.type != NK_LAYOUT_DYNAMIC_FIXED &&
19243         layout->row.type != NK_LAYOUT_STATIC_FIXED) {
```

```
19244         for (i = 0; i < cols; ++i)
19245             nk_panel_alloc_space(&none, ctx);
19246     } layout->row.index = index;
19247 }
19248
19249
19250
19251
19252
19253 /* ==============================================================
19254  *
19255  *                              TEXT
19256  *
19257  * ===============================================================*/
19258 NK_LIB void
19259 nk_widget_text(struct nk_command_buffer *o, struct nk_rect b,
19260     const char *string, int len, const struct nk_text *t,
19261     nk_flags a, const struct nk_user_font *f)
19262 {
19263     struct nk_rect label;
19264     float text_width;
19265
19266     NK_ASSERT(o);
19267     NK_ASSERT(t);
19268     if (!o || !t) return;
19269
19270     b.h = NK_MAX(b.h, 2 * t->padding.y);
19271     label.x = 0; label.w = 0;
19272     label.y = b.y + t->padding.y;
19273     label.h = NK_MIN(f->height, b.h - 2 * t->padding.y);
19274
19275     text_width = f->width(f->userdata, f->height, (const char*)string, len);
19276     text_width += (2.0f * t->padding.x);
19277
19278     /* align in x-axis */
19279     if (a & NK_TEXT_ALIGN_LEFT) {
19280         label.x = b.x + t->padding.x;
19281         label.w = NK_MAX(0, b.w - 2 * t->padding.x);
19282     } else if (a & NK_TEXT_ALIGN_CENTERED) {
19283         label.w = NK_MAX(1, 2 * t->padding.x + (float)text_width);
19284         label.x = (b.x + t->padding.x + ((b.w - 2 * t->padding.x) - label.w) / 2);
19285         label.x = NK_MAX(b.x + t->padding.x, label.x);
19286         label.w = NK_MIN(b.x + b.w, label.x + label.w);
19287         if (label.w >= label.x) label.w -= label.x;
19288     } else if (a & NK_TEXT_ALIGN_RIGHT) {
19289         label.x = NK_MAX(b.x + t->padding.x, (b.x + b.w) - (2 * t->padding.x + (float)text_width));
19290         label.w = (float)text_width + 2 * t->padding.x;
19291     } else return;
19292
19293     /* align in y-axis */
19294     if (a & NK_TEXT_ALIGN_MIDDLE) {
19295         label.y = b.y + b.h/2.0f - (float)f->height/2.0f;
19296         label.h = NK_MAX(b.h/2.0f, b.h - (b.h/2.0f + f->height/2.0f));
19297     } else if (a & NK_TEXT_ALIGN_BOTTOM) {
19298         label.y = b.y + b.h - f->height;
19299         label.h = f->height;
19300     }
19301     nk_draw_text(o, label, (const char*)string, len, f, t->background, t->text);
19302 }
19303 NK_LIB void
19304 nk_widget_text_wrap(struct nk_command_buffer *o, struct nk_rect b,
19305     const char *string, int len, const struct nk_text *t,
19306     const struct nk_user_font *f)
19307 {
19308     float width;
19309     int glyphs = 0;
19310     int fitting = 0;
19311     int done = 0;
19312     struct nk_rect line;
19313     struct nk_text text;
19314     NK_INTERN nk_rune seperator[] = {' '};
19315
19316     NK_ASSERT(o);
19317     NK_ASSERT(t);
19318     if (!o || !t) return;
19319
19320     text.padding = nk_vec2(0,0);
19321     text.background = t->background;
19322     text.text = t->text;
19323
19324     b.w = NK_MAX(b.w, 2 * t->padding.x);
19325     b.h = NK_MAX(b.h, 2 * t->padding.y);
19326     b.h = b.h - 2 * t->padding.y;
19327
19328     line.x = b.x + t->padding.x;
19329     line.y = b.y + t->padding.y;
19330     line.w = b.w - 2 * t->padding.x;
```

```
19331     line.h = 2 * t->padding.y + f->height;
19332
19333     fitting = nk_text_clamp(f, string, len, line.w, &glyphs, &width, seperator,NK_LEN(seperator));
19334     while (done < len) {
19335         if (!fitting || line.y + line.h >= (b.y + b.h)) break;
19336         nk_widget_text(o, line, &string[done], fitting, &text, NK_TEXT_LEFT, f);
19337         done += fitting;
19338         line.y += f->height + 2 * t->padding.y;
19339         fitting = nk_text_clamp(f, &string[done], len - done, line.w, &glyphs, &width,
      seperator,NK_LEN(seperator));
19340     }
19341 }
19342 NK_API void
19343 nk_text_colored(struct nk_context *ctx, const char *str, int len,
19344     nk_flags alignment, struct nk_color color)
19345 {
19346     struct nk_window *win;
19347     const struct nk_style *style;
19348
19349     struct nk_vec2 item_padding;
19350     struct nk_rect bounds;
19351     struct nk_text text;
19352
19353     NK_ASSERT(ctx);
19354     NK_ASSERT(ctx->current);
19355     NK_ASSERT(ctx->current->layout);
19356     if (!ctx || !ctx->current || !ctx->current->layout) return;
19357
19358     win = ctx->current;
19359     style = &ctx->style;
19360     nk_panel_alloc_space(&bounds, ctx);
19361     item_padding = style->text.padding;
19362
19363     text.padding.x = item_padding.x;
19364     text.padding.y = item_padding.y;
19365     text.background = style->window.background;
19366     text.text = color;
19367     nk_widget_text(&win->buffer, bounds, str, len, &text, alignment, style->font);
19368 }
19369 NK_API void
19370 nk_text_wrap_colored(struct nk_context *ctx, const char *str,
19371     int len, struct nk_color color)
19372 {
19373     struct nk_window *win;
19374     const struct nk_style *style;
19375
19376     struct nk_vec2 item_padding;
19377     struct nk_rect bounds;
19378     struct nk_text text;
19379
19380     NK_ASSERT(ctx);
19381     NK_ASSERT(ctx->current);
19382     NK_ASSERT(ctx->current->layout);
19383     if (!ctx || !ctx->current || !ctx->current->layout) return;
19384
19385     win = ctx->current;
19386     style = &ctx->style;
19387     nk_panel_alloc_space(&bounds, ctx);
19388     item_padding = style->text.padding;
19389
19390     text.padding.x = item_padding.x;
19391     text.padding.y = item_padding.y;
19392     text.background = style->window.background;
19393     text.text = color;
19394     nk_widget_text_wrap(&win->buffer, bounds, str, len, &text, style->font);
19395 }
19396 #ifdef NK_INCLUDE_STANDARD_VARARGS
19397 NK_API void
19398 nk_labelf_colored(struct nk_context *ctx, nk_flags flags,
19399     struct nk_color color, const char *fmt, ...)
19400 {
19401     va_list args;
19402     va_start(args, fmt);
19403     nk_labelfv_colored(ctx, flags, color, fmt, args);
19404     va_end(args);
19405 }
19406 NK_API void
19407 nk_labelf_colored_wrap(struct nk_context *ctx, struct nk_color color,
19408     const char *fmt, ...)
19409 {
19410     va_list args;
19411     va_start(args, fmt);
19412     nk_labelfv_colored_wrap(ctx, color, fmt, args);
19413     va_end(args);
19414 }
19415 NK_API void
19416 nk_labelf(struct nk_context *ctx, nk_flags flags, const char *fmt, ...)
```

```
19417 {
19418     va_list args;
19419     va_start(args, fmt);
19420     nk_labelfv(ctx, flags, fmt, args);
19421     va_end(args);
19422 }
19423 NK_API void
19424 nk_labelf_wrap(struct nk_context *ctx, const char *fmt,...)
19425 {
19426     va_list args;
19427     va_start(args, fmt);
19428     nk_labelfv_wrap(ctx, fmt, args);
19429     va_end(args);
19430 }
19431 NK_API void
19432 nk_labelfv_colored(struct nk_context *ctx, nk_flags flags,
19433     struct nk_color color, const char *fmt, va_list args)
19434 {
19435     char buf[256];
19436     nk_strfmt(buf, NK_LEN(buf), fmt, args);
19437     nk_label_colored(ctx, buf, flags, color);
19438 }
19439
19440 NK_API void
19441 nk_labelfv_colored_wrap(struct nk_context *ctx, struct nk_color color,
19442     const char *fmt, va_list args)
19443 {
19444     char buf[256];
19445     nk_strfmt(buf, NK_LEN(buf), fmt, args);
19446     nk_label_colored_wrap(ctx, buf, color);
19447 }
19448
19449 NK_API void
19450 nk_labelfv(struct nk_context *ctx, nk_flags flags, const char *fmt, va_list args)
19451 {
19452     char buf[256];
19453     nk_strfmt(buf, NK_LEN(buf), fmt, args);
19454     nk_label(ctx, buf, flags);
19455 }
19456
19457 NK_API void
19458 nk_labelfv_wrap(struct nk_context *ctx, const char *fmt, va_list args)
19459 {
19460     char buf[256];
19461     nk_strfmt(buf, NK_LEN(buf), fmt, args);
19462     nk_label_wrap(ctx, buf);
19463 }
19464
19465 NK_API void
19466 nk_value_bool(struct nk_context *ctx, const char *prefix, int value)
19467 {
19468     nk_labelf(ctx, NK_TEXT_LEFT, "%s: %s", prefix, ((value) ? "true": "false"));
19469 }
19470 NK_API void
19471 nk_value_int(struct nk_context *ctx, const char *prefix, int value)
19472 {
19473     nk_labelf(ctx, NK_TEXT_LEFT, "%s: %d", prefix, value);
19474 }
19475 NK_API void
19476 nk_value_uint(struct nk_context *ctx, const char *prefix, unsigned int value)
19477 {
19478     nk_labelf(ctx, NK_TEXT_LEFT, "%s: %u", prefix, value);
19479 }
19480 NK_API void
19481 nk_value_float(struct nk_context *ctx, const char *prefix, float value)
19482 {
19483     double double_value = (double)value;
19484     nk_labelf(ctx, NK_TEXT_LEFT, "%s: %.3f", prefix, double_value);
19485 }
19486 NK_API void
19487 nk_value_color_byte(struct nk_context *ctx, const char *p, struct nk_color c)
19488 {
19489     nk_labelf(ctx, NK_TEXT_LEFT, "%s: (%d, %d, %d, %d)", p, c.r, c.g, c.b, c.a);
19490 }
19491 NK_API void
19492 nk_value_color_float(struct nk_context *ctx, const char *p, struct nk_color color)
19493 {
19494     double c[4]; nk_color_dv(c, color);
19495     nk_labelf(ctx, NK_TEXT_LEFT, "%s: (%.2f, %.2f, %.2f, %.2f)",
19496         p, c[0], c[1], c[2], c[3]);
19497 }
19498 NK_API void
19499 nk_value_color_hex(struct nk_context *ctx, const char *prefix, struct nk_color color)
19500 {
19501     char hex[16];
19502     nk_color_hex_rgba(hex, color);
19503     nk_labelf(ctx, NK_TEXT_LEFT, "%s: %s", prefix, hex);
```

```
19504 }
19505 #endif
19506 NK_API void
19507 nk_text(struct nk_context *ctx, const char *str, int len, nk_flags alignment)
19508 {
19509     NK_ASSERT(ctx);
19510     if (!ctx) return;
19511     nk_text_colored(ctx, str, len, alignment, ctx->style.text.color);
19512 }
19513 NK_API void
19514 nk_text_wrap(struct nk_context *ctx, const char *str, int len)
19515 {
19516     NK_ASSERT(ctx);
19517     if (!ctx) return;
19518     nk_text_wrap_colored(ctx, str, len, ctx->style.text.color);
19519 }
19520 NK_API void
19521 nk_label(struct nk_context *ctx, const char *str, nk_flags alignment)
19522 {
19523     nk_text(ctx, str, nk_strlen(str), alignment);
19524 }
19525 NK_API void
19526 nk_label_colored(struct nk_context *ctx, const char *str, nk_flags align,
19527     struct nk_color color)
19528 {
19529     nk_text_colored(ctx, str, nk_strlen(str), align, color);
19530 }
19531 NK_API void
19532 nk_label_wrap(struct nk_context *ctx, const char *str)
19533 {
19534     nk_text_wrap(ctx, str, nk_strlen(str));
19535 }
19536 NK_API void
19537 nk_label_colored_wrap(struct nk_context *ctx, const char *str, struct nk_color color)
19538 {
19539     nk_text_wrap_colored(ctx, str, nk_strlen(str), color);
19540 }
19541
19542
19543
19544
19545
19546 /* ===============================================================
19547  *
19548  *                          IMAGE
19549  *
19550  * ===============================================================*/
19551 NK_API nk_handle
19552 nk_handle_ptr(void *ptr)
19553 {
19554     nk_handle handle = {0};
19555     handle.ptr = ptr;
19556     return handle;
19557 }
19558 NK_API nk_handle
19559 nk_handle_id(int id)
19560 {
19561     nk_handle handle;
19562     nk_zero_struct(handle);
19563     handle.id = id;
19564     return handle;
19565 }
19566 NK_API struct nk_image
19567 nk_subimage_ptr(void *ptr, unsigned short w, unsigned short h, struct nk_rect r)
19568 {
19569     struct nk_image s;
19570     nk_zero(&s, sizeof(s));
19571     s.handle.ptr = ptr;
19572     s.w = w; s.h = h;
19573     s.region[0] = (unsigned short)r.x;
19574     s.region[1] = (unsigned short)r.y;
19575     s.region[2] = (unsigned short)r.w;
19576     s.region[3] = (unsigned short)r.h;
19577     return s;
19578 }
19579 NK_API struct nk_image
19580 nk_subimage_id(int id, unsigned short w, unsigned short h, struct nk_rect r)
19581 {
19582     struct nk_image s;
19583     nk_zero(&s, sizeof(s));
19584     s.handle.id = id;
19585     s.w = w; s.h = h;
19586     s.region[0] = (unsigned short)r.x;
19587     s.region[1] = (unsigned short)r.y;
19588     s.region[2] = (unsigned short)r.w;
19589     s.region[3] = (unsigned short)r.h;
19590     return s;
```

```
19591 }
19592 NK_API struct nk_image
19593 nk_subimage_handle(nk_handle handle, unsigned short w, unsigned short h,
19594     struct nk_rect r)
19595 {
19596     struct nk_image s;
19597     nk_zero(&s, sizeof(s));
19598     s.handle = handle;
19599     s.w = w; s.h = h;
19600     s.region[0] = (unsigned short)r.x;
19601     s.region[1] = (unsigned short)r.y;
19602     s.region[2] = (unsigned short)r.w;
19603     s.region[3] = (unsigned short)r.h;
19604     return s;
19605 }
19606 NK_API struct nk_image
19607 nk_image_handle(nk_handle handle)
19608 {
19609     struct nk_image s;
19610     nk_zero(&s, sizeof(s));
19611     s.handle = handle;
19612     s.w = 0; s.h = 0;
19613     s.region[0] = 0;
19614     s.region[1] = 0;
19615     s.region[2] = 0;
19616     s.region[3] = 0;
19617     return s;
19618 }
19619 NK_API struct nk_image
19620 nk_image_ptr(void *ptr)
19621 {
19622     struct nk_image s;
19623     nk_zero(&s, sizeof(s));
19624     NK_ASSERT(ptr);
19625     s.handle.ptr = ptr;
19626     s.w = 0; s.h = 0;
19627     s.region[0] = 0;
19628     s.region[1] = 0;
19629     s.region[2] = 0;
19630     s.region[3] = 0;
19631     return s;
19632 }
19633 NK_API struct nk_image
19634 nk_image_id(int id)
19635 {
19636     struct nk_image s;
19637     nk_zero(&s, sizeof(s));
19638     s.handle.id = id;
19639     s.w = 0; s.h = 0;
19640     s.region[0] = 0;
19641     s.region[1] = 0;
19642     s.region[2] = 0;
19643     s.region[3] = 0;
19644     return s;
19645 }
19646 NK_API int
19647 nk_image_is_subimage(const struct nk_image* img)
19648 {
19649     NK_ASSERT(img);
19650     return !(img->w == 0 && img->h == 0);
19651 }
19652 NK_API void
19653 nk_image(struct nk_context *ctx, struct nk_image img)
19654 {
19655     struct nk_window *win;
19656     struct nk_rect bounds;
19657
19658     NK_ASSERT(ctx);
19659     NK_ASSERT(ctx->current);
19660     NK_ASSERT(ctx->current->layout);
19661     if (!ctx || !ctx->current || !ctx->current->layout) return;
19662
19663     win = ctx->current;
19664     if (!nk_widget(&bounds, ctx)) return;
19665     nk_draw_image(&win->buffer, bounds, &img, nk_white);
19666 }
19667 NK_API void
19668 nk_image_color(struct nk_context *ctx, struct nk_image img, struct nk_color col)
19669 {
19670     struct nk_window *win;
19671     struct nk_rect bounds;
19672
19673     NK_ASSERT(ctx);
19674     NK_ASSERT(ctx->current);
19675     NK_ASSERT(ctx->current->layout);
19676     if (!ctx || !ctx->current || !ctx->current->layout) return;
19677
```

```
19678     win = ctx->current;
19679     if (!nk_widget(&bounds, ctx)) return;
19680     nk_draw_image(&win->buffer, bounds, &img, col);
19681 }
19682
19683
19684
19685
19686
19687 /* ===============================================================
19688  *
19689  *                          BUTTON
19690  *
19691  * ===============================================================*/
19692 NK_LIB void
19693 nk_draw_symbol(struct nk_command_buffer *out, enum nk_symbol_type type,
19694     struct nk_rect content, struct nk_color background, struct nk_color foreground,
19695     float border_width, const struct nk_user_font *font)
19696 {
19697     switch (type) {
19698     case NK_SYMBOL_X:
19699     case NK_SYMBOL_UNDERSCORE:
19700     case NK_SYMBOL_PLUS:
19701     case NK_SYMBOL_MINUS: {
19702         /* single character text symbol */
19703         const char *X = (type == NK_SYMBOL_X) ? "x":
19704             (type == NK_SYMBOL_UNDERSCORE) ? "_":
19705             (type == NK_SYMBOL_PLUS) ? "+": "-";
19706         struct nk_text text;
19707         text.padding = nk_vec2(0,0);
19708         text.background = background;
19709         text.text = foreground;
19710         nk_widget_text(out, content, X, 1, &text, NK_TEXT_CENTERED, font);
19711     } break;
19712     case NK_SYMBOL_CIRCLE_SOLID:
19713     case NK_SYMBOL_CIRCLE_OUTLINE:
19714     case NK_SYMBOL_RECT_SOLID:
19715     case NK_SYMBOL_RECT_OUTLINE: {
19716         /* simple empty/filled shapes */
19717         if (type == NK_SYMBOL_RECT_SOLID || type == NK_SYMBOL_RECT_OUTLINE) {
19718             nk_fill_rect(out, content,  0, foreground);
19719             if (type == NK_SYMBOL_RECT_OUTLINE)
19720                 nk_fill_rect(out, nk_shrink_rect(content, border_width), 0, background);
19721         } else {
19722             nk_fill_circle(out, content, foreground);
19723             if (type == NK_SYMBOL_CIRCLE_OUTLINE)
19724                 nk_fill_circle(out, nk_shrink_rect(content, 1), background);
19725         }
19726     } break;
19727     case NK_SYMBOL_TRIANGLE_UP:
19728     case NK_SYMBOL_TRIANGLE_DOWN:
19729     case NK_SYMBOL_TRIANGLE_LEFT:
19730     case NK_SYMBOL_TRIANGLE_RIGHT: {
19731         enum nk_heading heading;
19732         struct nk_vec2 points[3];
19733         heading = (type == NK_SYMBOL_TRIANGLE_RIGHT) ? NK_RIGHT :
19734             (type == NK_SYMBOL_TRIANGLE_LEFT) ? NK_LEFT:
19735             (type == NK_SYMBOL_TRIANGLE_UP) ? NK_UP: NK_DOWN;
19736         nk_triangle_from_direction(points, content, 0, 0, heading);
19737         nk_fill_triangle(out, points[0].x, points[0].y, points[1].x, points[1].y,
19738             points[2].x, points[2].y, foreground);
19739     } break;
19740     default:
19741     case NK_SYMBOL_NONE:
19742     case NK_SYMBOL_MAX: break;
19743     }
19744 }
19745 NK_LIB int
19746 nk_button_behavior(nk_flags *state, struct nk_rect r,
19747     const struct nk_input *i, enum nk_button_behavior behavior)
19748 {
19749     int ret = 0;
19750     nk_widget_state_reset(state);
19751     if (!i) return 0;
19752     if (nk_input_is_mouse_hovering_rect(i, r)) {
19753         *state = NK_WIDGET_STATE_HOVERED;
19754         if (nk_input_is_mouse_down(i, NK_BUTTON_LEFT))
19755             *state = NK_WIDGET_STATE_ACTIVE;
19756         if (nk_input_has_mouse_click_in_rect(i, NK_BUTTON_LEFT, r)) {
19757             ret = (behavior != NK_BUTTON_DEFAULT) ?
19758                 nk_input_is_mouse_down(i, NK_BUTTON_LEFT):
19759 #ifdef NK_BUTTON_TRIGGER_ON_RELEASE
19760                 nk_input_is_mouse_released(i, NK_BUTTON_LEFT);
19761 #else
19762                 nk_input_is_mouse_pressed(i, NK_BUTTON_LEFT);
19763 #endif
19764         }
```

```
19765     }
19766     if (*state & NK_WIDGET_STATE_HOVER && !nk_input_is_mouse_prev_hovering_rect(i, r))
19767         *state |= NK_WIDGET_STATE_ENTERED;
19768     else if (nk_input_is_mouse_prev_hovering_rect(i, r))
19769         *state |= NK_WIDGET_STATE_LEFT;
19770     return ret;
19771 }
19772 NK_LIB const struct nk_style_item*
19773 nk_draw_button(struct nk_command_buffer *out,
19774     const struct nk_rect *bounds, nk_flags state,
19775     const struct nk_style_button *style)
19776 {
19777     const struct nk_style_item *background;
19778     if (state & NK_WIDGET_STATE_HOVER)
19779         background = &style->hover;
19780     else if (state & NK_WIDGET_STATE_ACTIVED)
19781         background = &style->active;
19782     else background = &style->normal;
19783
19784     if (background->type == NK_STYLE_ITEM_IMAGE) {
19785         nk_draw_image(out, *bounds, &background->data.image, nk_white);
19786     } else {
19787         nk_fill_rect(out, *bounds, style->rounding, background->data.color);
19788         nk_stroke_rect(out, *bounds, style->rounding, style->border, style->border_color);
19789     }
19790     return background;
19791 }
19792 NK_LIB int
19793 nk_do_button(nk_flags *state, struct nk_command_buffer *out, struct nk_rect r,
19794     const struct nk_style_button *style, const struct nk_input *in,
19795     enum nk_button_behavior behavior, struct nk_rect *content)
19796 {
19797     struct nk_rect bounds;
19798     NK_ASSERT(style);
19799     NK_ASSERT(state);
19800     NK_ASSERT(out);
19801     if (!out || !style)
19802         return nk_false;
19803
19804     /* calculate button content space */
19805     content->x = r.x + style->padding.x + style->border + style->rounding;
19806     content->y = r.y + style->padding.y + style->border + style->rounding;
19807     content->w = r.w - (2 * style->padding.x + style->border + style->rounding*2);
19808     content->h = r.h - (2 * style->padding.y + style->border + style->rounding*2);
19809
19810     /* execute button behavior */
19811     bounds.x = r.x - style->touch_padding.x;
19812     bounds.y = r.y - style->touch_padding.y;
19813     bounds.w = r.w + 2 * style->touch_padding.x;
19814     bounds.h = r.h + 2 * style->touch_padding.y;
19815     return nk_button_behavior(state, bounds, in, behavior);
19816 }
19817 NK_LIB void
19818 nk_draw_button_text(struct nk_command_buffer *out,
19819     const struct nk_rect *bounds, const struct nk_rect *content, nk_flags state,
19820     const struct nk_style_button *style, const char *txt, int len,
19821     nk_flags text_alignment, const struct nk_user_font *font)
19822 {
19823     struct nk_text text;
19824     const struct nk_style_item *background;
19825     background = nk_draw_button(out, bounds, state, style);
19826
19827     /* select correct colors/images */
19828     if (background->type == NK_STYLE_ITEM_COLOR)
19829         text.background = background->data.color;
19830     else text.background = style->text_background;
19831     if (state & NK_WIDGET_STATE_HOVER)
19832         text.text = style->text_hover;
19833     else if (state & NK_WIDGET_STATE_ACTIVED)
19834         text.text = style->text_active;
19835     else text.text = style->text_normal;
19836
19837     text.padding = nk_vec2(0,0);
19838     nk_widget_text(out, *content, txt, len, &text, text_alignment, font);
19839 }
19840 NK_LIB int
19841 nk_do_button_text(nk_flags *state,
19842     struct nk_command_buffer *out, struct nk_rect bounds,
19843     const char *string, int len, nk_flags align, enum nk_button_behavior behavior,
19844     const struct nk_style_button *style, const struct nk_input *in,
19845     const struct nk_user_font *font)
19846 {
19847     struct nk_rect content;
19848     int ret = nk_false;
19849
19850     NK_ASSERT(state);
19851     NK_ASSERT(style);
```

```
19852     NK_ASSERT(out);
19853     NK_ASSERT(string);
19854     NK_ASSERT(font);
19855     if (!out || !style || !font || !string)
19856         return nk_false;
19857
19858     ret = nk_do_button(state, out, bounds, style, in, behavior, &content);
19859     if (style->draw_begin) style->draw_begin(out, style->userdata);
19860     nk_draw_button_text(out, &bounds, &content, *state, style, string, len, align, font);
19861     if (style->draw_end) style->draw_end(out, style->userdata);
19862     return ret;
19863 }
19864 NK_LIB void
19865 nk_draw_button_symbol(struct nk_command_buffer *out,
19866     const struct nk_rect *bounds, const struct nk_rect *content,
19867     nk_flags state, const struct nk_style_button *style,
19868     enum nk_symbol_type type, const struct nk_user_font *font)
19869 {
19870     struct nk_color sym, bg;
19871     const struct nk_style_item *background;
19872
19873     /* select correct colors/images */
19874     background = nk_draw_button(out, bounds, state, style);
19875     if (background->type == NK_STYLE_ITEM_COLOR)
19876         bg = background->data.color;
19877     else bg = style->text_background;
19878
19879     if (state & NK_WIDGET_STATE_HOVER)
19880         sym = style->text_hover;
19881     else if (state & NK_WIDGET_STATE_ACTIVED)
19882         sym = style->text_active;
19883     else sym = style->text_normal;
19884     nk_draw_symbol(out, type, *content, bg, sym, 1, font);
19885 }
19886 NK_LIB int
19887 nk_do_button_symbol(nk_flags *state,
19888     struct nk_command_buffer *out, struct nk_rect bounds,
19889     enum nk_symbol_type symbol, enum nk_button_behavior behavior,
19890     const struct nk_style_button *style, const struct nk_input *in,
19891     const struct nk_user_font *font)
19892 {
19893     int ret;
19894     struct nk_rect content;
19895
19896     NK_ASSERT(state);
19897     NK_ASSERT(style);
19898     NK_ASSERT(font);
19899     NK_ASSERT(out);
19900     if (!out || !style || !font || !state)
19901         return nk_false;
19902
19903     ret = nk_do_button(state, out, bounds, style, in, behavior, &content);
19904     if (style->draw_begin) style->draw_begin(out, style->userdata);
19905     nk_draw_button_symbol(out, &bounds, &content, *state, style, symbol, font);
19906     if (style->draw_end) style->draw_end(out, style->userdata);
19907     return ret;
19908 }
19909 NK_LIB void
19910 nk_draw_button_image(struct nk_command_buffer *out,
19911     const struct nk_rect *bounds, const struct nk_rect *content,
19912     nk_flags state, const struct nk_style_button *style, const struct nk_image *img)
19913 {
19914     nk_draw_button(out, bounds, state, style);
19915     nk_draw_image(out, *content, img, nk_white);
19916 }
19917 NK_LIB int
19918 nk_do_button_image(nk_flags *state,
19919     struct nk_command_buffer *out, struct nk_rect bounds,
19920     struct nk_image img, enum nk_button_behavior b,
19921     const struct nk_style_button *style, const struct nk_input *in)
19922 {
19923     int ret;
19924     struct nk_rect content;
19925
19926     NK_ASSERT(state);
19927     NK_ASSERT(style);
19928     NK_ASSERT(out);
19929     if (!out || !style || !state)
19930         return nk_false;
19931
19932     ret = nk_do_button(state, out, bounds, style, in, b, &content);
19933     content.x += style->image_padding.x;
19934     content.y += style->image_padding.y;
19935     content.w -= 2 * style->image_padding.x;
19936     content.h -= 2 * style->image_padding.y;
19937
19938     if (style->draw_begin) style->draw_begin(out, style->userdata);
```

```
19939     nk_draw_button_image(out, &bounds, &content, *state, style, &img);
19940     if (style->draw_end) style->draw_end(out, style->userdata);
19941     return ret;
19942 }
19943 NK_LIB void
19944 nk_draw_button_text_symbol(struct nk_command_buffer *out,
19945     const struct nk_rect *bounds, const struct nk_rect *label,
19946     const struct nk_rect *symbol, nk_flags state, const struct nk_style_button *style,
19947     const char *str, int len, enum nk_symbol_type type,
19948     const struct nk_user_font *font)
19949 {
19950     struct nk_color sym;
19951     struct nk_text text;
19952     const struct nk_style_item *background;
19953
19954     /* select correct background colors/images */
19955     background = nk_draw_button(out, bounds, state, style);
19956     if (background->type == NK_STYLE_ITEM_COLOR)
19957         text.background = background->data.color;
19958     else text.background = style->text_background;
19959
19960     /* select correct text colors */
19961     if (state & NK_WIDGET_STATE_HOVER) {
19962         sym = style->text_hover;
19963         text.text = style->text_hover;
19964     } else if (state & NK_WIDGET_STATE_ACTIVED) {
19965         sym = style->text_active;
19966         text.text = style->text_active;
19967     } else {
19968         sym = style->text_normal;
19969         text.text = style->text_normal;
19970     }
19971
19972     text.padding = nk_vec2(0,0);
19973     nk_draw_symbol(out, type, *symbol, style->text_background, sym, 0, font);
19974     nk_widget_text(out, *label, str, len, &text, NK_TEXT_CENTERED, font);
19975 }
19976 NK_LIB int
19977 nk_do_button_text_symbol(nk_flags *state,
19978     struct nk_command_buffer *out, struct nk_rect bounds,
19979     enum nk_symbol_type symbol, const char *str, int len, nk_flags align,
19980     enum nk_button_behavior behavior, const struct nk_style_button *style,
19981     const struct nk_user_font *font, const struct nk_input *in)
19982 {
19983     int ret;
19984     struct nk_rect tri = {0,0,0,0};
19985     struct nk_rect content;
19986
19987     NK_ASSERT(style);
19988     NK_ASSERT(out);
19989     NK_ASSERT(font);
19990     if (!out || !style || !font)
19991         return nk_false;
19992
19993     ret = nk_do_button(state, out, bounds, style, in, behavior, &content);
19994     tri.y = content.y + (content.h/2) - font->height/2;
19995     tri.w = font->height; tri.h = font->height;
19996     if (align & NK_TEXT_ALIGN_LEFT) {
19997         tri.x = (content.x + content.w) - (2 * style->padding.x + tri.w);
19998         tri.x = NK_MAX(tri.x, 0);
19999     } else tri.x = content.x + 2 * style->padding.x;
20000
20001     /* draw button */
20002     if (style->draw_begin) style->draw_begin(out, style->userdata);
20003     nk_draw_button_text_symbol(out, &bounds, &content, &tri,
20004         *state, style, str, len, symbol, font);
20005     if (style->draw_end) style->draw_end(out, style->userdata);
20006     return ret;
20007 }
20008 NK_LIB void
20009 nk_draw_button_text_image(struct nk_command_buffer *out,
20010     const struct nk_rect *bounds, const struct nk_rect *label,
20011     const struct nk_rect *image, nk_flags state, const struct nk_style_button *style,
20012     const char *str, int len, const struct nk_user_font *font,
20013     const struct nk_image *img)
20014 {
20015     struct nk_text text;
20016     const struct nk_style_item *background;
20017     background = nk_draw_button(out, bounds, state, style);
20018
20019     /* select correct colors */
20020     if (background->type == NK_STYLE_ITEM_COLOR)
20021         text.background = background->data.color;
20022     else text.background = style->text_background;
20023     if (state & NK_WIDGET_STATE_HOVER)
20024         text.text = style->text_hover;
20025     else if (state & NK_WIDGET_STATE_ACTIVED)
```

```
20026         text.text = style->text_active;
20027     else text.text = style->text_normal;
20028
20029     text.padding = nk_vec2(0,0);
20030     nk_widget_text(out, *label, str, len, &text, NK_TEXT_CENTERED, font);
20031     nk_draw_image(out, *image, img, nk_white);
20032 }
20033 NK_LIB int
20034 nk_do_button_text_image(nk_flags *state,
20035     struct nk_command_buffer *out, struct nk_rect bounds,
20036     struct nk_image img, const char* str, int len, nk_flags align,
20037     enum nk_button_behavior behavior, const struct nk_style_button *style,
20038     const struct nk_user_font *font, const struct nk_input *in)
20039 {
20040     int ret;
20041     struct nk_rect icon;
20042     struct nk_rect content;
20043
20044     NK_ASSERT(style);
20045     NK_ASSERT(state);
20046     NK_ASSERT(font);
20047     NK_ASSERT(out);
20048     if (!out || !font || !style || !str)
20049         return nk_false;
20050
20051     ret = nk_do_button(state, out, bounds, style, in, behavior, &content);
20052     icon.y = bounds.y + style->padding.y;
20053     icon.w = icon.h = bounds.h - 2 * style->padding.y;
20054     if (align & NK_TEXT_ALIGN_LEFT) {
20055         icon.x = (bounds.x + bounds.w) - (2 * style->padding.x + icon.w);
20056         icon.x = NK_MAX(icon.x, 0);
20057     } else icon.x = bounds.x + 2 * style->padding.x;
20058
20059     icon.x += style->image_padding.x;
20060     icon.y += style->image_padding.y;
20061     icon.w -= 2 * style->image_padding.x;
20062     icon.h -= 2 * style->image_padding.y;
20063
20064     if (style->draw_begin) style->draw_begin(out, style->userdata);
20065     nk_draw_button_text_image(out, &bounds, &content, &icon, *state, style, str, len, font, &img);
20066     if (style->draw_end) style->draw_end(out, style->userdata);
20067     return ret;
20068 }
20069 NK_API void
20070 nk_button_set_behavior(struct nk_context *ctx, enum nk_button_behavior behavior)
20071 {
20072     NK_ASSERT(ctx);
20073     if (!ctx) return;
20074     ctx->button_behavior = behavior;
20075 }
20076 NK_API int
20077 nk_button_push_behavior(struct nk_context *ctx, enum nk_button_behavior behavior)
20078 {
20079     struct nk_config_stack_button_behavior *button_stack;
20080     struct nk_config_stack_button_behavior_element *element;
20081
20082     NK_ASSERT(ctx);
20083     if (!ctx) return 0;
20084
20085     button_stack = &ctx->stacks.button_behaviors;
20086     NK_ASSERT(button_stack->head < (int)NK_LEN(button_stack->elements));
20087     if (button_stack->head >= (int)NK_LEN(button_stack->elements))
20088         return 0;
20089
20090     element = &button_stack->elements[button_stack->head++];
20091     element->address = &ctx->button_behavior;
20092     element->old_value = ctx->button_behavior;
20093     ctx->button_behavior = behavior;
20094     return 1;
20095 }
20096 NK_API int
20097 nk_button_pop_behavior(struct nk_context *ctx)
20098 {
20099     struct nk_config_stack_button_behavior *button_stack;
20100     struct nk_config_stack_button_behavior_element *element;
20101
20102     NK_ASSERT(ctx);
20103     if (!ctx) return 0;
20104
20105     button_stack = &ctx->stacks.button_behaviors;
20106     NK_ASSERT(button_stack->head > 0);
20107     if (button_stack->head < 1)
20108         return 0;
20109
20110     element = &button_stack->elements[--button_stack->head];
20111     *element->address = element->old_value;
20112     return 1;
```

```
20113 }
20114 NK_API int
20115 nk_button_text_styled(struct nk_context *ctx,
20116     const struct nk_style_button *style, const char *title, int len)
20117 {
20118     struct nk_window *win;
20119     struct nk_panel *layout;
20120     const struct nk_input *in;
20121
20122     struct nk_rect bounds;
20123     enum nk_widget_layout_states state;
20124
20125     NK_ASSERT(ctx);
20126     NK_ASSERT(style);
20127     NK_ASSERT(ctx->current);
20128     NK_ASSERT(ctx->current->layout);
20129     if (!style || !ctx || !ctx->current || !ctx->current->layout) return 0;
20130
20131     win = ctx->current;
20132     layout = win->layout;
20133     state = nk_widget(&bounds, ctx);
20134
20135     if (!state) return 0;
20136     in = (state == NK_WIDGET_ROM || layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
20137     return nk_do_button_text(&ctx->last_widget_state, &win->buffer, bounds,
20138                     title, len, style->text_alignment, ctx->button_behavior,
20139                     style, in, ctx->style.font);
20140 }
20141 NK_API int
20142 nk_button_text(struct nk_context *ctx, const char *title, int len)
20143 {
20144     NK_ASSERT(ctx);
20145     if (!ctx) return 0;
20146     return nk_button_text_styled(ctx, &ctx->style.button, title, len);
20147 }
20148 NK_API int nk_button_label_styled(struct nk_context *ctx,
20149     const struct nk_style_button *style, const char *title)
20150 {
20151     return nk_button_text_styled(ctx, style, title, nk_strlen(title));
20152 }
20153 NK_API int nk_button_label(struct nk_context *ctx, const char *title)
20154 {
20155     return nk_button_text(ctx, title, nk_strlen(title));
20156 }
20157 NK_API int
20158 nk_button_color(struct nk_context *ctx, struct nk_color color)
20159 {
20160     struct nk_window *win;
20161     struct nk_panel *layout;
20162     const struct nk_input *in;
20163     struct nk_style_button button;
20164
20165     int ret = 0;
20166     struct nk_rect bounds;
20167     struct nk_rect content;
20168     enum nk_widget_layout_states state;
20169
20170     NK_ASSERT(ctx);
20171     NK_ASSERT(ctx->current);
20172     NK_ASSERT(ctx->current->layout);
20173     if (!ctx || !ctx->current || !ctx->current->layout)
20174         return 0;
20175
20176     win = ctx->current;
20177     layout = win->layout;
20178
20179     state = nk_widget(&bounds, ctx);
20180     if (!state) return 0;
20181     in = (state == NK_WIDGET_ROM || layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
20182
20183     button = ctx->style.button;
20184     button.normal = nk_style_item_color(color);
20185     button.hover = nk_style_item_color(color);
20186     button.active = nk_style_item_color(color);
20187     ret = nk_do_button(&ctx->last_widget_state, &win->buffer, bounds,
20188                 &button, in, ctx->button_behavior, &content);
20189     nk_draw_button(&win->buffer, &bounds, ctx->last_widget_state, &button);
20190     return ret;
20191 }
20192 NK_API int
20193 nk_button_symbol_styled(struct nk_context *ctx,
20194     const struct nk_style_button *style, enum nk_symbol_type symbol)
20195 {
20196     struct nk_window *win;
20197     struct nk_panel *layout;
20198     const struct nk_input *in;
20199
```

```
20200     struct nk_rect bounds;
20201     enum nk_widget_layout_states state;
20202
20203     NK_ASSERT(ctx);
20204     NK_ASSERT(ctx->current);
20205     NK_ASSERT(ctx->current->layout);
20206     if (!ctx || !ctx->current || !ctx->current->layout)
20207         return 0;
20208
20209     win = ctx->current;
20210     layout = win->layout;
20211     state = nk_widget(&bounds, ctx);
20212     if (!state) return 0;
20213     in = (state == NK_WIDGET_ROM || layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
20214     return nk_do_button_symbol(&ctx->last_widget_state, &win->buffer, bounds,
20215             symbol, ctx->button_behavior, style, in, ctx->style.font);
20216 }
20217 NK_API int
20218 nk_button_symbol(struct nk_context *ctx, enum nk_symbol_type symbol)
20219 {
20220     NK_ASSERT(ctx);
20221     if (!ctx) return 0;
20222     return nk_button_symbol_styled(ctx, &ctx->style.button, symbol);
20223 }
20224 NK_API int
20225 nk_button_image_styled(struct nk_context *ctx, const struct nk_style_button *style,
20226     struct nk_image img)
20227 {
20228     struct nk_window *win;
20229     struct nk_panel *layout;
20230     const struct nk_input *in;
20231
20232     struct nk_rect bounds;
20233     enum nk_widget_layout_states state;
20234
20235     NK_ASSERT(ctx);
20236     NK_ASSERT(ctx->current);
20237     NK_ASSERT(ctx->current->layout);
20238     if (!ctx || !ctx->current || !ctx->current->layout)
20239         return 0;
20240
20241     win = ctx->current;
20242     layout = win->layout;
20243
20244     state = nk_widget(&bounds, ctx);
20245     if (!state) return 0;
20246     in = (state == NK_WIDGET_ROM || layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
20247     return nk_do_button_image(&ctx->last_widget_state, &win->buffer, bounds,
20248                 img, ctx->button_behavior, style, in);
20249 }
20250 NK_API int
20251 nk_button_image(struct nk_context *ctx, struct nk_image img)
20252 {
20253     NK_ASSERT(ctx);
20254     if (!ctx) return 0;
20255     return nk_button_image_styled(ctx, &ctx->style.button, img);
20256 }
20257 NK_API int
20258 nk_button_symbol_text_styled(struct nk_context *ctx,
20259     const struct nk_style_button *style, enum nk_symbol_type symbol,
20260     const char *text, int len, nk_flags align)
20261 {
20262     struct nk_window *win;
20263     struct nk_panel *layout;
20264     const struct nk_input *in;
20265
20266     struct nk_rect bounds;
20267     enum nk_widget_layout_states state;
20268
20269     NK_ASSERT(ctx);
20270     NK_ASSERT(ctx->current);
20271     NK_ASSERT(ctx->current->layout);
20272     if (!ctx || !ctx->current || !ctx->current->layout)
20273         return 0;
20274
20275     win = ctx->current;
20276     layout = win->layout;
20277
20278     state = nk_widget(&bounds, ctx);
20279     if (!state) return 0;
20280     in = (state == NK_WIDGET_ROM || layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
20281     return nk_do_button_text_symbol(&ctx->last_widget_state, &win->buffer, bounds,
20282                 symbol, text, len, align, ctx->button_behavior,
20283                 style, ctx->style.font, in);
20284 }
20285 NK_API int
20286 nk_button_symbol_text(struct nk_context *ctx, enum nk_symbol_type symbol,
```

```
20287     const char* text, int len, nk_flags align)
20288 {
20289     NK_ASSERT(ctx);
20290     if (!ctx) return 0;
20291     return nk_button_symbol_text_styled(ctx, &ctx->style.button, symbol, text, len, align);
20292 }
20293 NK_API int nk_button_symbol_label(struct nk_context *ctx, enum nk_symbol_type symbol,
20294     const char *label, nk_flags align)
20295 {
20296     return nk_button_symbol_text(ctx, symbol, label, nk_strlen(label), align);
20297 }
20298 NK_API int nk_button_symbol_label_styled(struct nk_context *ctx,
20299     const struct nk_style_button *style, enum nk_symbol_type symbol,
20300     const char *title, nk_flags align)
20301 {
20302     return nk_button_symbol_text_styled(ctx, style, symbol, title, nk_strlen(title), align);
20303 }
20304 NK_API int
20305 nk_button_image_text_styled(struct nk_context *ctx,
20306     const struct nk_style_button *style, struct nk_image img, const char *text,
20307     int len, nk_flags align)
20308 {
20309     struct nk_window *win;
20310     struct nk_panel *layout;
20311     const struct nk_input *in;
20312
20313     struct nk_rect bounds;
20314     enum nk_widget_layout_states state;
20315
20316     NK_ASSERT(ctx);
20317     NK_ASSERT(ctx->current);
20318     NK_ASSERT(ctx->current->layout);
20319     if (!ctx || !ctx->current || !ctx->current->layout)
20320         return 0;
20321
20322     win = ctx->current;
20323     layout = win->layout;
20324
20325     state = nk_widget(&bounds, ctx);
20326     if (!state) return 0;
20327     in = (state == NK_WIDGET_ROM || layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
20328     return nk_do_button_text_image(&ctx->last_widget_state, &win->buffer,
20329             bounds, img, text, len, align, ctx->button_behavior,
20330             style, ctx->style.font, in);
20331 }
20332 NK_API int
20333 nk_button_image_text(struct nk_context *ctx, struct nk_image img,
20334     const char *text, int len, nk_flags align)
20335 {
20336     return nk_button_image_text_styled(ctx, &ctx->style.button,img, text, len, align);
20337 }
20338 NK_API int nk_button_image_label(struct nk_context *ctx, struct nk_image img,
20339     const char *label, nk_flags align)
20340 {
20341     return nk_button_image_text(ctx, img, label, nk_strlen(label), align);
20342 }
20343 NK_API int nk_button_image_label_styled(struct nk_context *ctx,
20344     const struct nk_style_button *style, struct nk_image img,
20345     const char *label, nk_flags text_alignment)
20346 {
20347     return nk_button_image_text_styled(ctx, style, img, label, nk_strlen(label), text_alignment);
20348 }
20349
20350
20351
20352
20353
20354 /* ==============================================================
20355  *
20356  *                              TOGGLE
20357  *
20358  * ==============================================================*/
20359 NK_LIB int
20360 nk_toggle_behavior(const struct nk_input *in, struct nk_rect select,
20361     nk_flags *state, int active)
20362 {
20363     nk_widget_state_reset(state);
20364     if (nk_button_behavior(state, select, in, NK_BUTTON_DEFAULT)) {
20365         *state = NK_WIDGET_STATE_ACTIVE;
20366         active = !active;
20367     }
20368     if (*state & NK_WIDGET_STATE_HOVER && !nk_input_is_mouse_prev_hovering_rect(in, select))
20369         *state |= NK_WIDGET_STATE_ENTERED;
20370     else if (nk_input_is_mouse_prev_hovering_rect(in, select))
20371         *state |= NK_WIDGET_STATE_LEFT;
20372     return active;
20373 }
```

```
20374 NK_LIB void
20375 nk_draw_checkbox(struct nk_command_buffer *out,
20376     nk_flags state, const struct nk_style_toggle *style, int active,
20377     const struct nk_rect *label, const struct nk_rect *selector,
20378     const struct nk_rect *cursors, const char *string, int len,
20379     const struct nk_user_font *font)
20380 {
20381     const struct nk_style_item *background;
20382     const struct nk_style_item *cursor;
20383     struct nk_text text;
20384
20385     /* select correct colors/images */
20386     if (state & NK_WIDGET_STATE_HOVER) {
20387         background = &style->hover;
20388         cursor = &style->cursor_hover;
20389         text.text = style->text_hover;
20390     } else if (state & NK_WIDGET_STATE_ACTIVED) {
20391         background = &style->hover;
20392         cursor = &style->cursor_hover;
20393         text.text = style->text_active;
20394     } else {
20395         background = &style->normal;
20396         cursor = &style->cursor_normal;
20397         text.text = style->text_normal;
20398     }
20399
20400     /* draw background and cursor */
20401     if (background->type == NK_STYLE_ITEM_COLOR) {
20402         nk_fill_rect(out, *selector, 0, style->border_color);
20403         nk_fill_rect(out, nk_shrink_rect(*selector, style->border), 0, background->data.color);
20404     } else nk_draw_image(out, *selector, &background->data.image, nk_white);
20405     if (active) {
20406         if (cursor->type == NK_STYLE_ITEM_IMAGE)
20407             nk_draw_image(out, *cursors, &cursor->data.image, nk_white);
20408         else nk_fill_rect(out, *cursors, 0, cursor->data.color);
20409     }
20410
20411     text.padding.x = 0;
20412     text.padding.y = 0;
20413     text.background = style->text_background;
20414     nk_widget_text(out, *label, string, len, &text, NK_TEXT_LEFT, font);
20415 }
20416 NK_LIB void
20417 nk_draw_option(struct nk_command_buffer *out,
20418     nk_flags state, const struct nk_style_toggle *style, int active,
20419     const struct nk_rect *label, const struct nk_rect *selector,
20420     const struct nk_rect *cursors, const char *string, int len,
20421     const struct nk_user_font *font)
20422 {
20423     const struct nk_style_item *background;
20424     const struct nk_style_item *cursor;
20425     struct nk_text text;
20426
20427     /* select correct colors/images */
20428     if (state & NK_WIDGET_STATE_HOVER) {
20429         background = &style->hover;
20430         cursor = &style->cursor_hover;
20431         text.text = style->text_hover;
20432     } else if (state & NK_WIDGET_STATE_ACTIVED) {
20433         background = &style->hover;
20434         cursor = &style->cursor_hover;
20435         text.text = style->text_active;
20436     } else {
20437         background = &style->normal;
20438         cursor = &style->cursor_normal;
20439         text.text = style->text_normal;
20440     }
20441
20442     /* draw background and cursor */
20443     if (background->type == NK_STYLE_ITEM_COLOR) {
20444         nk_fill_circle(out, *selector, style->border_color);
20445         nk_fill_circle(out, nk_shrink_rect(*selector, style->border), background->data.color);
20446     } else nk_draw_image(out, *selector, &background->data.image, nk_white);
20447     if (active) {
20448         if (cursor->type == NK_STYLE_ITEM_IMAGE)
20449             nk_draw_image(out, *cursors, &cursor->data.image, nk_white);
20450         else nk_fill_circle(out, *cursors, cursor->data.color);
20451     }
20452
20453     text.padding.x = 0;
20454     text.padding.y = 0;
20455     text.background = style->text_background;
20456     nk_widget_text(out, *label, string, len, &text, NK_TEXT_LEFT, font);
20457 }
20458 NK_LIB int
20459 nk_do_toggle(nk_flags *state,
20460     struct nk_command_buffer *out, struct nk_rect r,
```

```
20461     int *active, const char *str, int len, enum nk_toggle_type type,
20462     const struct nk_style_toggle *style, const struct nk_input *in,
20463     const struct nk_user_font *font)
20464 {
20465     int was_active;
20466     struct nk_rect bounds;
20467     struct nk_rect select;
20468     struct nk_rect cursor;
20469     struct nk_rect label;
20470
20471     NK_ASSERT(style);
20472     NK_ASSERT(out);
20473     NK_ASSERT(font);
20474     if (!out || !style || !font || !active)
20475         return 0;
20476
20477     r.w = NK_MAX(r.w, font->height + 2 * style->padding.x);
20478     r.h = NK_MAX(r.h, font->height + 2 * style->padding.y);
20479
20480     /* add additional touch padding for touch screen devices */
20481     bounds.x = r.x - style->touch_padding.x;
20482     bounds.y = r.y - style->touch_padding.y;
20483     bounds.w = r.w + 2 * style->touch_padding.x;
20484     bounds.h = r.h + 2 * style->touch_padding.y;
20485
20486     /* calculate the selector space */
20487     select.w = font->height;
20488     select.h = select.w;
20489     select.y = r.y + r.h/2.0f - select.h/2.0f;
20490     select.x = r.x;
20491
20492     /* calculate the bounds of the cursor inside the selector */
20493     cursor.x = select.x + style->padding.x + style->border;
20494     cursor.y = select.y + style->padding.y + style->border;
20495     cursor.w = select.w - (2 * style->padding.x + 2 * style->border);
20496     cursor.h = select.h - (2 * style->padding.y + 2 * style->border);
20497
20498     /* label behind the selector */
20499     label.x = select.x + select.w + style->spacing;
20500     label.y = select.y;
20501     label.w = NK_MAX(r.x + r.w, label.x) - label.x;
20502     label.h = select.w;
20503
20504     /* update selector */
20505     was_active = *active;
20506     *active = nk_toggle_behavior(in, bounds, state, *active);
20507
20508     /* draw selector */
20509     if (style->draw_begin)
20510         style->draw_begin(out, style->userdata);
20511     if (type == NK_TOGGLE_CHECK) {
20512         nk_draw_checkbox(out, *state, style, *active, &label, &select, &cursor, str, len, font);
20513     } else {
20514         nk_draw_option(out, *state, style, *active, &label, &select, &cursor, str, len, font);
20515     }
20516     if (style->draw_end)
20517         style->draw_end(out, style->userdata);
20518     return (was_active != *active);
20519 }
20520 /*---------------------------------------------------------------
20521  *
20522  *                          CHECKBOX
20523  *
20524  * ---------------------------------------------------------------*/
20525 NK_API int
20526 nk_check_text(struct nk_context *ctx, const char *text, int len, int active)
20527 {
20528     struct nk_window *win;
20529     struct nk_panel *layout;
20530     const struct nk_input *in;
20531     const struct nk_style *style;
20532
20533     struct nk_rect bounds;
20534     enum nk_widget_layout_states state;
20535
20536     NK_ASSERT(ctx);
20537     NK_ASSERT(ctx->current);
20538     NK_ASSERT(ctx->current->layout);
20539     if (!ctx || !ctx->current || !ctx->current->layout)
20540         return active;
20541
20542     win = ctx->current;
20543     style = &ctx->style;
20544     layout = win->layout;
20545
20546     state = nk_widget(&bounds, ctx);
20547     if (!state) return active;
```

```
20548     in = (state == NK_WIDGET_ROM || layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
20549     nk_do_toggle(&ctx->last_widget_state, &win->buffer, bounds, &active,
20550         text, len, NK_TOGGLE_CHECK, &style->checkbox, in, style->font);
20551     return active;
20552 }
20553 NK_API unsigned int
20554 nk_check_flags_text(struct nk_context *ctx, const char *text, int len,
20555     unsigned int flags, unsigned int value)
20556 {
20557     int old_active;
20558     NK_ASSERT(ctx);
20559     NK_ASSERT(text);
20560     if (!ctx || !text) return flags;
20561     old_active = (int)((flags & value) & value);
20562     if (nk_check_text(ctx, text, len, old_active))
20563         flags |= value;
20564     else flags &= ~value;
20565     return flags;
20566 }
20567 NK_API int
20568 nk_checkbox_text(struct nk_context *ctx, const char *text, int len, int *active)
20569 {
20570     int old_val;
20571     NK_ASSERT(ctx);
20572     NK_ASSERT(text);
20573     NK_ASSERT(active);
20574     if (!ctx || !text || !active) return 0;
20575     old_val = *active;
20576     *active = nk_check_text(ctx, text, len, *active);
20577     return old_val != *active;
20578 }
20579 NK_API int
20580 nk_checkbox_flags_text(struct nk_context *ctx, const char *text, int len,
20581     unsigned int *flags, unsigned int value)
20582 {
20583     int active;
20584     NK_ASSERT(ctx);
20585     NK_ASSERT(text);
20586     NK_ASSERT(flags);
20587     if (!ctx || !text || !flags) return 0;
20588
20589     active = (int)((*flags & value) & value);
20590     if (nk_checkbox_text(ctx, text, len, &active)) {
20591         if (active) *flags |= value;
20592         else *flags &= ~value;
20593         return 1;
20594     }
20595     return 0;
20596 }
20597 NK_API int nk_check_label(struct nk_context *ctx, const char *label, int active)
20598 {
20599     return nk_check_text(ctx, label, nk_strlen(label), active);
20600 }
20601 NK_API unsigned int nk_check_flags_label(struct nk_context *ctx, const char *label,
20602     unsigned int flags, unsigned int value)
20603 {
20604     return nk_check_flags_text(ctx, label, nk_strlen(label), flags, value);
20605 }
20606 NK_API int nk_checkbox_label(struct nk_context *ctx, const char *label, int *active)
20607 {
20608     return nk_checkbox_text(ctx, label, nk_strlen(label), active);
20609 }
20610 NK_API int nk_checkbox_flags_label(struct nk_context *ctx, const char *label,
20611     unsigned int *flags, unsigned int value)
20612 {
20613     return nk_checkbox_flags_text(ctx, label, nk_strlen(label), flags, value);
20614 }
20615 /*---------------------------------------------------------------
20616  *
20617  *                          OPTION
20618  *
20619  * ---------------------------------------------------------------*/
20620 NK_API int
20621 nk_option_text(struct nk_context *ctx, const char *text, int len, int is_active)
20622 {
20623     struct nk_window *win;
20624     struct nk_panel *layout;
20625     const struct nk_input *in;
20626     const struct nk_style *style;
20627
20628     struct nk_rect bounds;
20629     enum nk_widget_layout_states state;
20630
20631     NK_ASSERT(ctx);
20632     NK_ASSERT(ctx->current);
20633     NK_ASSERT(ctx->current->layout);
20634     if (!ctx || !ctx->current || !ctx->current->layout)
```

```
20635         return is_active;
20636
20637     win = ctx->current;
20638     style = &ctx->style;
20639     layout = win->layout;
20640
20641     state = nk_widget(&bounds, ctx);
20642     if (!state) return (int)state;
20643     in = (state == NK_WIDGET_ROM || layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
20644     nk_do_toggle(&ctx->last_widget_state, &win->buffer, bounds, &is_active,
20645         text, len, NK_TOGGLE_OPTION, &style->option, in, style->font);
20646     return is_active;
20647 }
20648 NK_API int
20649 nk_radio_text(struct nk_context *ctx, const char *text, int len, int *active)
20650 {
20651     int old_value;
20652     NK_ASSERT(ctx);
20653     NK_ASSERT(text);
20654     NK_ASSERT(active);
20655     if (!ctx || !text || !active) return 0;
20656     old_value = *active;
20657     *active = nk_option_text(ctx, text, len, old_value);
20658     return old_value != *active;
20659 }
20660 NK_API int
20661 nk_option_label(struct nk_context *ctx, const char *label, int active)
20662 {
20663     return nk_option_text(ctx, label, nk_strlen(label), active);
20664 }
20665 NK_API int
20666 nk_radio_label(struct nk_context *ctx, const char *label, int *active)
20667 {
20668     return nk_radio_text(ctx, label, nk_strlen(label), active);
20669 }
20670
20671
20672
20673
20674
20675 /* ===============================================================
20676  *
20677  *                              SELECTABLE
20678  *
20679  * ===============================================================*/
20680 NK_LIB void
20681 nk_draw_selectable(struct nk_command_buffer *out,
20682     nk_flags state, const struct nk_style_selectable *style, int active,
20683     const struct nk_rect *bounds,
20684     const struct nk_rect *icon, const struct nk_image *img, enum nk_symbol_type sym,
20685     const char *string, int len, nk_flags align, const struct nk_user_font *font)
20686 {
20687     const struct nk_style_item *background;
20688     struct nk_text text;
20689     text.padding = style->padding;
20690
20691     /* select correct colors/images */
20692     if (!active) {
20693         if (state & NK_WIDGET_STATE_ACTIVED) {
20694             background = &style->pressed;
20695             text.text = style->text_pressed;
20696         } else if (state & NK_WIDGET_STATE_HOVER) {
20697             background = &style->hover;
20698             text.text = style->text_hover;
20699         } else {
20700             background = &style->normal;
20701             text.text = style->text_normal;
20702         }
20703     } else {
20704         if (state & NK_WIDGET_STATE_ACTIVED) {
20705             background = &style->pressed_active;
20706             text.text = style->text_pressed_active;
20707         } else if (state & NK_WIDGET_STATE_HOVER) {
20708             background = &style->hover_active;
20709             text.text = style->text_hover_active;
20710         } else {
20711             background = &style->normal_active;
20712             text.text = style->text_normal_active;
20713         }
20714     }
20715     /* draw selectable background and text */
20716     if (background->type == NK_STYLE_ITEM_IMAGE) {
20717         nk_draw_image(out, *bounds, &background->data.image, nk_white);
20718         text.background = nk_rgba(0,0,0,0);
20719     } else {
20720         nk_fill_rect(out, *bounds, style->rounding, background->data.color);
20721         text.background = background->data.color;
```

```
20722     }
20723     if (icon) {
20724         if (img) nk_draw_image(out, *icon, img, nk_white);
20725         else nk_draw_symbol(out, sym, *icon, text.background, text.text, 1, font);
20726     }
20727     nk_widget_text(out, *bounds, string, len, &text, align, font);
20728 }
20729 NK_LIB int
20730 nk_do_selectable(nk_flags *state, struct nk_command_buffer *out,
20731     struct nk_rect bounds, const char *str, int len, nk_flags align, int *value,
20732     const struct nk_style_selectable *style, const struct nk_input *in,
20733     const struct nk_user_font *font)
20734 {
20735     int old_value;
20736     struct nk_rect touch;
20737
20738     NK_ASSERT(state);
20739     NK_ASSERT(out);
20740     NK_ASSERT(str);
20741     NK_ASSERT(len);
20742     NK_ASSERT(value);
20743     NK_ASSERT(style);
20744     NK_ASSERT(font);
20745
20746     if (!state || !out || !str || !len || !value || !style || !font) return 0;
20747     old_value = *value;
20748
20749     /* remove padding */
20750     touch.x = bounds.x - style->touch_padding.x;
20751     touch.y = bounds.y - style->touch_padding.y;
20752     touch.w = bounds.w + style->touch_padding.x * 2;
20753     touch.h = bounds.h + style->touch_padding.y * 2;
20754
20755     /* update button */
20756     if (nk_button_behavior(state, touch, in, NK_BUTTON_DEFAULT))
20757         *value = !(*value);
20758
20759     /* draw selectable */
20760     if (style->draw_begin) style->draw_begin(out, style->userdata);
20761     nk_draw_selectable(out, *state, style, *value, &bounds, 0,0,NK_SYMBOL_NONE, str, len, align,
      font);
20762     if (style->draw_end) style->draw_end(out, style->userdata);
20763     return old_value != *value;
20764 }
20765 NK_LIB int
20766 nk_do_selectable_image(nk_flags *state, struct nk_command_buffer *out,
20767     struct nk_rect bounds, const char *str, int len, nk_flags align, int *value,
20768     const struct nk_image *img, const struct nk_style_selectable *style,
20769     const struct nk_input *in, const struct nk_user_font *font)
20770 {
20771     int old_value;
20772     struct nk_rect touch;
20773     struct nk_rect icon;
20774
20775     NK_ASSERT(state);
20776     NK_ASSERT(out);
20777     NK_ASSERT(str);
20778     NK_ASSERT(len);
20779     NK_ASSERT(value);
20780     NK_ASSERT(style);
20781     NK_ASSERT(font);
20782
20783     if (!state || !out || !str || !len || !value || !style || !font) return 0;
20784     old_value = *value;
20785
20786     /* toggle behavior */
20787     touch.x = bounds.x - style->touch_padding.x;
20788     touch.y = bounds.y - style->touch_padding.y;
20789     touch.w = bounds.w + style->touch_padding.x * 2;
20790     touch.h = bounds.h + style->touch_padding.y * 2;
20791     if (nk_button_behavior(state, touch, in, NK_BUTTON_DEFAULT))
20792         *value = !(*value);
20793
20794     icon.y = bounds.y + style->padding.y;
20795     icon.w = icon.h = bounds.h - 2 * style->padding.y;
20796     if (align & NK_TEXT_ALIGN_LEFT) {
20797         icon.x = (bounds.x + bounds.w) - (2 * style->padding.x + icon.w);
20798         icon.x = NK_MAX(icon.x, 0);
20799     } else icon.x = bounds.x + 2 * style->padding.x;
20800
20801     icon.x += style->image_padding.x;
20802     icon.y += style->image_padding.y;
20803     icon.w -= 2 * style->image_padding.x;
20804     icon.h -= 2 * style->image_padding.y;
20805
20806     /* draw selectable */
20807     if (style->draw_begin) style->draw_begin(out, style->userdata);
```

```
20808     nk_draw_selectable(out, *state, style, *value, &bounds, &icon, img, NK_SYMBOL_NONE, str, len,
      align, font);
20809     if (style->draw_end) style->draw_end(out, style->userdata);
20810     return old_value != *value;
20811 }
20812 NK_LIB int
20813 nk_do_selectable_symbol(nk_flags *state, struct nk_command_buffer *out,
20814     struct nk_rect bounds, const char *str, int len, nk_flags align, int *value,
20815     enum nk_symbol_type sym, const struct nk_style_selectable *style,
20816     const struct nk_input *in, const struct nk_user_font *font)
20817 {
20818     int old_value;
20819     struct nk_rect touch;
20820     struct nk_rect icon;
20821
20822     NK_ASSERT(state);
20823     NK_ASSERT(out);
20824     NK_ASSERT(str);
20825     NK_ASSERT(len);
20826     NK_ASSERT(value);
20827     NK_ASSERT(style);
20828     NK_ASSERT(font);
20829
20830     if (!state || !out || !str || !len || !value || !style || !font) return 0;
20831     old_value = *value;
20832
20833     /* toggle behavior */
20834     touch.x = bounds.x - style->touch_padding.x;
20835     touch.y = bounds.y - style->touch_padding.y;
20836     touch.w = bounds.w + style->touch_padding.x * 2;
20837     touch.h = bounds.h + style->touch_padding.y * 2;
20838     if (nk_button_behavior(state, touch, in, NK_BUTTON_DEFAULT))
20839         *value = !(*value);
20840
20841     icon.y = bounds.y + style->padding.y;
20842     icon.w = icon.h = bounds.h - 2 * style->padding.y;
20843     if (align & NK_TEXT_ALIGN_LEFT) {
20844         icon.x = (bounds.x + bounds.w) - (2 * style->padding.x + icon.w);
20845         icon.x = NK_MAX(icon.x, 0);
20846     } else icon.x = bounds.x + 2 * style->padding.x;
20847
20848     icon.x += style->image_padding.x;
20849     icon.y += style->image_padding.y;
20850     icon.w -= 2 * style->image_padding.x;
20851     icon.h -= 2 * style->image_padding.y;
20852
20853     /* draw selectable */
20854     if (style->draw_begin) style->draw_begin(out, style->userdata);
20855     nk_draw_selectable(out, *state, style, *value, &bounds, &icon, 0, sym, str, len, align, font);
20856     if (style->draw_end) style->draw_end(out, style->userdata);
20857     return old_value != *value;
20858 }
20859
20860 NK_API int
20861 nk_selectable_text(struct nk_context *ctx, const char *str, int len,
20862     nk_flags align, int *value)
20863 {
20864     struct nk_window *win;
20865     struct nk_panel *layout;
20866     const struct nk_input *in;
20867     const struct nk_style *style;
20868
20869     enum nk_widget_layout_states state;
20870     struct nk_rect bounds;
20871
20872     NK_ASSERT(ctx);
20873     NK_ASSERT(value);
20874     NK_ASSERT(ctx->current);
20875     NK_ASSERT(ctx->current->layout);
20876     if (!ctx || !ctx->current || !ctx->current->layout || !value)
20877         return 0;
20878
20879     win = ctx->current;
20880     layout = win->layout;
20881     style = &ctx->style;
20882
20883     state = nk_widget(&bounds, ctx);
20884     if (!state) return 0;
20885     in = (state == NK_WIDGET_ROM || layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
20886     return nk_do_selectable(&ctx->last_widget_state, &win->buffer, bounds,
20887                 str, len, align, value, &style->selectable, in, style->font);
20888 }
20889 NK_API int
20890 nk_selectable_image_text(struct nk_context *ctx, struct nk_image img,
20891     const char *str, int len, nk_flags align, int *value)
20892 {
20893     struct nk_window *win;
```

```
20894     struct nk_panel *layout;
20895     const struct nk_input *in;
20896     const struct nk_style *style;
20897
20898     enum nk_widget_layout_states state;
20899     struct nk_rect bounds;
20900
20901     NK_ASSERT(ctx);
20902     NK_ASSERT(value);
20903     NK_ASSERT(ctx->current);
20904     NK_ASSERT(ctx->current->layout);
20905     if (!ctx || !ctx->current || !ctx->current->layout || !value)
20906         return 0;
20907
20908     win = ctx->current;
20909     layout = win->layout;
20910     style = &ctx->style;
20911
20912     state = nk_widget(&bounds, ctx);
20913     if (!state) return 0;
20914     in = (state == NK_WIDGET_ROM || layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
20915     return nk_do_selectable_image(&ctx->last_widget_state, &win->buffer, bounds,
20916                 str, len, align, value, &img, &style->selectable, in, style->font);
20917 }
20918 NK_API int
20919 nk_selectable_symbol_text(struct nk_context *ctx, enum nk_symbol_type sym,
20920     const char *str, int len, nk_flags align, int *value)
20921 {
20922     struct nk_window *win;
20923     struct nk_panel *layout;
20924     const struct nk_input *in;
20925     const struct nk_style *style;
20926
20927     enum nk_widget_layout_states state;
20928     struct nk_rect bounds;
20929
20930     NK_ASSERT(ctx);
20931     NK_ASSERT(value);
20932     NK_ASSERT(ctx->current);
20933     NK_ASSERT(ctx->current->layout);
20934     if (!ctx || !ctx->current || !ctx->current->layout || !value)
20935         return 0;
20936
20937     win = ctx->current;
20938     layout = win->layout;
20939     style = &ctx->style;
20940
20941     state = nk_widget(&bounds, ctx);
20942     if (!state) return 0;
20943     in = (state == NK_WIDGET_ROM || layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
20944     return nk_do_selectable_symbol(&ctx->last_widget_state, &win->buffer, bounds,
20945                 str, len, align, value, sym, &style->selectable, in, style->font);
20946 }
20947 NK_API int
20948 nk_selectable_symbol_label(struct nk_context *ctx, enum nk_symbol_type sym,
20949     const char *title, nk_flags align, int *value)
20950 {
20951     return nk_selectable_symbol_text(ctx, sym, title, nk_strlen(title), align, value);
20952 }
20953 NK_API int nk_select_text(struct nk_context *ctx, const char *str, int len,
20954     nk_flags align, int value)
20955 {
20956     nk_selectable_text(ctx, str, len, align, &value);return value;
20957 }
20958 NK_API int nk_selectable_label(struct nk_context *ctx, const char *str, nk_flags align, int *value)
20959 {
20960     return nk_selectable_text(ctx, str, nk_strlen(str), align, value);
20961 }
20962 NK_API int nk_selectable_image_label(struct nk_context *ctx,struct nk_image img,
20963     const char *str, nk_flags align, int *value)
20964 {
20965     return nk_selectable_image_text(ctx, img, str, nk_strlen(str), align, value);
20966 }
20967 NK_API int nk_select_label(struct nk_context *ctx, const char *str, nk_flags align, int value)
20968 {
20969     nk_selectable_text(ctx, str, nk_strlen(str), align, &value);return value;
20970 }
20971 NK_API int nk_select_image_label(struct nk_context *ctx, struct nk_image img,
20972     const char *str, nk_flags align, int value)
20973 {
20974     nk_selectable_image_text(ctx, img, str, nk_strlen(str), align, &value);return value;
20975 }
20976 NK_API int nk_select_image_text(struct nk_context *ctx, struct nk_image img,
20977     const char *str, int len, nk_flags align, int value)
20978 {
20979     nk_selectable_image_text(ctx, img, str, len, align, &value);return value;
20980 }
```

```
20981 NK_API int
20982 nk_select_symbol_text(struct nk_context *ctx, enum nk_symbol_type sym,
20983     const char *title, int title_len, nk_flags align, int value)
20984 {
20985     nk_selectable_symbol_text(ctx, sym, title, title_len, align, &value);return value;
20986 }
20987 NK_API int
20988 nk_select_symbol_label(struct nk_context *ctx, enum nk_symbol_type sym,
20989     const char *title, nk_flags align, int value)
20990 {
20991     return nk_select_symbol_text(ctx, sym, title, nk_strlen(title), align, value);
20992 }
20993
20994
20995
20996
20997
20998 /* ==============================================================
20999  *
21000  *                              SLIDER
21001  *
21002  * ==============================================================*/
21003 NK_LIB float
21004 nk_slider_behavior(nk_flags *state, struct nk_rect *logical_cursor,
21005     struct nk_rect *visual_cursor, struct nk_input *in,
21006     struct nk_rect bounds, float slider_min, float slider_max, float slider_value,
21007     float slider_step, float slider_steps)
21008 {
21009     int left_mouse_down;
21010     int left_mouse_click_in_cursor;
21011
21012     /* check if visual cursor is being dragged */
21013     nk_widget_state_reset(state);
21014     left_mouse_down = in && in->mouse.buttons[NK_BUTTON_LEFT].down;
21015     left_mouse_click_in_cursor = in && nk_input_has_mouse_click_down_in_rect(in,
21016             NK_BUTTON_LEFT, *visual_cursor, nk_true);
21017
21018     if (left_mouse_down && left_mouse_click_in_cursor) {
21019         float ratio = 0;
21020         const float d = in->mouse.pos.x - (visual_cursor->x+visual_cursor->w*0.5f);
21021         const float pxstep = bounds.w / slider_steps;
21022
21023         /* only update value if the next slider step is reached */
21024         *state = NK_WIDGET_STATE_ACTIVE;
21025         if (NK_ABS(d) >= pxstep) {
21026             const float steps = (float)((int)(NK_ABS(d) / pxstep));
21027             slider_value += (d > 0) ? (slider_step*steps) : -(slider_step*steps);
21028             slider_value = NK_CLAMP(slider_min, slider_value, slider_max);
21029             ratio = (slider_value - slider_min)/slider_step;
21030             logical_cursor->x = bounds.x + (logical_cursor->w * ratio);
21031             in->mouse.buttons[NK_BUTTON_LEFT].clicked_pos.x = logical_cursor->x;
21032         }
21033     }
21034
21035     /* slider widget state */
21036     if (nk_input_is_mouse_hovering_rect(in, bounds))
21037         *state = NK_WIDGET_STATE_HOVERED;
21038     if (*state & NK_WIDGET_STATE_HOVER &&
21039         !nk_input_is_mouse_prev_hovering_rect(in, bounds))
21040         *state |= NK_WIDGET_STATE_ENTERED;
21041     else if (nk_input_is_mouse_prev_hovering_rect(in, bounds))
21042         *state |= NK_WIDGET_STATE_LEFT;
21043     return slider_value;
21044 }
21045 NK_LIB void
21046 nk_draw_slider(struct nk_command_buffer *out, nk_flags state,
21047     const struct nk_style_slider *style, const struct nk_rect *bounds,
21048     const struct nk_rect *visual_cursor, float min, float value, float max)
21049 {
21050     struct nk_rect fill;
21051     struct nk_rect bar;
21052     const struct nk_style_item *background;
21053
21054     /* select correct slider images/colors */
21055     struct nk_color bar_color;
21056     const struct nk_style_item *cursor;
21057
21058     NK_UNUSED(min);
21059     NK_UNUSED(max);
21060     NK_UNUSED(value);
21061
21062     if (state & NK_WIDGET_STATE_ACTIVED) {
21063         background = &style->active;
21064         bar_color = style->bar_active;
21065         cursor = &style->cursor_active;
21066     } else if (state & NK_WIDGET_STATE_HOVER) {
21067         background = &style->hover;
```

```
21068         bar_color = style->bar_hover;
21069         cursor = &style->cursor_hover;
21070     } else {
21071         background = &style->normal;
21072         bar_color = style->bar_normal;
21073         cursor = &style->cursor_normal;
21074     }
21075     /* calculate slider background bar */
21076     bar.x = bounds->x;
21077     bar.y = (visual_cursor->y + visual_cursor->h/2) - bounds->h/12;
21078     bar.w = bounds->w;
21079     bar.h = bounds->h/6;
21080
21081     /* filled background bar style */
21082     fill.w = (visual_cursor->x + (visual_cursor->w/2.0f)) - bar.x;
21083     fill.x = bar.x;
21084     fill.y = bar.y;
21085     fill.h = bar.h;
21086
21087     /* draw background */
21088     if (background->type == NK_STYLE_ITEM_IMAGE) {
21089         nk_draw_image(out, *bounds, &background->data.image, nk_white);
21090     } else {
21091         nk_fill_rect(out, *bounds, style->rounding, background->data.color);
21092         nk_stroke_rect(out, *bounds, style->rounding, style->border, style->border_color);
21093     }
21094
21095     /* draw slider bar */
21096     nk_fill_rect(out, bar, style->rounding, bar_color);
21097     nk_fill_rect(out, fill, style->rounding, style->bar_filled);
21098
21099     /* draw cursor */
21100     if (cursor->type == NK_STYLE_ITEM_IMAGE)
21101         nk_draw_image(out, *visual_cursor, &cursor->data.image, nk_white);
21102     else nk_fill_circle(out, *visual_cursor, cursor->data.color);
21103 }
21104 NK_LIB float
21105 nk_do_slider(nk_flags *state,
21106     struct nk_command_buffer *out, struct nk_rect bounds,
21107     float min, float val, float max, float step,
21108     const struct nk_style_slider *style, struct nk_input *in,
21109     const struct nk_user_font *font)
21110 {
21111     float slider_range;
21112     float slider_min;
21113     float slider_max;
21114     float slider_value;
21115     float slider_steps;
21116     float cursor_offset;
21117
21118     struct nk_rect visual_cursor;
21119     struct nk_rect logical_cursor;
21120
21121     NK_ASSERT(style);
21122     NK_ASSERT(out);
21123     if (!out || !style)
21124         return 0;
21125
21126     /* remove padding from slider bounds */
21127     bounds.x = bounds.x + style->padding.x;
21128     bounds.y = bounds.y + style->padding.y;
21129     bounds.h = NK_MAX(bounds.h, 2*style->padding.y);
21130     bounds.w = NK_MAX(bounds.w, 2*style->padding.x + style->cursor_size.x);
21131     bounds.w -= 2 * style->padding.x;
21132     bounds.h -= 2 * style->padding.y;
21133
21134     /* optional buttons */
21135     if (style->show_buttons) {
21136         nk_flags ws;
21137         struct nk_rect button;
21138         button.y = bounds.y;
21139         button.w = bounds.h;
21140         button.h = bounds.h;
21141
21142         /* decrement button */
21143         button.x = bounds.x;
21144         if (nk_do_button_symbol(&ws, out, button, style->dec_symbol, NK_BUTTON_DEFAULT,
21145             &style->dec_button, in, font))
21146             val -= step;
21147
21148         /* increment button */
21149         button.x = (bounds.x + bounds.w) - button.w;
21150         if (nk_do_button_symbol(&ws, out, button, style->inc_symbol, NK_BUTTON_DEFAULT,
21151             &style->inc_button, in, font))
21152             val += step;
21153
21154         bounds.x = bounds.x + button.w + style->spacing.x;
```

```
21155         bounds.w = bounds.w - (2*button.w + 2*style->spacing.x);
21156     }
21157
21158     /* remove one cursor size to support visual cursor */
21159     bounds.x += style->cursor_size.x*0.5f;
21160     bounds.w -= style->cursor_size.x;
21161
21162     /* make sure the provided values are correct */
21163     slider_max = NK_MAX(min, max);
21164     slider_min = NK_MIN(min, max);
21165     slider_value = NK_CLAMP(slider_min, val, slider_max);
21166     slider_range = slider_max - slider_min;
21167     slider_steps = slider_range / step;
21168     cursor_offset = (slider_value - slider_min) / step;
21169
21170     /* calculate cursor
21171     Basically you have two cursors. One for visual representation and interaction
21172     and one for updating the actual cursor value. */
21173     logical_cursor.h = bounds.h;
21174     logical_cursor.w = bounds.w / slider_steps;
21175     logical_cursor.x = bounds.x + (logical_cursor.w * cursor_offset);
21176     logical_cursor.y = bounds.y;
21177
21178     visual_cursor.h = style->cursor_size.y;
21179     visual_cursor.w = style->cursor_size.x;
21180     visual_cursor.y = (bounds.y + bounds.h*0.5f) - visual_cursor.h*0.5f;
21181     visual_cursor.x = logical_cursor.x - visual_cursor.w*0.5f;
21182
21183     slider_value = nk_slider_behavior(state, &logical_cursor, &visual_cursor,
21184         in, bounds, slider_min, slider_max, slider_value, step, slider_steps);
21185     visual_cursor.x = logical_cursor.x - visual_cursor.w*0.5f;
21186
21187     /* draw slider */
21188     if (style->draw_begin) style->draw_begin(out, style->userdata);
21189     nk_draw_slider(out, *state, style, &bounds, &visual_cursor, slider_min, slider_value, slider_max);
21190     if (style->draw_end) style->draw_end(out, style->userdata);
21191     return slider_value;
21192 }
21193 NK_API int
21194 nk_slider_float(struct nk_context *ctx, float min_value, float *value, float max_value,
21195     float value_step)
21196 {
21197     struct nk_window *win;
21198     struct nk_panel *layout;
21199     struct nk_input *in;
21200     const struct nk_style *style;
21201
21202     int ret = 0;
21203     float old_value;
21204     struct nk_rect bounds;
21205     enum nk_widget_layout_states state;
21206
21207     NK_ASSERT(ctx);
21208     NK_ASSERT(ctx->current);
21209     NK_ASSERT(ctx->current->layout);
21210     NK_ASSERT(value);
21211     if (!ctx || !ctx->current || !ctx->current->layout || !value)
21212         return ret;
21213
21214     win = ctx->current;
21215     style = &ctx->style;
21216     layout = win->layout;
21217
21218     state = nk_widget(&bounds, ctx);
21219     if (!state) return ret;
21220     in = (/*state == NK_WIDGET_ROM || */ layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
21221
21222     old_value = *value;
21223     *value = nk_do_slider(&ctx->last_widget_state, &win->buffer, bounds, min_value,
21224                 old_value, max_value, value_step, &style->slider, in, style->font);
21225     return (old_value > *value || old_value < *value);
21226 }
21227 NK_API float
21228 nk_slide_float(struct nk_context *ctx, float min, float val, float max, float step)
21229 {
21230     nk_slider_float(ctx, min, &val, max, step); return val;
21231 }
21232 NK_API int
21233 nk_slide_int(struct nk_context *ctx, int min, int val, int max, int step)
21234 {
21235     float value = (float)val;
21236     nk_slider_float(ctx, (float)min, &value, (float)max, (float)step);
21237     return (int)value;
21238 }
21239 NK_API int
21240 nk_slider_int(struct nk_context *ctx, int min, int *val, int max, int step)
21241 {
```

```
21242     int ret;
21243     float value = (float)*val;
21244     ret = nk_slider_float(ctx, (float)min, &value, (float)max, (float)step);
21245     *val =  (int)value;
21246     return ret;
21247 }
21248
21249
21250
21251
21252
21253 /* ===============================================================
21254  *
21255  *                          PROGRESS
21256  *
21257  * ===============================================================*/
21258 NK_LIB nk_size
21259 nk_progress_behavior(nk_flags *state, struct nk_input *in,
21260     struct nk_rect r, struct nk_rect cursor, nk_size max, nk_size value, int modifiable)
21261 {
21262     int left_mouse_down = 0;
21263     int left_mouse_click_in_cursor = 0;
21264
21265     nk_widget_state_reset(state);
21266     if (!in || !modifiable) return value;
21267     left_mouse_down = in && in->mouse.buttons[NK_BUTTON_LEFT].down;
21268     left_mouse_click_in_cursor = in && nk_input_has_mouse_click_down_in_rect(in,
21269             NK_BUTTON_LEFT, cursor, nk_true);
21270     if (nk_input_is_mouse_hovering_rect(in, r))
21271         *state = NK_WIDGET_STATE_HOVERED;
21272
21273     if (in && left_mouse_down && left_mouse_click_in_cursor) {
21274         if (left_mouse_down && left_mouse_click_in_cursor) {
21275             float ratio = NK_MAX(0, (float)(in->mouse.pos.x - cursor.x)) / (float)cursor.w;
21276             value = (nk_size)NK_CLAMP(0, (float)max * ratio, (float)max);
21277             in->mouse.buttons[NK_BUTTON_LEFT].clicked_pos.x = cursor.x + cursor.w/2.0f;
21278             *state |= NK_WIDGET_STATE_ACTIVE;
21279         }
21280     }
21281     /* set progressbar widget state */
21282     if (*state & NK_WIDGET_STATE_HOVER && !nk_input_is_mouse_prev_hovering_rect(in, r))
21283         *state |= NK_WIDGET_STATE_ENTERED;
21284     else if (nk_input_is_mouse_prev_hovering_rect(in, r))
21285         *state |= NK_WIDGET_STATE_LEFT;
21286     return value;
21287 }
21288 NK_LIB void
21289 nk_draw_progress(struct nk_command_buffer *out, nk_flags state,
21290     const struct nk_style_progress *style, const struct nk_rect *bounds,
21291     const struct nk_rect *scursor, nk_size value, nk_size max)
21292 {
21293     const struct nk_style_item *background;
21294     const struct nk_style_item *cursor;
21295
21296     NK_UNUSED(max);
21297     NK_UNUSED(value);
21298
21299     /* select correct colors/images to draw */
21300     if (state & NK_WIDGET_STATE_ACTIVED) {
21301         background = &style->active;
21302         cursor = &style->cursor_active;
21303     } else if (state & NK_WIDGET_STATE_HOVER){
21304         background = &style->hover;
21305         cursor = &style->cursor_hover;
21306     } else {
21307         background = &style->normal;
21308         cursor = &style->cursor_normal;
21309     }
21310
21311     /* draw background */
21312     if (background->type == NK_STYLE_ITEM_COLOR) {
21313         nk_fill_rect(out, *bounds, style->rounding, background->data.color);
21314         nk_stroke_rect(out, *bounds, style->rounding, style->border, style->border_color);
21315     } else nk_draw_image(out, *bounds, &background->data.image, nk_white);
21316
21317     /* draw cursor */
21318     if (cursor->type == NK_STYLE_ITEM_COLOR) {
21319         nk_fill_rect(out, *scursor, style->rounding, cursor->data.color);
21320         nk_stroke_rect(out, *scursor, style->rounding, style->border, style->border_color);
21321     } else nk_draw_image(out, *scursor, &cursor->data.image, nk_white);
21322 }
21323 NK_LIB nk_size
21324 nk_do_progress(nk_flags *state,
21325     struct nk_command_buffer *out, struct nk_rect bounds,
21326     nk_size value, nk_size max, int modifiable,
21327     const struct nk_style_progress *style, struct nk_input *in)
21328 {
```

```
21329     float prog_scale;
21330     nk_size prog_value;
21331     struct nk_rect cursor;
21332
21333     NK_ASSERT(style);
21334     NK_ASSERT(out);
21335     if (!out || !style) return 0;
21336
21337     /* calculate progressbar cursor */
21338     cursor.w = NK_MAX(bounds.w, 2 * style->padding.x + 2 * style->border);
21339     cursor.h = NK_MAX(bounds.h, 2 * style->padding.y + 2 * style->border);
21340     cursor = nk_pad_rect(bounds, nk_vec2(style->padding.x + style->border, style->padding.y +
      style->border));
21341     prog_scale = (float)value / (float)max;
21342
21343     /* update progressbar */
21344     prog_value = NK_MIN(value, max);
21345     prog_value = nk_progress_behavior(state, in, bounds, cursor,max, prog_value, modifiable);
21346     cursor.w = cursor.w * prog_scale;
21347
21348     /* draw progressbar */
21349     if (style->draw_begin) style->draw_begin(out, style->userdata);
21350     nk_draw_progress(out, *state, style, &bounds, &cursor, value, max);
21351     if (style->draw_end) style->draw_end(out, style->userdata);
21352     return prog_value;
21353 }
21354 NK_API int
21355 nk_progress(struct nk_context *ctx, nk_size *cur, nk_size max, int is_modifyable)
21356 {
21357     struct nk_window *win;
21358     struct nk_panel *layout;
21359     const struct nk_style *style;
21360     struct nk_input *in;
21361
21362     struct nk_rect bounds;
21363     enum nk_widget_layout_states state;
21364     nk_size old_value;
21365
21366     NK_ASSERT(ctx);
21367     NK_ASSERT(cur);
21368     NK_ASSERT(ctx->current);
21369     NK_ASSERT(ctx->current->layout);
21370     if (!ctx || !ctx->current || !ctx->current->layout || !cur)
21371         return 0;
21372
21373     win = ctx->current;
21374     style = &ctx->style;
21375     layout = win->layout;
21376     state = nk_widget(&bounds, ctx);
21377     if (!state) return 0;
21378
21379     in = (state == NK_WIDGET_ROM || layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
21380     old_value = *cur;
21381     *cur = nk_do_progress(&ctx->last_widget_state, &win->buffer, bounds,
21382             *cur, max, is_modifyable, &style->progress, in);
21383     return (*cur != old_value);
21384 }
21385 NK_API nk_size
21386 nk_prog(struct nk_context *ctx, nk_size cur, nk_size max, int modifyable)
21387 {
21388     nk_progress(ctx, &cur, max, modifyable);
21389     return cur;
21390 }
21391
21392
21393
21394
21395
21396 /* ===============================================================
21397  *
21398  *                              SCROLLBAR
21399  *
21400  * ===============================================================*/
21401 NK_LIB float
21402 nk_scrollbar_behavior(nk_flags *state, struct nk_input *in,
21403     int has_scrolling, const struct nk_rect *scroll,
21404     const struct nk_rect *cursor, const struct nk_rect *empty0,
21405     const struct nk_rect *empty1, float scroll_offset,
21406     float target, float scroll_step, enum nk_orientation o)
21407 {
21408     nk_flags ws = 0;
21409     int left_mouse_down;
21410     int left_mouse_clicked;
21411     int left_mouse_click_in_cursor;
21412     float scroll_delta;
21413
21414     nk_widget_state_reset(state);
```

```
21415     if (!in) return scroll_offset;
21416
21417     left_mouse_down = in->mouse.buttons[NK_BUTTON_LEFT].down;
21418     left_mouse_clicked = in->mouse.buttons[NK_BUTTON_LEFT].clicked;
21419     left_mouse_click_in_cursor = nk_input_has_mouse_click_down_in_rect(in,
21420         NK_BUTTON_LEFT, *cursor, nk_true);
21421     if (nk_input_is_mouse_hovering_rect(in, *scroll))
21422         *state = NK_WIDGET_STATE_HOVERED;
21423
21424     scroll_delta = (o == NK_VERTICAL) ? in->mouse.scroll_delta.y: in->mouse.scroll_delta.x;
21425     if (left_mouse_down && left_mouse_click_in_cursor && !left_mouse_clicked) {
21426         /* update cursor by mouse dragging */
21427         float pixel, delta;
21428         *state = NK_WIDGET_STATE_ACTIVE;
21429         if (o == NK_VERTICAL) {
21430             float cursor_y;
21431             pixel = in->mouse.delta.y;
21432             delta = (pixel / scroll->h) * target;
21433             scroll_offset = NK_CLAMP(0, scroll_offset + delta, target - scroll->h);
21434             cursor_y = scroll->y + ((scroll_offset/target) * scroll->h);
21435             in->mouse.buttons[NK_BUTTON_LEFT].clicked_pos.y = cursor_y + cursor->h/2.0f;
21436         } else {
21437             float cursor_x;
21438             pixel = in->mouse.delta.x;
21439             delta = (pixel / scroll->w) * target;
21440             scroll_offset = NK_CLAMP(0, scroll_offset + delta, target - scroll->w);
21441             cursor_x = scroll->x + ((scroll_offset/target) * scroll->w);
21442             in->mouse.buttons[NK_BUTTON_LEFT].clicked_pos.x = cursor_x + cursor->w/2.0f;
21443         }
21444     } else if ((nk_input_is_key_pressed(in, NK_KEY_SCROLL_UP) && o == NK_VERTICAL && has_scrolling)||
21445             nk_button_behavior(&ws, *empty0, in, NK_BUTTON_DEFAULT)) {
21446         /* scroll page up by click on empty space or shortcut */
21447         if (o == NK_VERTICAL)
21448             scroll_offset = NK_MAX(0, scroll_offset - scroll->h);
21449         else scroll_offset = NK_MAX(0, scroll_offset - scroll->w);
21450     } else if ((nk_input_is_key_pressed(in, NK_KEY_SCROLL_DOWN) && o == NK_VERTICAL && has_scrolling)
      ||
21451         nk_button_behavior(&ws, *empty1, in, NK_BUTTON_DEFAULT)) {
21452         /* scroll page down by click on empty space or shortcut */
21453         if (o == NK_VERTICAL)
21454             scroll_offset = NK_MIN(scroll_offset + scroll->h, target - scroll->h);
21455         else scroll_offset = NK_MIN(scroll_offset + scroll->w, target - scroll->w);
21456     } else if (has_scrolling) {
21457         if ((scroll_delta < 0 || (scroll_delta > 0))) {
21458             /* update cursor by mouse scrolling */
21459             scroll_offset = scroll_offset + scroll_step * (-scroll_delta);
21460             if (o == NK_VERTICAL)
21461                 scroll_offset = NK_CLAMP(0, scroll_offset, target - scroll->h);
21462             else scroll_offset = NK_CLAMP(0, scroll_offset, target - scroll->w);
21463         } else if (nk_input_is_key_pressed(in, NK_KEY_SCROLL_START)) {
21464             /* update cursor to the beginning  */
21465             if (o == NK_VERTICAL) scroll_offset = 0;
21466         } else if (nk_input_is_key_pressed(in, NK_KEY_SCROLL_END)) {
21467             /* update cursor to the end */
21468             if (o == NK_VERTICAL) scroll_offset = target - scroll->h;
21469         }
21470     }
21471     if (*state & NK_WIDGET_STATE_HOVER && !nk_input_is_mouse_prev_hovering_rect(in, *scroll))
21472         *state |= NK_WIDGET_STATE_ENTERED;
21473     else if (nk_input_is_mouse_prev_hovering_rect(in, *scroll))
21474         *state |= NK_WIDGET_STATE_LEFT;
21475     return scroll_offset;
21476 }
21477 NK_LIB void
21478 nk_draw_scrollbar(struct nk_command_buffer *out, nk_flags state,
21479     const struct nk_style_scrollbar *style, const struct nk_rect *bounds,
21480     const struct nk_rect *scroll)
21481 {
21482     const struct nk_style_item *background;
21483     const struct nk_style_item *cursor;
21484
21485     /* select correct colors/images to draw */
21486     if (state & NK_WIDGET_STATE_ACTIVED) {
21487         background = &style->active;
21488         cursor = &style->cursor_active;
21489     } else if (state & NK_WIDGET_STATE_HOVER) {
21490         background = &style->hover;
21491         cursor = &style->cursor_hover;
21492     } else {
21493         background = &style->normal;
21494         cursor = &style->cursor_normal;
21495     }
21496
21497     /* draw background */
21498     if (background->type == NK_STYLE_ITEM_COLOR) {
21499         nk_fill_rect(out, *bounds, style->rounding, background->data.color);
21500         nk_stroke_rect(out, *bounds, style->rounding, style->border, style->border_color);
```

```
21501     } else {
21502         nk_draw_image(out, *bounds, &background->data.image, nk_white);
21503     }
21504
21505     /* draw cursor */
21506     if (cursor->type == NK_STYLE_ITEM_COLOR) {
21507         nk_fill_rect(out, *scroll, style->rounding_cursor, cursor->data.color);
21508         nk_stroke_rect(out, *scroll, style->rounding_cursor, style->border_cursor,
      style->cursor_border_color);
21509     } else nk_draw_image(out, *scroll, &cursor->data.image, nk_white);
21510 }
21511 NK_LIB float
21512 nk_do_scrollbarv(nk_flags *state,
21513     struct nk_command_buffer *out, struct nk_rect scroll, int has_scrolling,
21514     float offset, float target, float step, float button_pixel_inc,
21515     const struct nk_style_scrollbar *style, struct nk_input *in,
21516     const struct nk_user_font *font)
21517 {
21518     struct nk_rect empty_north;
21519     struct nk_rect empty_south;
21520     struct nk_rect cursor;
21521
21522     float scroll_step;
21523     float scroll_offset;
21524     float scroll_off;
21525     float scroll_ratio;
21526
21527     NK_ASSERT(out);
21528     NK_ASSERT(style);
21529     NK_ASSERT(state);
21530     if (!out || !style) return 0;
21531
21532     scroll.w = NK_MAX(scroll.w, 1);
21533     scroll.h = NK_MAX(scroll.h, 0);
21534     if (target <= scroll.h) return 0;
21535
21536     /* optional scrollbar buttons */
21537     if (style->show_buttons) {
21538         nk_flags ws;
21539         float scroll_h;
21540         struct nk_rect button;
21541
21542         button.x = scroll.x;
21543         button.w = scroll.w;
21544         button.h = scroll.w;
21545
21546         scroll_h = NK_MAX(scroll.h - 2 * button.h,0);
21547         scroll_step = NK_MIN(step, button_pixel_inc);
21548
21549         /* decrement button */
21550         button.y = scroll.y;
21551         if (nk_do_button_symbol(&ws, out, button, style->dec_symbol,
21552             NK_BUTTON_REPEATER, &style->dec_button, in, font))
21553             offset = offset - scroll_step;
21554
21555         /* increment button */
21556         button.y = scroll.y + scroll.h - button.h;
21557         if (nk_do_button_symbol(&ws, out, button, style->inc_symbol,
21558             NK_BUTTON_REPEATER, &style->inc_button, in, font))
21559             offset = offset + scroll_step;
21560
21561         scroll.y = scroll.y + button.h;
21562         scroll.h = scroll_h;
21563     }
21564
21565     /* calculate scrollbar constants */
21566     scroll_step = NK_MIN(step, scroll.h);
21567     scroll_offset = NK_CLAMP(0, offset, target - scroll.h);
21568     scroll_ratio = scroll.h / target;
21569     scroll_off = scroll_offset / target;
21570
21571     /* calculate scrollbar cursor bounds */
21572     cursor.h = NK_MAX((scroll_ratio * scroll.h) - (2*style->border + 2*style->padding.y), 0);
21573     cursor.y = scroll.y + (scroll_off * scroll.h) + style->border + style->padding.y;
21574     cursor.w = scroll.w - (2 * style->border + 2 * style->padding.x);
21575     cursor.x = scroll.x + style->border + style->padding.x;
21576
21577     /* calculate empty space around cursor */
21578     empty_north.x = scroll.x;
21579     empty_north.y = scroll.y;
21580     empty_north.w = scroll.w;
21581     empty_north.h = NK_MAX(cursor.y - scroll.y, 0);
21582
21583     empty_south.x = scroll.x;
21584     empty_south.y = cursor.y + cursor.h;
21585     empty_south.w = scroll.w;
21586     empty_south.h = NK_MAX((scroll.y + scroll.h) - (cursor.y + cursor.h), 0);
```

```
21587
21588     /* update scrollbar */
21589     scroll_offset = nk_scrollbar_behavior(state, in, has_scrolling, &scroll, &cursor,
21590         &empty_north, &empty_south, scroll_offset, target, scroll_step, NK_VERTICAL);
21591     scroll_off = scroll_offset / target;
21592     cursor.y = scroll.y + (scroll_off * scroll.h) + style->border_cursor + style->padding.y;
21593
21594     /* draw scrollbar */
21595     if (style->draw_begin) style->draw_begin(out, style->userdata);
21596     nk_draw_scrollbar(out, *state, style, &scroll, &cursor);
21597     if (style->draw_end) style->draw_end(out, style->userdata);
21598     return scroll_offset;
21599 }
21600 NK_LIB float
21601 nk_do_scrollbarh(nk_flags *state,
21602     struct nk_command_buffer *out, struct nk_rect scroll, int has_scrolling,
21603     float offset, float target, float step, float button_pixel_inc,
21604     const struct nk_style_scrollbar *style, struct nk_input *in,
21605     const struct nk_user_font *font)
21606 {
21607     struct nk_rect cursor;
21608     struct nk_rect empty_west;
21609     struct nk_rect empty_east;
21610
21611     float scroll_step;
21612     float scroll_offset;
21613     float scroll_off;
21614     float scroll_ratio;
21615
21616     NK_ASSERT(out);
21617     NK_ASSERT(style);
21618     if (!out || !style) return 0;
21619
21620     /* scrollbar background */
21621     scroll.h = NK_MAX(scroll.h, 1);
21622     scroll.w = NK_MAX(scroll.w, 2 * scroll.h);
21623     if (target <= scroll.w) return 0;
21624
21625     /* optional scrollbar buttons */
21626     if (style->show_buttons) {
21627         nk_flags ws;
21628         float scroll_w;
21629         struct nk_rect button;
21630         button.y = scroll.y;
21631         button.w = scroll.h;
21632         button.h = scroll.h;
21633
21634         scroll_w = scroll.w - 2 * button.w;
21635         scroll_step = NK_MIN(step, button_pixel_inc);
21636
21637         /* decrement button */
21638         button.x = scroll.x;
21639         if (nk_do_button_symbol(&ws, out, button, style->dec_symbol,
21640             NK_BUTTON_REPEATER, &style->dec_button, in, font))
21641             offset = offset - scroll_step;
21642
21643         /* increment button */
21644         button.x = scroll.x + scroll.w - button.w;
21645         if (nk_do_button_symbol(&ws, out, button, style->inc_symbol,
21646             NK_BUTTON_REPEATER, &style->inc_button, in, font))
21647             offset = offset + scroll_step;
21648
21649         scroll.x = scroll.x + button.w;
21650         scroll.w = scroll_w;
21651     }
21652
21653     /* calculate scrollbar constants */
21654     scroll_step = NK_MIN(step, scroll.w);
21655     scroll_offset = NK_CLAMP(0, offset, target - scroll.w);
21656     scroll_ratio = scroll.w / target;
21657     scroll_off = scroll_offset / target;
21658
21659     /* calculate cursor bounds */
21660     cursor.w = (scroll_ratio * scroll.w) - (2*style->border + 2*style->padding.x);
21661     cursor.x = scroll.x + (scroll_off * scroll.w) + style->border + style->padding.x;
21662     cursor.h = scroll.h - (2 * style->border + 2 * style->padding.y);
21663     cursor.y = scroll.y + style->border + style->padding.y;
21664
21665     /* calculate empty space around cursor */
21666     empty_west.x = scroll.x;
21667     empty_west.y = scroll.y;
21668     empty_west.w = cursor.x - scroll.x;
21669     empty_west.h = scroll.h;
21670
21671     empty_east.x = cursor.x + cursor.w;
21672     empty_east.y = scroll.y;
21673     empty_east.w = (scroll.x + scroll.w) - (cursor.x + cursor.w);
```

```
21674     empty_east.h = scroll.h;
21675
21676     /* update scrollbar */
21677     scroll_offset = nk_scrollbar_behavior(state, in, has_scrolling, &scroll, &cursor,
21678         &empty_west, &empty_east, scroll_offset, target, scroll_step, NK_HORIZONTAL);
21679     scroll_off = scroll_offset / target;
21680     cursor.x = scroll.x + (scroll_off * scroll.w);
21681
21682     /* draw scrollbar */
21683     if (style->draw_begin) style->draw_begin(out, style->userdata);
21684     nk_draw_scrollbar(out, *state, style, &scroll, &cursor);
21685     if (style->draw_end) style->draw_end(out, style->userdata);
21686     return scroll_offset;
21687 }
21688
21689
21690
21691
21692
21693 /* ===============================================================
21694  *
21695  *                          TEXT EDITOR
21696  *
21697  * ===============================================================*/
21698 /* stb_textedit.h - v1.8  - public domain - Sean Barrett */
21699 struct nk_text_find {
21700    float x,y;    /* position of n'th character */
21701    float height; /* height of line */
21702    int first_char, length; /* first char of row, and length */
21703    int prev_first;  /*_ first char of previous row */
21704 };
21705
21706 struct nk_text_edit_row {
21707    float x0,x1;
21708    /* starting x location, end x location (allows for align=right, etc) */
21709    float baseline_y_delta;
21710    /* position of baseline relative to previous row's baseline*/
21711    float ymin,ymax;
21712    /* height of row above and below baseline */
21713    int num_chars;
21714 };
21715
21716 /* forward declarations */
21717 NK_INTERN void nk_textedit_makeundo_delete(struct nk_text_edit*, int, int);
21718 NK_INTERN void nk_textedit_makeundo_insert(struct nk_text_edit*, int, int);
21719 NK_INTERN void nk_textedit_makeundo_replace(struct nk_text_edit*, int, int, int);
21720 #define NK_TEXT_HAS_SELECTION(s)   ((s)->select_start != (s)->select_end)
21721
21722 NK_INTERN float
21723 nk_textedit_get_width(const struct nk_text_edit *edit, int line_start, int char_id,
21724     const struct nk_user_font *font)
21725 {
21726     int len = 0;
21727     nk_rune unicode = 0;
21728     const char *str = nk_str_at_const(&edit->string, line_start + char_id, &unicode, &len);
21729     return font->width(font->userdata, font->height, str, len);
21730 }
21731 NK_INTERN void
21732 nk_textedit_layout_row(struct nk_text_edit_row *r, struct nk_text_edit *edit,
21733     int line_start_id, float row_height, const struct nk_user_font *font)
21734 {
21735     int l;
21736     int glyphs = 0;
21737     nk_rune unicode;
21738     const char *remaining;
21739     int len = nk_str_len_char(&edit->string);
21740     const char *end = nk_str_get_const(&edit->string) + len;
21741     const char *text = nk_str_at_const(&edit->string, line_start_id, &unicode, &l);
21742     const struct nk_vec2 size = nk_text_calculate_text_bounds(font,
21743         text, (int)(end - text), row_height, &remaining, 0, &glyphs, NK_STOP_ON_NEW_LINE);
21744
21745     r->x0 = 0.0f;
21746     r->x1 = size.x;
21747     r->baseline_y_delta = size.y;
21748     r->ymin = 0.0f;
21749     r->ymax = size.y;
21750     r->num_chars = glyphs;
21751 }
21752 NK_INTERN int
21753 nk_textedit_locate_coord(struct nk_text_edit *edit, float x, float y,
21754     const struct nk_user_font *font, float row_height)
21755 {
21756     struct nk_text_edit_row r;
21757     int n = edit->string.len;
21758     float base_y = 0, prev_x;
21759     int i=0, k;
21760
```

```
21761     r.x0 = r.x1 = 0;
21762     r.ymin = r.ymax = 0;
21763     r.num_chars = 0;
21764
21765     /* search rows to find one that straddles 'y' */
21766     while (i < n) {
21767         nk_textedit_layout_row(&r, edit, i, row_height, font);
21768         if (r.num_chars <= 0)
21769             return n;
21770
21771         if (i==0 && y < base_y + r.ymin)
21772             return 0;
21773
21774         if (y < base_y + r.ymax)
21775             break;
21776
21777         i += r.num_chars;
21778         base_y += r.baseline_y_delta;
21779     }
21780
21781     /* below all text, return 'after' last character */
21782     if (i >= n)
21783         return n;
21784
21785     /* check if it's before the beginning of the line */
21786     if (x < r.x0)
21787         return i;
21788
21789     /* check if it's before the end of the line */
21790     if (x < r.x1) {
21791         /* search characters in row for one that straddles 'x' */
21792         k = i;
21793         prev_x = r.x0;
21794         for (i=0; i < r.num_chars; ++i) {
21795             float w = nk_textedit_get_width(edit, k, i, font);
21796             if (x < prev_x+w) {
21797                 if (x < prev_x+w/2)
21798                     return k+i;
21799                 else return k+i+1;
21800             }
21801             prev_x += w;
21802         }
21803         /* shouldn't happen, but if it does, fall through to end-of-line case */
21804     }
21805
21806     /* if the last character is a newline, return that.
21807      * otherwise return 'after' the last character */
21808     if (nk_str_rune_at(&edit->string, i+r.num_chars-1) == '\n')
21809         return i+r.num_chars-1;
21810     else return i+r.num_chars;
21811 }
21812 NK_LIB void
21813 nk_textedit_click(struct nk_text_edit *state, float x, float y,
21814     const struct nk_user_font *font, float row_height)
21815 {
21816     /* API click: on mouse down, move the cursor to the clicked location,
21817      * and reset the selection */
21818     state->cursor = nk_textedit_locate_coord(state, x, y, font, row_height);
21819     state->select_start = state->cursor;
21820     state->select_end = state->cursor;
21821     state->has_preferred_x = 0;
21822 }
21823 NK_LIB void
21824 nk_textedit_drag(struct nk_text_edit *state, float x, float y,
21825     const struct nk_user_font *font, float row_height)
21826 {
21827     /* API drag: on mouse drag, move the cursor and selection endpoint
21828      * to the clicked location */
21829     int p = nk_textedit_locate_coord(state, x, y, font, row_height);
21830     if (state->select_start == state->select_end)
21831         state->select_start = state->cursor;
21832     state->cursor = state->select_end = p;
21833 }
21834 NK_INTERN void
21835 nk_textedit_find_charpos(struct nk_text_find *find, struct nk_text_edit *state,
21836     int n, int single_line, const struct nk_user_font *font, float row_height)
21837 {
21838     /* find the x/y location of a character, and remember info about the previous
21839      * row in case we get a move-up event (for page up, we'll have to rescan) */
21840     struct nk_text_edit_row r;
21841     int prev_start = 0;
21842     int z = state->string.len;
21843     int i=0, first;
21844
21845     nk_zero_struct(r);
21846     if (n == z) {
21847         /* if it's at the end, then find the last line -- simpler than trying to
```

```
21848         explicitly handle this case in the regular code */
21849         nk_textedit_layout_row(&r, state, 0, row_height, font);
21850         if (single_line) {
21851             find->first_char = 0;
21852             find->length = z;
21853         } else {
21854             while (i < z) {
21855                 prev_start = i;
21856                 i += r.num_chars;
21857                 nk_textedit_layout_row(&r, state, i, row_height, font);
21858             }
21859
21860             find->first_char = i;
21861             find->length = r.num_chars;
21862         }
21863         find->x = r.x1;
21864         find->y = r.ymin;
21865         find->height = r.ymax - r.ymin;
21866         find->prev_first = prev_start;
21867         return;
21868     }
21869
21870     /* search rows to find the one that straddles character n */
21871     find->y = 0;
21872
21873     for(;;) {
21874         nk_textedit_layout_row(&r, state, i, row_height, font);
21875         if (n < i + r.num_chars) break;
21876         prev_start = i;
21877         i += r.num_chars;
21878         find->y += r.baseline_y_delta;
21879     }
21880
21881     find->first_char = first = i;
21882     find->length = r.num_chars;
21883     find->height = r.ymax - r.ymin;
21884     find->prev_first = prev_start;
21885
21886     /* now scan to find xpos */
21887     find->x = r.x0;
21888     for (i=0; first+i < n; ++i)
21889         find->x += nk_textedit_get_width(state, first, i, font);
21890 }
21891 NK_INTERN void
21892 nk_textedit_clamp(struct nk_text_edit *state)
21893 {
21894     /* make the selection/cursor state valid if client altered the string */
21895     int n = state->string.len;
21896     if (NK_TEXT_HAS_SELECTION(state)) {
21897         if (state->select_start > n) state->select_start = n;
21898         if (state->select_end   > n) state->select_end = n;
21899         /* if clamping forced them to be equal, move the cursor to match */
21900         if (state->select_start == state->select_end)
21901             state->cursor = state->select_start;
21902     }
21903     if (state->cursor > n) state->cursor = n;
21904 }
21905 NK_API void
21906 nk_textedit_delete(struct nk_text_edit *state, int where, int len)
21907 {
21908     /* delete characters while updating undo */
21909     nk_textedit_makeundo_delete(state, where, len);
21910     nk_str_delete_runes(&state->string, where, len);
21911     state->has_preferred_x = 0;
21912 }
21913 NK_API void
21914 nk_textedit_delete_selection(struct nk_text_edit *state)
21915 {
21916     /* delete the section */
21917     nk_textedit_clamp(state);
21918     if (NK_TEXT_HAS_SELECTION(state)) {
21919         if (state->select_start < state->select_end) {
21920             nk_textedit_delete(state, state->select_start,
21921                 state->select_end - state->select_start);
21922             state->select_end = state->cursor = state->select_start;
21923         } else {
21924             nk_textedit_delete(state, state->select_end,
21925                 state->select_start - state->select_end);
21926             state->select_start = state->cursor = state->select_end;
21927         }
21928         state->has_preferred_x = 0;
21929     }
21930 }
21931 NK_INTERN void
21932 nk_textedit_sortselection(struct nk_text_edit *state)
21933 {
21934     /* canonicalize the selection so start <= end */
```

```
21935     if (state->select_end < state->select_start) {
21936         int temp = state->select_end;
21937         state->select_end = state->select_start;
21938         state->select_start = temp;
21939     }
21940 }
21941 NK_INTERN void
21942 nk_textedit_move_to_first(struct nk_text_edit *state)
21943 {
21944     /* move cursor to first character of selection */
21945     if (NK_TEXT_HAS_SELECTION(state)) {
21946         nk_textedit_sortselection(state);
21947         state->cursor = state->select_start;
21948         state->select_end = state->select_start;
21949         state->has_preferred_x = 0;
21950     }
21951 }
21952 NK_INTERN void
21953 nk_textedit_move_to_last(struct nk_text_edit *state)
21954 {
21955     /* move cursor to last character of selection */
21956     if (NK_TEXT_HAS_SELECTION(state)) {
21957         nk_textedit_sortselection(state);
21958         nk_textedit_clamp(state);
21959         state->cursor = state->select_end;
21960         state->select_start = state->select_end;
21961         state->has_preferred_x = 0;
21962     }
21963 }
21964 NK_INTERN int
21965 nk_is_word_boundary( struct nk_text_edit *state, int idx)
21966 {
21967     int len;
21968     nk_rune c;
21969     if (idx <= 0) return 1;
21970     if (!nk_str_at_rune(&state->string, idx, &c, &len)) return 1;
21971     return (c == ' ' || c == '\t' ||c == 0x3000 || c == ',' || c == ';' ||
21972             c == '(' || c == ')' || c == '{' || c == '}' || c == '[' || c == ']' ||
21973             c == '|');
21974 }
21975 NK_INTERN int
21976 nk_textedit_move_to_word_previous(struct nk_text_edit *state)
21977 {
21978    int c = state->cursor - 1;
21979    while( c >= 0 && !nk_is_word_boundary(state, c))
21980       --c;
21981
21982    if( c < 0 )
21983       c = 0;
21984
21985    return c;
21986 }
21987 NK_INTERN int
21988 nk_textedit_move_to_word_next(struct nk_text_edit *state)
21989 {
21990    const int len = state->string.len;
21991    int c = state->cursor+1;
21992    while( c < len && !nk_is_word_boundary(state, c))
21993       ++c;
21994
21995    if( c > len )
21996       c = len;
21997
21998    return c;
21999 }
22000 NK_INTERN void
22001 nk_textedit_prep_selection_at_cursor(struct nk_text_edit *state)
22002 {
22003     /* update selection and cursor to match each other */
22004     if (!NK_TEXT_HAS_SELECTION(state))
22005         state->select_start = state->select_end = state->cursor;
22006     else state->cursor = state->select_end;
22007 }
22008 NK_API int
22009 nk_textedit_cut(struct nk_text_edit *state)
22010 {
22011     /* API cut: delete selection */
22012     if (state->mode == NK_TEXT_EDIT_MODE_VIEW)
22013         return 0;
22014     if (NK_TEXT_HAS_SELECTION(state)) {
22015         nk_textedit_delete_selection(state); /* implicitly clamps */
22016         state->has_preferred_x = 0;
22017         return 1;
22018     }
22019    return 0;
22020 }
22021 NK_API int
```

```
22022 nk_textedit_paste(struct nk_text_edit *state, char const *ctext, int len)
22023 {
22024     /* API paste: replace existing selection with passed-in text */
22025     int glyphs;
22026     const char *text = (const char *) ctext;
22027     if (state->mode == NK_TEXT_EDIT_MODE_VIEW) return 0;
22028
22029     /* if there's a selection, the paste should delete it */
22030     nk_textedit_clamp(state);
22031     nk_textedit_delete_selection(state);
22032
22033     /* try to insert the characters */
22034     glyphs = nk_utf_len(ctext, len);
22035     if (nk_str_insert_text_char(&state->string, state->cursor, text, len)) {
22036         nk_textedit_makeundo_insert(state, state->cursor, glyphs);
22037         state->cursor += len;
22038         state->has_preferred_x = 0;
22039         return 1;
22040     }
22041     /* remove the undo since we didn't actually insert the characters */
22042     if (state->undo.undo_point)
22043         --state->undo.undo_point;
22044     return 0;
22045 }
22046 NK_API void
22047 nk_textedit_text(struct nk_text_edit *state, const char *text, int total_len)
22048 {
22049     nk_rune unicode;
22050     int glyph_len;
22051     int text_len = 0;
22052
22053     NK_ASSERT(state);
22054     NK_ASSERT(text);
22055     if (!text || !total_len || state->mode == NK_TEXT_EDIT_MODE_VIEW) return;
22056
22057     glyph_len = nk_utf_decode(text, &unicode, total_len);
22058     while ((text_len < total_len) && glyph_len)
22059     {
22060         /* don't insert a backward delete, just process the event */
22061         if (unicode == 127) goto next;
22062         /* can't add newline in single-line mode */
22063         if (unicode == '\n' && state->single_line) goto next;
22064         /* filter incoming text */
22065         if (state->filter && !state->filter(state, unicode)) goto next;
22066
22067         if (!NK_TEXT_HAS_SELECTION(state) &&
22068             state->cursor < state->string.len)
22069         {
22070             if (state->mode == NK_TEXT_EDIT_MODE_REPLACE) {
22071                 nk_textedit_makeundo_replace(state, state->cursor, 1, 1);
22072                 nk_str_delete_runes(&state->string, state->cursor, 1);
22073             }
22074             if (nk_str_insert_text_utf8(&state->string, state->cursor,
22075                                         text+text_len, 1))
22076             {
22077                 ++state->cursor;
22078                 state->has_preferred_x = 0;
22079             }
22080         } else {
22081             nk_textedit_delete_selection(state); /* implicitly clamps */
22082             if (nk_str_insert_text_utf8(&state->string, state->cursor,
22083                                         text+text_len, 1))
22084             {
22085                 nk_textedit_makeundo_insert(state, state->cursor, 1);
22086                 ++state->cursor;
22087                 state->has_preferred_x = 0;
22088             }
22089         }
22090         next:
22091         text_len += glyph_len;
22092         glyph_len = nk_utf_decode(text + text_len, &unicode, total_len-text_len);
22093     }
22094 }
22095 NK_LIB void
22096 nk_textedit_key(struct nk_text_edit *state, enum nk_keys key, int shift_mod,
22097     const struct nk_user_font *font, float row_height)
22098 {
22099 retry:
22100     switch (key)
22101     {
22102     case NK_KEY_NONE:
22103     case NK_KEY_CTRL:
22104     case NK_KEY_ENTER:
22105     case NK_KEY_SHIFT:
22106     case NK_KEY_TAB:
22107     case NK_KEY_COPY:
22108     case NK_KEY_CUT:
```

```
22109     case NK_KEY_PASTE:
22110     case NK_KEY_MAX:
22111     default: break;
22112     case NK_KEY_TEXT_UNDO:
22113          nk_textedit_undo(state);
22114          state->has_preferred_x = 0;
22115          break;
22116
22117     case NK_KEY_TEXT_REDO:
22118         nk_textedit_redo(state);
22119         state->has_preferred_x = 0;
22120         break;
22121
22122     case NK_KEY_TEXT_SELECT_ALL:
22123         nk_textedit_select_all(state);
22124         state->has_preferred_x = 0;
22125         break;
22126
22127     case NK_KEY_TEXT_INSERT_MODE:
22128         if (state->mode == NK_TEXT_EDIT_MODE_VIEW)
22129             state->mode = NK_TEXT_EDIT_MODE_INSERT;
22130         break;
22131     case NK_KEY_TEXT_REPLACE_MODE:
22132         if (state->mode == NK_TEXT_EDIT_MODE_VIEW)
22133             state->mode = NK_TEXT_EDIT_MODE_REPLACE;
22134         break;
22135     case NK_KEY_TEXT_RESET_MODE:
22136         if (state->mode == NK_TEXT_EDIT_MODE_INSERT ||
22137             state->mode == NK_TEXT_EDIT_MODE_REPLACE)
22138             state->mode = NK_TEXT_EDIT_MODE_VIEW;
22139         break;
22140
22141     case NK_KEY_LEFT:
22142         if (shift_mod) {
22143             nk_textedit_clamp(state);
22144             nk_textedit_prep_selection_at_cursor(state);
22145             /* move selection left */
22146             if (state->select_end > 0)
22147                 --state->select_end;
22148             state->cursor = state->select_end;
22149             state->has_preferred_x = 0;
22150         } else {
22151             /* if currently there's a selection,
22152              * move cursor to start of selection */
22153             if (NK_TEXT_HAS_SELECTION(state))
22154                 nk_textedit_move_to_first(state);
22155             else if (state->cursor > 0)
22156                --state->cursor;
22157             state->has_preferred_x = 0;
22158         } break;
22159
22160     case NK_KEY_RIGHT:
22161         if (shift_mod) {
22162             nk_textedit_prep_selection_at_cursor(state);
22163             /* move selection right */
22164             ++state->select_end;
22165             nk_textedit_clamp(state);
22166             state->cursor = state->select_end;
22167             state->has_preferred_x = 0;
22168         } else {
22169             /* if currently there's a selection,
22170              * move cursor to end of selection */
22171             if (NK_TEXT_HAS_SELECTION(state))
22172                 nk_textedit_move_to_last(state);
22173             else ++state->cursor;
22174             nk_textedit_clamp(state);
22175             state->has_preferred_x = 0;
22176         } break;
22177
22178     case NK_KEY_TEXT_WORD_LEFT:
22179         if (shift_mod) {
22180             if( !NK_TEXT_HAS_SELECTION( state ) )
22181             nk_textedit_prep_selection_at_cursor(state);
22182             state->cursor = nk_textedit_move_to_word_previous(state);
22183             state->select_end = state->cursor;
22184             nk_textedit_clamp(state );
22185         } else {
22186             if (NK_TEXT_HAS_SELECTION(state))
22187                 nk_textedit_move_to_first(state);
22188             else {
22189                 state->cursor = nk_textedit_move_to_word_previous(state);
22190                 nk_textedit_clamp(state );
22191             }
22192         } break;
22193
22194     case NK_KEY_TEXT_WORD_RIGHT:
22195         if (shift_mod) {
```

```
22196             if( !NK_TEXT_HAS_SELECTION( state ) )
22197                 nk_textedit_prep_selection_at_cursor(state);
22198             state->cursor = nk_textedit_move_to_word_next(state);
22199             state->select_end = state->cursor;
22200             nk_textedit_clamp(state);
22201         } else {
22202             if (NK_TEXT_HAS_SELECTION(state))
22203                 nk_textedit_move_to_last(state);
22204             else {
22205                 state->cursor = nk_textedit_move_to_word_next(state);
22206                 nk_textedit_clamp(state );
22207             }
22208         } break;
22209
22210     case NK_KEY_DOWN: {
22211         struct nk_text_find find;
22212         struct nk_text_edit_row row;
22213         int i, sel = shift_mod;
22214
22215         if (state->single_line) {
22216             /* on windows, up&down in single-line behave like left&right */
22217             key = NK_KEY_RIGHT;
22218             goto retry;
22219         }
22220
22221         if (sel)
22222             nk_textedit_prep_selection_at_cursor(state);
22223         else if (NK_TEXT_HAS_SELECTION(state))
22224             nk_textedit_move_to_last(state);
22225
22226         /* compute current position of cursor point */
22227         nk_textedit_clamp(state);
22228         nk_textedit_find_charpos(&find, state, state->cursor, state->single_line,
22229             font, row_height);
22230
22231         /* now find character position down a row */
22232         if (find.length)
22233         {
22234             float x;
22235             float goal_x = state->has_preferred_x ? state->preferred_x : find.x;
22236             int start = find.first_char + find.length;
22237
22238             state->cursor = start;
22239             nk_textedit_layout_row(&row, state, state->cursor, row_height, font);
22240             x = row.x0;
22241
22242             for (i=0; i < row.num_chars && x < row.x1; ++i) {
22243                 float dx = nk_textedit_get_width(state, start, i, font);
22244                 x += dx;
22245                 if (x > goal_x)
22246                     break;
22247                 ++state->cursor;
22248             }
22249             nk_textedit_clamp(state);
22250
22251             state->has_preferred_x = 1;
22252             state->preferred_x = goal_x;
22253             if (sel)
22254                 state->select_end = state->cursor;
22255         }
22256     } break;
22257
22258     case NK_KEY_UP: {
22259         struct nk_text_find find;
22260         struct nk_text_edit_row row;
22261         int i, sel = shift_mod;
22262
22263         if (state->single_line) {
22264             /* on windows, up&down become left&right */
22265             key = NK_KEY_LEFT;
22266             goto retry;
22267         }
22268
22269         if (sel)
22270             nk_textedit_prep_selection_at_cursor(state);
22271         else if (NK_TEXT_HAS_SELECTION(state))
22272             nk_textedit_move_to_first(state);
22273
22274          /* compute current position of cursor point */
22275          nk_textedit_clamp(state);
22276          nk_textedit_find_charpos(&find, state, state->cursor, state->single_line,
22277                 font, row_height);
22278
22279          /* can only go up if there's a previous row */
22280          if (find.prev_first != find.first_char) {
22281             /* now find character position up a row */
22282             float x;
```

```
22283             float goal_x = state->has_preferred_x ? state->preferred_x : find.x;
22284
22285             state->cursor = find.prev_first;
22286             nk_textedit_layout_row(&row, state, state->cursor, row_height, font);
22287             x = row.x0;
22288
22289             for (i=0; i < row.num_chars && x < row.x1; ++i) {
22290                 float dx = nk_textedit_get_width(state, find.prev_first, i, font);
22291                 x += dx;
22292                 if (x > goal_x)
22293                     break;
22294                 ++state->cursor;
22295             }
22296             nk_textedit_clamp(state);
22297
22298             state->has_preferred_x = 1;
22299             state->preferred_x = goal_x;
22300             if (sel) state->select_end = state->cursor;
22301         }
22302       } break;
22303
22304     case NK_KEY_DEL:
22305         if (state->mode == NK_TEXT_EDIT_MODE_VIEW)
22306             break;
22307         if (NK_TEXT_HAS_SELECTION(state))
22308             nk_textedit_delete_selection(state);
22309         else {
22310             int n = state->string.len;
22311             if (state->cursor < n)
22312                 nk_textedit_delete(state, state->cursor, 1);
22313         }
22314         state->has_preferred_x = 0;
22315         break;
22316
22317     case NK_KEY_BACKSPACE:
22318         if (state->mode == NK_TEXT_EDIT_MODE_VIEW)
22319             break;
22320         if (NK_TEXT_HAS_SELECTION(state))
22321             nk_textedit_delete_selection(state);
22322         else {
22323             nk_textedit_clamp(state);
22324             if (state->cursor > 0) {
22325                 nk_textedit_delete(state, state->cursor-1, 1);
22326                 --state->cursor;
22327             }
22328         }
22329         state->has_preferred_x = 0;
22330         break;
22331
22332     case NK_KEY_TEXT_START:
22333         if (shift_mod) {
22334             nk_textedit_prep_selection_at_cursor(state);
22335             state->cursor = state->select_end = 0;
22336             state->has_preferred_x = 0;
22337         } else {
22338             state->cursor = state->select_start = state->select_end = 0;
22339             state->has_preferred_x = 0;
22340         }
22341         break;
22342
22343     case NK_KEY_TEXT_END:
22344         if (shift_mod) {
22345             nk_textedit_prep_selection_at_cursor(state);
22346             state->cursor = state->select_end = state->string.len;
22347             state->has_preferred_x = 0;
22348         } else {
22349             state->cursor = state->string.len;
22350             state->select_start = state->select_end = 0;
22351             state->has_preferred_x = 0;
22352         }
22353         break;
22354
22355     case NK_KEY_TEXT_LINE_START: {
22356         if (shift_mod) {
22357             struct nk_text_find find;
22358           nk_textedit_clamp(state);
22359             nk_textedit_prep_selection_at_cursor(state);
22360             if (state->string.len && state->cursor == state->string.len)
22361                 --state->cursor;
22362             nk_textedit_find_charpos(&find, state,state->cursor, state->single_line,
22363                 font, row_height);
22364             state->cursor = state->select_end = find.first_char;
22365             state->has_preferred_x = 0;
22366         } else {
22367             struct nk_text_find find;
22368             if (state->string.len && state->cursor == state->string.len)
22369                 --state->cursor;
```

```
22370             nk_textedit_clamp(state);
22371             nk_textedit_move_to_first(state);
22372             nk_textedit_find_charpos(&find, state, state->cursor, state->single_line,
22373                 font, row_height);
22374             state->cursor = find.first_char;
22375             state->has_preferred_x = 0;
22376         }
22377       } break;
22378
22379     case NK_KEY_TEXT_LINE_END: {
22380         if (shift_mod) {
22381             struct nk_text_find find;
22382             nk_textedit_clamp(state);
22383             nk_textedit_prep_selection_at_cursor(state);
22384             nk_textedit_find_charpos(&find, state, state->cursor, state->single_line,
22385                 font, row_height);
22386             state->has_preferred_x = 0;
22387             state->cursor = find.first_char + find.length;
22388             if (find.length > 0 && nk_str_rune_at(&state->string, state->cursor-1) == '\n')
22389                 --state->cursor;
22390             state->select_end = state->cursor;
22391         } else {
22392             struct nk_text_find find;
22393             nk_textedit_clamp(state);
22394             nk_textedit_move_to_first(state);
22395             nk_textedit_find_charpos(&find, state, state->cursor, state->single_line,
22396                 font, row_height);
22397
22398             state->has_preferred_x = 0;
22399             state->cursor = find.first_char + find.length;
22400             if (find.length > 0 && nk_str_rune_at(&state->string, state->cursor-1) == '\n')
22401                 --state->cursor;
22402         }} break;
22403     }
22404 }
22405 NK_INTERN void
22406 nk_textedit_flush_redo(struct nk_text_undo_state *state)
22407 {
22408     state->redo_point = NK_TEXTEDIT_UNDOSTATECOUNT;
22409     state->redo_char_point = NK_TEXTEDIT_UNDOCHARCOUNT;
22410 }
22411 NK_INTERN void
22412 nk_textedit_discard_undo(struct nk_text_undo_state *state)
22413 {
22414     /* discard the oldest entry in the undo list */
22415     if (state->undo_point > 0) {
22416         /* if the 0th undo state has characters, clean those up */
22417         if (state->undo_rec[0].char_storage >= 0) {
22418             int n = state->undo_rec[0].insert_length, i;
22419             /* delete n characters from all other records */
22420             state->undo_char_point = (short)(state->undo_char_point - n);
22421             NK_MEMCPY(state->undo_char, state->undo_char + n,
22422                 (nk_size)state->undo_char_point*sizeof(nk_rune));
22423             for (i=0; i < state->undo_point; ++i) {
22424                 if (state->undo_rec[i].char_storage >= 0)
22425                 state->undo_rec[i].char_storage = (short)
22426                     (state->undo_rec[i].char_storage - n);
22427             }
22428         }
22429         --state->undo_point;
22430         NK_MEMCPY(state->undo_rec, state->undo_rec+1,
22431             (nk_size)((nk_size)state->undo_point * sizeof(state->undo_rec[0])));
22432     }
22433 }
22434 NK_INTERN void
22435 nk_textedit_discard_redo(struct nk_text_undo_state *state)
22436 {
22437 /*  discard the oldest entry in the redo list--it's bad if this
22438     ever happens, but because undo & redo have to store the actual
22439     characters in different cases, the redo character buffer can
22440     fill up even though the undo buffer didn't */
22441     nk_size num;
22442     int k = NK_TEXTEDIT_UNDOSTATECOUNT-1;
22443     if (state->redo_point <= k) {
22444         /* if the k'th undo state has characters, clean those up */
22445         if (state->undo_rec[k].char_storage >= 0) {
22446             int n = state->undo_rec[k].insert_length, i;
22447             /* delete n characters from all other records */
22448             state->redo_char_point = (short)(state->redo_char_point + n);
22449             num = (nk_size)(NK_TEXTEDIT_UNDOCHARCOUNT - state->redo_char_point);
22450             NK_MEMCPY(state->undo_char + state->redo_char_point,
22451                 state->undo_char + state->redo_char_point-n, num * sizeof(char));
22452             for (i = state->redo_point; i < k; ++i) {
22453                 if (state->undo_rec[i].char_storage >= 0) {
22454                     state->undo_rec[i].char_storage = (short)
22455                         (state->undo_rec[i].char_storage + n);
22456                 }
```

```
22457             }
22458         }
22459         ++state->redo_point;
22460         num = (nk_size)(NK_TEXTEDIT_UNDOSTATECOUNT - state->redo_point);
22461         if (num) NK_MEMCPY(state->undo_rec + state->redo_point-1,
22462             state->undo_rec + state->redo_point, num * sizeof(state->undo_rec[0]));
22463     }
22464 }
22465 NK_INTERN struct nk_text_undo_record*
22466 nk_textedit_create_undo_record(struct nk_text_undo_state *state, int numchars)
22467 {
22468     /* any time we create a new undo record, we discard redo*/
22469     nk_textedit_flush_redo(state);
22470
22471     /* if we have no free records, we have to make room,
22472      * by sliding the existing records down */
22473     if (state->undo_point == NK_TEXTEDIT_UNDOSTATECOUNT)
22474         nk_textedit_discard_undo(state);
22475
22476     /* if the characters to store won't possibly fit in the buffer,
22477      * we can't undo */
22478     if (numchars > NK_TEXTEDIT_UNDOCHARCOUNT) {
22479         state->undo_point = 0;
22480         state->undo_char_point = 0;
22481         return 0;
22482     }
22483
22484     /* if we don't have enough free characters in the buffer,
22485      * we have to make room */
22486     while (state->undo_char_point + numchars > NK_TEXTEDIT_UNDOCHARCOUNT)
22487         nk_textedit_discard_undo(state);
22488     return &state->undo_rec[state->undo_point++];
22489 }
22490 NK_INTERN nk_rune*
22491 nk_textedit_createundo(struct nk_text_undo_state *state, int pos,
22492     int insert_len, int delete_len)
22493 {
22494     struct nk_text_undo_record *r = nk_textedit_create_undo_record(state, insert_len);
22495     if (r == 0)
22496         return 0;
22497
22498     r->where = pos;
22499     r->insert_length = (short) insert_len;
22500     r->delete_length = (short) delete_len;
22501
22502     if (insert_len == 0) {
22503         r->char_storage = -1;
22504         return 0;
22505     } else {
22506         r->char_storage = state->undo_char_point;
22507         state->undo_char_point = (short)(state->undo_char_point +  insert_len);
22508         return &state->undo_char[r->char_storage];
22509     }
22510 }
22511 NK_API void
22512 nk_textedit_undo(struct nk_text_edit *state)
22513 {
22514     struct nk_text_undo_state *s = &state->undo;
22515     struct nk_text_undo_record u, *r;
22516     if (s->undo_point == 0)
22517         return;
22518
22519     /* we need to do two things: apply the undo record, and create a redo record */
22520     u = s->undo_rec[s->undo_point-1];
22521     r = &s->undo_rec[s->redo_point-1];
22522     r->char_storage = -1;
22523
22524     r->insert_length = u.delete_length;
22525     r->delete_length = u.insert_length;
22526     r->where = u.where;
22527
22528     if (u.delete_length)
22529     {
22530        /*   if the undo record says to delete characters, then the redo record will
22531             need to re-insert the characters that get deleted, so we need to store
22532             them.
22533             there are three cases:
22534                 - there's enough room to store the characters
22535                 - characters stored for *redoing* don't leave room for redo
22536                 - characters stored for *undoing* don't leave room for redo
22537             if the last is true, we have to bail */
22538         if (s->undo_char_point + u.delete_length >= NK_TEXTEDIT_UNDOCHARCOUNT) {
22539             /* the undo records take up too much character space; there's no space
22540             * to store the redo characters */
22541             r->insert_length = 0;
22542         } else {
22543             int i;
```

```
22544             /* there's definitely room to store the characters eventually */
22545             while (s->undo_char_point + u.delete_length > s->redo_char_point) {
22546                 /* there's currently not enough room, so discard a redo record */
22547                 nk_textedit_discard_redo(s);
22548                 /* should never happen: */
22549                 if (s->redo_point == NK_TEXTEDIT_UNDOSTATECOUNT)
22550                     return;
22551             }
22552
22553             r = &s->undo_rec[s->redo_point-1];
22554             r->char_storage = (short)(s->redo_char_point - u.delete_length);
22555             s->redo_char_point = (short)(s->redo_char_point -  u.delete_length);
22556
22557             /* now save the characters */
22558             for (i=0; i < u.delete_length; ++i)
22559                 s->undo_char[r->char_storage + i] =
22560                     nk_str_rune_at(&state->string, u.where + i);
22561         }
22562         /* now we can carry out the deletion */
22563         nk_str_delete_runes(&state->string, u.where, u.delete_length);
22564     }
22565
22566     /* check type of recorded action: */
22567     if (u.insert_length) {
22568         /* easy case: was a deletion, so we need to insert n characters */
22569         nk_str_insert_text_runes(&state->string, u.where,
22570             &s->undo_char[u.char_storage], u.insert_length);
22571         s->undo_char_point = (short)(s->undo_char_point - u.insert_length);
22572     }
22573     state->cursor = (short)(u.where + u.insert_length);
22574
22575     s->undo_point--;
22576     s->redo_point--;
22577 }
22578 NK_API void
22579 nk_textedit_redo(struct nk_text_edit *state)
22580 {
22581     struct nk_text_undo_state *s = &state->undo;
22582     struct nk_text_undo_record *u, r;
22583     if (s->redo_point == NK_TEXTEDIT_UNDOSTATECOUNT)
22584         return;
22585
22586     /* we need to do two things: apply the redo record, and create an undo record */
22587     u = &s->undo_rec[s->undo_point];
22588     r = s->undo_rec[s->redo_point];
22589
22590     /* we KNOW there must be room for the undo record, because the redo record
22591     was derived from an undo record */
22592     u->delete_length = r.insert_length;
22593     u->insert_length = r.delete_length;
22594     u->where = r.where;
22595     u->char_storage = -1;
22596
22597     if (r.delete_length) {
22598         /* the redo record requires us to delete characters, so the undo record
22599         needs to store the characters */
22600         if (s->undo_char_point + u->insert_length > s->redo_char_point) {
22601             u->insert_length = 0;
22602             u->delete_length = 0;
22603         } else {
22604             int i;
22605             u->char_storage = s->undo_char_point;
22606             s->undo_char_point = (short)(s->undo_char_point + u->insert_length);
22607
22608             /* now save the characters */
22609             for (i=0; i < u->insert_length; ++i) {
22610                 s->undo_char[u->char_storage + i] =
22611                     nk_str_rune_at(&state->string, u->where + i);
22612             }
22613         }
22614         nk_str_delete_runes(&state->string, r.where, r.delete_length);
22615     }
22616
22617     if (r.insert_length) {
22618         /* easy case: need to insert n characters */
22619         nk_str_insert_text_runes(&state->string, r.where,
22620             &s->undo_char[r.char_storage], r.insert_length);
22621     }
22622     state->cursor = r.where + r.insert_length;
22623
22624     s->undo_point++;
22625     s->redo_point++;
22626 }
22627 NK_INTERN void
22628 nk_textedit_makeundo_insert(struct nk_text_edit *state, int where, int length)
22629 {
22630     nk_textedit_createundo(&state->undo, where, 0, length);
```

```
22631 }
22632 NK_INTERN void
22633 nk_textedit_makeundo_delete(struct nk_text_edit *state, int where, int length)
22634 {
22635     int i;
22636     nk_rune *p = nk_textedit_createundo(&state->undo, where, length, 0);
22637     if (p) {
22638         for (i=0; i < length; ++i)
22639             p[i] = nk_str_rune_at(&state->string, where+i);
22640     }
22641 }
22642 NK_INTERN void
22643 nk_textedit_makeundo_replace(struct nk_text_edit *state, int where,
22644     int old_length, int new_length)
22645 {
22646     int i;
22647     nk_rune *p = nk_textedit_createundo(&state->undo, where, old_length, new_length);
22648     if (p) {
22649         for (i=0; i < old_length; ++i)
22650             p[i] = nk_str_rune_at(&state->string, where+i);
22651     }
22652 }
22653 NK_LIB void
22654 nk_textedit_clear_state(struct nk_text_edit *state, enum nk_text_edit_type type,
22655     nk_plugin_filter filter)
22656 {
22657     /* reset the state to default */
22658    state->undo.undo_point = 0;
22659    state->undo.undo_char_point = 0;
22660    state->undo.redo_point = NK_TEXTEDIT_UNDOSTATECOUNT;
22661    state->undo.redo_char_point = NK_TEXTEDIT_UNDOCHARCOUNT;
22662    state->select_end = state->select_start = 0;
22663    state->cursor = 0;
22664    state->has_preferred_x = 0;
22665    state->preferred_x = 0;
22666    state->cursor_at_end_of_line = 0;
22667    state->initialized = 1;
22668    state->single_line = (unsigned char)(type == NK_TEXT_EDIT_SINGLE_LINE);
22669    state->mode = NK_TEXT_EDIT_MODE_VIEW;
22670    state->filter = filter;
22671    state->scrollbar = nk_vec2(0,0);
22672 }
22673 NK_API void
22674 nk_textedit_init_fixed(struct nk_text_edit *state, void *memory, nk_size size)
22675 {
22676     NK_ASSERT(state);
22677     NK_ASSERT(memory);
22678     if (!state || !memory || !size) return;
22679     NK_MEMSET(state, 0, sizeof(struct nk_text_edit));
22680     nk_textedit_clear_state(state, NK_TEXT_EDIT_SINGLE_LINE, 0);
22681     nk_str_init_fixed(&state->string, memory, size);
22682 }
22683 NK_API void
22684 nk_textedit_init(struct nk_text_edit *state, struct nk_allocator *alloc, nk_size size)
22685 {
22686     NK_ASSERT(state);
22687     NK_ASSERT(alloc);
22688     if (!state || !alloc) return;
22689     NK_MEMSET(state, 0, sizeof(struct nk_text_edit));
22690     nk_textedit_clear_state(state, NK_TEXT_EDIT_SINGLE_LINE, 0);
22691     nk_str_init(&state->string, alloc, size);
22692 }
22693 #ifdef NK_INCLUDE_DEFAULT_ALLOCATOR
22694 NK_API void
22695 nk_textedit_init_default(struct nk_text_edit *state)
22696 {
22697     NK_ASSERT(state);
22698     if (!state) return;
22699     NK_MEMSET(state, 0, sizeof(struct nk_text_edit));
22700     nk_textedit_clear_state(state, NK_TEXT_EDIT_SINGLE_LINE, 0);
22701     nk_str_init_default(&state->string);
22702 }
22703 #endif
22704 NK_API void
22705 nk_textedit_select_all(struct nk_text_edit *state)
22706 {
22707     NK_ASSERT(state);
22708     state->select_start = 0;
22709     state->select_end = state->string.len;
22710 }
22711 NK_API void
22712 nk_textedit_free(struct nk_text_edit *state)
22713 {
22714     NK_ASSERT(state);
22715     if (!state) return;
22716     nk_str_free(&state->string);
22717 }
```

```
22718
22719
22720
22721
22722
22723 /* ===============================================================
22724  *
22725  *                          FILTER
22726  *
22727  * ===============================================================*/
22728 NK_API int
22729 nk_filter_default(const struct nk_text_edit *box, nk_rune unicode)
22730 {
22731     NK_UNUSED(unicode);
22732     NK_UNUSED(box);
22733     return nk_true;
22734 }
22735 NK_API int
22736 nk_filter_ascii(const struct nk_text_edit *box, nk_rune unicode)
22737 {
22738     NK_UNUSED(box);
22739     if (unicode > 128) return nk_false;
22740     else return nk_true;
22741 }
22742 NK_API int
22743 nk_filter_float(const struct nk_text_edit *box, nk_rune unicode)
22744 {
22745     NK_UNUSED(box);
22746     if ((unicode < '0' || unicode > '9') && unicode != '.' && unicode != '-')
22747         return nk_false;
22748     else return nk_true;
22749 }
22750 NK_API int
22751 nk_filter_decimal(const struct nk_text_edit *box, nk_rune unicode)
22752 {
22753     NK_UNUSED(box);
22754     if ((unicode < '0' || unicode > '9') && unicode != '-')
22755         return nk_false;
22756     else return nk_true;
22757 }
22758 NK_API int
22759 nk_filter_hex(const struct nk_text_edit *box, nk_rune unicode)
22760 {
22761     NK_UNUSED(box);
22762     if ((unicode < '0' || unicode > '9') &&
22763         (unicode < 'a' || unicode > 'f') &&
22764         (unicode < 'A' || unicode > 'F'))
22765         return nk_false;
22766     else return nk_true;
22767 }
22768 NK_API int
22769 nk_filter_oct(const struct nk_text_edit *box, nk_rune unicode)
22770 {
22771     NK_UNUSED(box);
22772     if (unicode < '0' || unicode > '7')
22773         return nk_false;
22774     else return nk_true;
22775 }
22776 NK_API int
22777 nk_filter_binary(const struct nk_text_edit *box, nk_rune unicode)
22778 {
22779     NK_UNUSED(box);
22780     if (unicode != '0' && unicode != '1')
22781         return nk_false;
22782     else return nk_true;
22783 }
22784
22785 /* ===============================================================
22786  *
22787  *                          EDIT
22788  *
22789  * ===============================================================*/
22790 NK_LIB void
22791 nk_edit_draw_text(struct nk_command_buffer *out,
22792     const struct nk_style_edit *style, float pos_x, float pos_y,
22793     float x_offset, const char *text, int byte_len, float row_height,
22794     const struct nk_user_font *font, struct nk_color background,
22795     struct nk_color foreground, int is_selected)
22796 {
22797     NK_ASSERT(out);
22798     NK_ASSERT(font);
22799     NK_ASSERT(style);
22800     if (!text || !byte_len || !out || !style) return;
22801
22802     {int glyph_len = 0;
22803     nk_rune unicode = 0;
22804     int text_len = 0;
```

```
22805     float line_width = 0;
22806     float glyph_width;
22807     const char *line = text;
22808     float line_offset = 0;
22809     int line_count = 0;
22810
22811     struct nk_text txt;
22812     txt.padding = nk_vec2(0,0);
22813     txt.background = background;
22814     txt.text = foreground;
22815
22816     glyph_len = nk_utf_decode(text+text_len, &unicode, byte_len-text_len);
22817     if (!glyph_len) return;
22818     while ((text_len < byte_len) && glyph_len)
22819     {
22820         if (unicode == '\n') {
22821             /* new line separator so draw previous line */
22822             struct nk_rect label;
22823             label.y = pos_y + line_offset;
22824             label.h = row_height;
22825             label.w = line_width;
22826             label.x = pos_x;
22827             if (!line_count)
22828                 label.x += x_offset;
22829
22830             if (is_selected) /* selection needs to draw different background color */
22831                 nk_fill_rect(out, label, 0, background);
22832             nk_widget_text(out, label, line, (int)((text + text_len) - line),
22833                 &txt, NK_TEXT_CENTERED, font);
22834
22835             text_len++;
22836             line_count++;
22837             line_width = 0;
22838             line = text + text_len;
22839             line_offset += row_height;
22840             glyph_len = nk_utf_decode(text + text_len, &unicode, (int)(byte_len-text_len));
22841             continue;
22842         }
22843         if (unicode == '\r') {
22844             text_len++;
22845             glyph_len = nk_utf_decode(text + text_len, &unicode, byte_len-text_len);
22846             continue;
22847         }
22848         glyph_width = font->width(font->userdata, font->height, text+text_len, glyph_len);
22849         line_width += (float)glyph_width;
22850         text_len += glyph_len;
22851         glyph_len = nk_utf_decode(text + text_len, &unicode, byte_len-text_len);
22852         continue;
22853     }
22854     if (line_width > 0) {
22855         /* draw last line */
22856         struct nk_rect label;
22857         label.y = pos_y + line_offset;
22858         label.h = row_height;
22859         label.w = line_width;
22860         label.x = pos_x;
22861         if (!line_count)
22862             label.x += x_offset;
22863
22864         if (is_selected)
22865             nk_fill_rect(out, label, 0, background);
22866         nk_widget_text(out, label, line, (int)((text + text_len) - line),
22867             &txt, NK_TEXT_LEFT, font);
22868     }}
22869 }
22870 NK_LIB nk_flags
22871 nk_do_edit(nk_flags *state, struct nk_command_buffer *out,
22872     struct nk_rect bounds, nk_flags flags, nk_plugin_filter filter,
22873     struct nk_text_edit *edit, const struct nk_style_edit *style,
22874     struct nk_input *in, const struct nk_user_font *font)
22875 {
22876     struct nk_rect area;
22877     nk_flags ret = 0;
22878     float row_height;
22879     char prev_state = 0;
22880     char is_hovered = 0;
22881     char select_all = 0;
22882     char cursor_follow = 0;
22883     struct nk_rect old_clip;
22884     struct nk_rect clip;
22885
22886     NK_ASSERT(state);
22887     NK_ASSERT(out);
22888     NK_ASSERT(style);
22889     if (!state || !out || !style)
22890         return ret;
22891
```

```
22892     /* visible text area calculation */
22893     area.x = bounds.x + style->padding.x + style->border;
22894     area.y = bounds.y + style->padding.y + style->border;
22895     area.w = bounds.w - (2.0f * style->padding.x + 2 * style->border);
22896     area.h = bounds.h - (2.0f * style->padding.y + 2 * style->border);
22897     if (flags & NK_EDIT_MULTILINE)
22898         area.w = NK_MAX(0, area.w - style->scrollbar_size.x);
22899     row_height = (flags & NK_EDIT_MULTILINE)? font->height + style->row_padding: area.h;
22900
22901     /* calculate clipping rectangle */
22902     old_clip = out->clip;
22903     nk_unify(&clip, &old_clip, area.x, area.y, area.x + area.w, area.y + area.h);
22904
22905     /* update edit state */
22906     prev_state = (char)edit->active;
22907     is_hovered = (char)nk_input_is_mouse_hovering_rect(in, bounds);
22908     if (in && in->mouse.buttons[NK_BUTTON_LEFT].clicked && in->mouse.buttons[NK_BUTTON_LEFT].down) {
22909         edit->active = NK_INBOX(in->mouse.pos.x, in->mouse.pos.y,
22910                                 bounds.x, bounds.y, bounds.w, bounds.h);
22911     }
22912
22913     /* (de)activate text editor */
22914     if (!prev_state && edit->active) {
22915         const enum nk_text_edit_type type = (flags & NK_EDIT_MULTILINE) ?
22916             NK_TEXT_EDIT_MULTI_LINE: NK_TEXT_EDIT_SINGLE_LINE;
22917         nk_textedit_clear_state(edit, type, filter);
22918         if (flags & NK_EDIT_AUTO_SELECT)
22919             select_all = nk_true;
22920         if (flags & NK_EDIT_GOTO_END_ON_ACTIVATE) {
22921             edit->cursor = edit->string.len;
22922             in = 0;
22923         }
22924     } else if (!edit->active) edit->mode = NK_TEXT_EDIT_MODE_VIEW;
22925     if (flags & NK_EDIT_READ_ONLY)
22926         edit->mode = NK_TEXT_EDIT_MODE_VIEW;
22927     else if (flags & NK_EDIT_ALWAYS_INSERT_MODE)
22928         edit->mode = NK_TEXT_EDIT_MODE_INSERT;
22929
22930     ret = (edit->active) ? NK_EDIT_ACTIVE: NK_EDIT_INACTIVE;
22931     if (prev_state != edit->active)
22932         ret |= (edit->active) ? NK_EDIT_ACTIVATED: NK_EDIT_DEACTIVATED;
22933
22934     /* handle user input */
22935     if (edit->active && in)
22936     {
22937         int shift_mod = in->keyboard.keys[NK_KEY_SHIFT].down;
22938         const float mouse_x = (in->mouse.pos.x - area.x) + edit->scrollbar.x;
22939         const float mouse_y = (in->mouse.pos.y - area.y) + edit->scrollbar.y;
22940
22941         /* mouse click handler */
22942         is_hovered = (char)nk_input_is_mouse_hovering_rect(in, area);
22943         if (select_all) {
22944             nk_textedit_select_all(edit);
22945         } else if (is_hovered && in->mouse.buttons[NK_BUTTON_LEFT].down &&
22946             in->mouse.buttons[NK_BUTTON_LEFT].clicked) {
22947             nk_textedit_click(edit, mouse_x, mouse_y, font, row_height);
22948         } else if (is_hovered && in->mouse.buttons[NK_BUTTON_LEFT].down &&
22949             (in->mouse.delta.x != 0.0f || in->mouse.delta.y != 0.0f)) {
22950             nk_textedit_drag(edit, mouse_x, mouse_y, font, row_height);
22951             cursor_follow = nk_true;
22952         } else if (is_hovered && in->mouse.buttons[NK_BUTTON_RIGHT].clicked &&
22953             in->mouse.buttons[NK_BUTTON_RIGHT].down) {
22954             nk_textedit_key(edit, NK_KEY_TEXT_WORD_LEFT, nk_false, font, row_height);
22955             nk_textedit_key(edit, NK_KEY_TEXT_WORD_RIGHT, nk_true, font, row_height);
22956             cursor_follow = nk_true;
22957         }
22958
22959         {int i; /* keyboard input */
22960         int old_mode = edit->mode;
22961         for (i = 0; i < NK_KEY_MAX; ++i) {
22962             if (i == NK_KEY_ENTER || i == NK_KEY_TAB) continue; /* special case */
22963             if (nk_input_is_key_pressed(in, (enum nk_keys)i)) {
22964                 nk_textedit_key(edit, (enum nk_keys)i, shift_mod, font, row_height);
22965                 cursor_follow = nk_true;
22966             }
22967         }
22968         if (old_mode != edit->mode) {
22969             in->keyboard.text_len = 0;
22970         }}
22971
22972         /* text input */
22973         edit->filter = filter;
22974         if (in->keyboard.text_len) {
22975             nk_textedit_text(edit, in->keyboard.text, in->keyboard.text_len);
22976             cursor_follow = nk_true;
22977             in->keyboard.text_len = 0;
22978         }
```

```
22979
22980         /* enter key handler */
22981         if (nk_input_is_key_pressed(in, NK_KEY_ENTER)) {
22982             cursor_follow = nk_true;
22983             if (flags & NK_EDIT_CTRL_ENTER_NEWLINE && shift_mod)
22984                 nk_textedit_text(edit, "\n", 1);
22985             else if (flags & NK_EDIT_SIG_ENTER)
22986                 ret |= NK_EDIT_COMMITED;
22987             else nk_textedit_text(edit, "\n", 1);
22988         }
22989
22990         /* cut & copy handler */
22991         {int copy= nk_input_is_key_pressed(in, NK_KEY_COPY);
22992         int cut = nk_input_is_key_pressed(in, NK_KEY_CUT);
22993         if ((copy || cut) && (flags & NK_EDIT_CLIPBOARD))
22994         {
22995             int glyph_len;
22996             nk_rune unicode;
22997             const char *text;
22998             int b = edit->select_start;
22999             int e = edit->select_end;
23000
23001             int begin = NK_MIN(b, e);
23002             int end = NK_MAX(b, e);
23003             text = nk_str_at_const(&edit->string, begin, &unicode, &glyph_len);
23004             if (edit->clip.copy)
23005                 edit->clip.copy(edit->clip.userdata, text, end - begin);
23006             if (cut && !(flags & NK_EDIT_READ_ONLY)){
23007                 nk_textedit_cut(edit);
23008                 cursor_follow = nk_true;
23009             }
23010         }}
23011
23012         /* paste handler */
23013         {int paste = nk_input_is_key_pressed(in, NK_KEY_PASTE);
23014         if (paste && (flags & NK_EDIT_CLIPBOARD) && edit->clip.paste) {
23015             edit->clip.paste(edit->clip.userdata, edit);
23016             cursor_follow = nk_true;
23017         }}
23018
23019         /* tab handler */
23020         {int tab = nk_input_is_key_pressed(in, NK_KEY_TAB);
23021         if (tab && (flags & NK_EDIT_ALLOW_TAB)) {
23022             nk_textedit_text(edit, "    ", 4);
23023             cursor_follow = nk_true;
23024         }}
23025     }
23026
23027     /* set widget state */
23028     if (edit->active)
23029         *state = NK_WIDGET_STATE_ACTIVE;
23030     else nk_widget_state_reset(state);
23031
23032     if (is_hovered)
23033         *state |= NK_WIDGET_STATE_HOVERED;
23034
23035     /* DRAW EDIT */
23036     {const char *text = nk_str_get_const(&edit->string);
23037     int len = nk_str_len_char(&edit->string);
23038
23039     {/* select background colors/images  */
23040     const struct nk_style_item *background;
23041     if (*state & NK_WIDGET_STATE_ACTIVED)
23042         background = &style->active;
23043     else if (*state & NK_WIDGET_STATE_HOVER)
23044         background = &style->hover;
23045     else background = &style->normal;
23046
23047     /* draw background frame */
23048     if (background->type == NK_STYLE_ITEM_COLOR) {
23049         nk_stroke_rect(out, bounds, style->rounding, style->border, style->border_color);
23050         nk_fill_rect(out, bounds, style->rounding, background->data.color);
23051     } else nk_draw_image(out, bounds, &background->data.image, nk_white);}
23052
23053     area.w = NK_MAX(0, area.w - style->cursor_size);
23054     if (edit->active)
23055     {
23056         int total_lines = 1;
23057         struct nk_vec2 text_size = nk_vec2(0,0);
23058
23059         /* text pointer positions */
23060         const char *cursor_ptr = 0;
23061         const char *select_begin_ptr = 0;
23062         const char *select_end_ptr = 0;
23063
23064         /* 2D pixel positions */
23065         struct nk_vec2 cursor_pos = nk_vec2(0,0);
```

```
23066         struct nk_vec2 selection_offset_start = nk_vec2(0,0);
23067         struct nk_vec2 selection_offset_end = nk_vec2(0,0);
23068
23069         int selection_begin = NK_MIN(edit->select_start, edit->select_end);
23070         int selection_end = NK_MAX(edit->select_start, edit->select_end);
23071
23072         /* calculate total line count + total space + cursor/selection position */
23073         float line_width = 0.0f;
23074         if (text && len)
23075         {
23076             /* utf8 encoding */
23077             float glyph_width;
23078             int glyph_len = 0;
23079             nk_rune unicode = 0;
23080             int text_len = 0;
23081             int glyphs = 0;
23082             int row_begin = 0;
23083
23084             glyph_len = nk_utf_decode(text, &unicode, len);
23085             glyph_width = font->width(font->userdata, font->height, text, glyph_len);
23086             line_width = 0;
23087
23088             /* iterate all lines */
23089             while ((text_len < len) && glyph_len)
23090             {
23091                 /* set cursor 2D position and line */
23092                 if (!cursor_ptr && glyphs == edit->cursor)
23093                 {
23094                     int glyph_offset;
23095                     struct nk_vec2 out_offset;
23096                     struct nk_vec2 row_size;
23097                     const char *remaining;
23098
23099                     /* calculate 2d position */
23100                     cursor_pos.y = (float)(total_lines-1) * row_height;
23101                     row_size = nk_text_calculate_text_bounds(font, text+row_begin,
23102                                 text_len-row_begin, row_height, &remaining,
23103                                 &out_offset, &glyph_offset, NK_STOP_ON_NEW_LINE);
23104                     cursor_pos.x = row_size.x;
23105                     cursor_ptr = text + text_len;
23106                 }
23107
23108                 /* set start selection 2D position and line */
23109                 if (!select_begin_ptr && edit->select_start != edit->select_end &&
23110                     glyphs == selection_begin)
23111                 {
23112                     int glyph_offset;
23113                     struct nk_vec2 out_offset;
23114                     struct nk_vec2 row_size;
23115                     const char *remaining;
23116
23117                     /* calculate 2d position */
23118                     selection_offset_start.y = (float)(NK_MAX(total_lines-1,0)) * row_height;
23119                     row_size = nk_text_calculate_text_bounds(font, text+row_begin,
23120                                 text_len-row_begin, row_height, &remaining,
23121                                 &out_offset, &glyph_offset, NK_STOP_ON_NEW_LINE);
23122                     selection_offset_start.x = row_size.x;
23123                     select_begin_ptr = text + text_len;
23124                 }
23125
23126                 /* set end selection 2D position and line */
23127                 if (!select_end_ptr && edit->select_start != edit->select_end &&
23128                     glyphs == selection_end)
23129                 {
23130                     int glyph_offset;
23131                     struct nk_vec2 out_offset;
23132                     struct nk_vec2 row_size;
23133                     const char *remaining;
23134
23135                     /* calculate 2d position */
23136                     selection_offset_end.y = (float)(total_lines-1) * row_height;
23137                     row_size = nk_text_calculate_text_bounds(font, text+row_begin,
23138                                 text_len-row_begin, row_height, &remaining,
23139                                 &out_offset, &glyph_offset, NK_STOP_ON_NEW_LINE);
23140                     selection_offset_end.x = row_size.x;
23141                     select_end_ptr = text + text_len;
23142                 }
23143                 if (unicode == '\n') {
23144                     text_size.x = NK_MAX(text_size.x, line_width);
23145                     total_lines++;
23146                     line_width = 0;
23147                     text_len++;
23148                     glyphs++;
23149                     row_begin = text_len;
23150                     glyph_len = nk_utf_decode(text + text_len, &unicode, len-text_len);
23151                     glyph_width = font->width(font->userdata, font->height, text+text_len, glyph_len);
23152                     continue;
```

```
23153                 }
23154
23155                 glyphs++;
23156                 text_len += glyph_len;
23157                 line_width += (float)glyph_width;
23158
23159                 glyph_len = nk_utf_decode(text + text_len, &unicode, len-text_len);
23160                 glyph_width = font->width(font->userdata, font->height,
23161                     text+text_len, glyph_len);
23162                 continue;
23163             }
23164             text_size.y = (float)total_lines * row_height;
23165
23166             /* handle case when cursor is at end of text buffer */
23167             if (!cursor_ptr && edit->cursor == edit->string.len) {
23168                 cursor_pos.x = line_width;
23169                 cursor_pos.y = text_size.y - row_height;
23170             }
23171         }
23172         {
23173             /* scrollbar */
23174             if (cursor_follow)
23175             {
23176                 /* update scrollbar to follow cursor */
23177                 if (!(flags & NK_EDIT_NO_HORIZONTAL_SCROLL)) {
23178                     /* horizontal scroll */
23179                     const float scroll_increment = area.w * 0.25f;
23180                     if (cursor_pos.x < edit->scrollbar.x)
23181                         edit->scrollbar.x = (float)(int)NK_MAX(0.0f, cursor_pos.x - scroll_increment);
23182                     if (cursor_pos.x >= edit->scrollbar.x + area.w)
23183                         edit->scrollbar.x = (float)(int)NK_MAX(0.0f, edit->scrollbar.x +
    scroll_increment);
23184                 } else edit->scrollbar.x = 0;
23185
23186                 if (flags & NK_EDIT_MULTILINE) {
23187                     /* vertical scroll */
23188                     if (cursor_pos.y < edit->scrollbar.y)
23189                         edit->scrollbar.y = NK_MAX(0.0f, cursor_pos.y - row_height);
23190                     if (cursor_pos.y >= edit->scrollbar.y + area.h)
23191                         edit->scrollbar.y = edit->scrollbar.y + row_height;
23192                 } else edit->scrollbar.y = 0;
23193             }
23194
23195             /* scrollbar widget */
23196             if (flags & NK_EDIT_MULTILINE)
23197             {
23198                 nk_flags ws;
23199                 struct nk_rect scroll;
23200                 float scroll_target;
23201                 float scroll_offset;
23202                 float scroll_step;
23203                 float scroll_inc;
23204
23205                 scroll = area;
23206                 scroll.x = (bounds.x + bounds.w - style->border) - style->scrollbar_size.x;
23207                 scroll.w = style->scrollbar_size.x;
23208
23209                 scroll_offset = edit->scrollbar.y;
23210                 scroll_step = scroll.h * 0.10f;
23211                 scroll_inc = scroll.h * 0.01f;
23212                 scroll_target = text_size.y;
23213                 edit->scrollbar.y = nk_do_scrollbarv(&ws, out, scroll, 0,
23214                         scroll_offset, scroll_target, scroll_step, scroll_inc,
23215                         &style->scrollbar, in, font);
23216             }
23217         }
23218
23219         /* draw text */
23220         {struct nk_color background_color;
23221         struct nk_color text_color;
23222         struct nk_color sel_background_color;
23223         struct nk_color sel_text_color;
23224         struct nk_color cursor_color;
23225         struct nk_color cursor_text_color;
23226         const struct nk_style_item *background;
23227         nk_push_scissor(out, clip);
23228
23229         /* select correct colors to draw */
23230         if (*state & NK_WIDGET_STATE_ACTIVED) {
23231             background = &style->active;
23232             text_color = style->text_active;
23233             sel_text_color = style->selected_text_hover;
23234             sel_background_color = style->selected_hover;
23235             cursor_color = style->cursor_hover;
23236             cursor_text_color = style->cursor_hover;
23237         } else if (*state & NK_WIDGET_STATE_HOVER) {
23238             background = &style->hover;
```

```
23239             text_color = style->text_hover;
23240             sel_text_color = style->selected_text_hover;
23241             sel_background_color = style->selected_hover;
23242             cursor_text_color = style->cursor_text_hover;
23243             cursor_color = style->cursor_hover;
23244         } else {
23245             background = &style->normal;
23246             text_color = style->text_normal;
23247             sel_text_color = style->selected_text_normal;
23248             sel_background_color = style->selected_normal;
23249             cursor_color = style->cursor_normal;
23250             cursor_text_color = style->cursor_text_normal;
23251         }
23252         if (background->type == NK_STYLE_ITEM_IMAGE)
23253             background_color = nk_rgba(0,0,0,0);
23254         else background_color = background->data.color;
23255
23256
23257         if (edit->select_start == edit->select_end) {
23258             /* no selection so just draw the complete text */
23259             const char *begin = nk_str_get_const(&edit->string);
23260             int l = nk_str_len_char(&edit->string);
23261             nk_edit_draw_text(out, style, area.x - edit->scrollbar.x,
23262                 area.y - edit->scrollbar.y, 0, begin, l, row_height, font,
23263                 background_color, text_color, nk_false);
23264         } else {
23265             /* edit has selection so draw 1-3 text chunks */
23266             if (edit->select_start != edit->select_end && selection_begin > 0){
23267                 /* draw unselected text before selection */
23268                 const char *begin = nk_str_get_const(&edit->string);
23269                 NK_ASSERT(select_begin_ptr);
23270                 nk_edit_draw_text(out, style, area.x - edit->scrollbar.x,
23271                     area.y - edit->scrollbar.y, 0, begin, (int)(select_begin_ptr - begin),
23272                     row_height, font, background_color, text_color, nk_false);
23273             }
23274             if (edit->select_start != edit->select_end) {
23275                 /* draw selected text */
23276                 NK_ASSERT(select_begin_ptr);
23277                 if (!select_end_ptr) {
23278                     const char *begin = nk_str_get_const(&edit->string);
23279                     select_end_ptr = begin + nk_str_len_char(&edit->string);
23280                 }
23281                 nk_edit_draw_text(out, style,
23282                     area.x - edit->scrollbar.x,
23283                     area.y + selection_offset_start.y - edit->scrollbar.y,
23284                     selection_offset_start.x,
23285                     select_begin_ptr, (int)(select_end_ptr - select_begin_ptr),
23286                     row_height, font, sel_background_color, sel_text_color, nk_true);
23287             }
23288             if ((edit->select_start != edit->select_end &&
23289                 selection_end < edit->string.len))
23290             {
23291                 /* draw unselected text after selected text */
23292                 const char *begin = select_end_ptr;
23293                 const char *end = nk_str_get_const(&edit->string) +
23294                                   nk_str_len_char(&edit->string);
23295                 NK_ASSERT(select_end_ptr);
23296                 nk_edit_draw_text(out, style,
23297                     area.x - edit->scrollbar.x,
23298                     area.y + selection_offset_end.y - edit->scrollbar.y,
23299                     selection_offset_end.x,
23300                     begin, (int)(end - begin), row_height, font,
23301                     background_color, text_color, nk_true);
23302             }
23303         }
23304
23305         /* cursor */
23306         if (edit->select_start == edit->select_end)
23307         {
23308             if (edit->cursor >= nk_str_len(&edit->string) ||
23309                 (cursor_ptr && *cursor_ptr == '\n')) {
23310                 /* draw cursor at end of line */
23311                 struct nk_rect cursor;
23312                 cursor.w = style->cursor_size;
23313                 cursor.h = font->height;
23314                 cursor.x = area.x + cursor_pos.x - edit->scrollbar.x;
23315                 cursor.y = area.y + cursor_pos.y + row_height/2.0f - cursor.h/2.0f;
23316                 cursor.y -= edit->scrollbar.y;
23317                 nk_fill_rect(out, cursor, 0, cursor_color);
23318             } else {
23319                 /* draw cursor inside text */
23320                 int glyph_len;
23321                 struct nk_rect label;
23322                 struct nk_text txt;
23323
23324                 nk_rune unicode;
23325                 NK_ASSERT(cursor_ptr);
```

```
23326                 glyph_len = nk_utf_decode(cursor_ptr, &unicode, 4);
23327
23328                 label.x = area.x + cursor_pos.x - edit->scrollbar.x;
23329                 label.y = area.y + cursor_pos.y - edit->scrollbar.y;
23330                 label.w = font->width(font->userdata, font->height, cursor_ptr, glyph_len);
23331                 label.h = row_height;
23332
23333                 txt.padding = nk_vec2(0,0);
23334                 txt.background = cursor_color;;
23335                 txt.text = cursor_text_color;
23336                 nk_fill_rect(out, label, 0, cursor_color);
23337                 nk_widget_text(out, label, cursor_ptr, glyph_len, &txt, NK_TEXT_LEFT, font);
23338             }
23339         }}
23340     } else {
23341         /* not active so just draw text */
23342         int l = nk_str_len_char(&edit->string);
23343         const char *begin = nk_str_get_const(&edit->string);
23344
23345         const struct nk_style_item *background;
23346         struct nk_color background_color;
23347         struct nk_color text_color;
23348         nk_push_scissor(out, clip);
23349         if (*state & NK_WIDGET_STATE_ACTIVED) {
23350             background = &style->active;
23351             text_color = style->text_active;
23352         } else if (*state & NK_WIDGET_STATE_HOVER) {
23353             background = &style->hover;
23354             text_color = style->text_hover;
23355         } else {
23356             background = &style->normal;
23357             text_color = style->text_normal;
23358         }
23359         if (background->type == NK_STYLE_ITEM_IMAGE)
23360             background_color = nk_rgba(0,0,0,0);
23361         else background_color = background->data.color;
23362         nk_edit_draw_text(out, style, area.x - edit->scrollbar.x,
23363             area.y - edit->scrollbar.y, 0, begin, l, row_height, font,
23364             background_color, text_color, nk_false);
23365     }
23366     nk_push_scissor(out, old_clip);}
23367     return ret;
23368 }
23369 NK_API void
23370 nk_edit_focus(struct nk_context *ctx, nk_flags flags)
23371 {
23372     nk_hash hash;
23373     struct nk_window *win;
23374
23375     NK_ASSERT(ctx);
23376     NK_ASSERT(ctx->current);
23377     if (!ctx || !ctx->current) return;
23378
23379     win = ctx->current;
23380     hash = win->edit.seq;
23381     win->edit.active = nk_true;
23382     win->edit.name = hash;
23383     if (flags & NK_EDIT_ALWAYS_INSERT_MODE)
23384         win->edit.mode = NK_TEXT_EDIT_MODE_INSERT;
23385 }
23386 NK_API void
23387 nk_edit_unfocus(struct nk_context *ctx)
23388 {
23389     struct nk_window *win;
23390     NK_ASSERT(ctx);
23391     NK_ASSERT(ctx->current);
23392     if (!ctx || !ctx->current) return;
23393
23394     win = ctx->current;
23395     win->edit.active = nk_false;
23396     win->edit.name = 0;
23397 }
23398 NK_API nk_flags
23399 nk_edit_string(struct nk_context *ctx, nk_flags flags,
23400     char *memory, int *len, int max, nk_plugin_filter filter)
23401 {
23402     nk_hash hash;
23403     nk_flags state;
23404     struct nk_text_edit *edit;
23405     struct nk_window *win;
23406
23407     NK_ASSERT(ctx);
23408     NK_ASSERT(memory);
23409     NK_ASSERT(len);
23410     if (!ctx || !memory || !len)
23411         return 0;
23412
```

```
23413     filter = (!filter) ? nk_filter_default: filter;
23414     win = ctx->current;
23415     hash = win->edit.seq;
23416     edit = &ctx->text_edit;
23417     nk_textedit_clear_state(&ctx->text_edit, (flags & NK_EDIT_MULTILINE)?
23418         NK_TEXT_EDIT_MULTI_LINE: NK_TEXT_EDIT_SINGLE_LINE, filter);
23419
23420     if (win->edit.active && hash == win->edit.name) {
23421         if (flags & NK_EDIT_NO_CURSOR)
23422             edit->cursor = nk_utf_len(memory, *len);
23423         else edit->cursor = win->edit.cursor;
23424         if (!(flags & NK_EDIT_SELECTABLE)) {
23425             edit->select_start = win->edit.cursor;
23426             edit->select_end = win->edit.cursor;
23427         } else {
23428             edit->select_start = win->edit.sel_start;
23429             edit->select_end = win->edit.sel_end;
23430         }
23431         edit->mode = win->edit.mode;
23432         edit->scrollbar.x = (float)win->edit.scrollbar.x;
23433         edit->scrollbar.y = (float)win->edit.scrollbar.y;
23434         edit->active = nk_true;
23435     } else edit->active = nk_false;
23436
23437     max = NK_MAX(1, max);
23438     *len = NK_MIN(*len, max-1);
23439     nk_str_init_fixed(&edit->string, memory, (nk_size)max);
23440     edit->string.buffer.allocated = (nk_size)*len;
23441     edit->string.len = nk_utf_len(memory, *len);
23442     state = nk_edit_buffer(ctx, flags, edit, filter);
23443     *len = (int)edit->string.buffer.allocated;
23444
23445     if (edit->active) {
23446         win->edit.cursor = edit->cursor;
23447         win->edit.sel_start = edit->select_start;
23448         win->edit.sel_end = edit->select_end;
23449         win->edit.mode = edit->mode;
23450         win->edit.scrollbar.x = (nk_uint)edit->scrollbar.x;
23451         win->edit.scrollbar.y = (nk_uint)edit->scrollbar.y;
23452     } return state;
23453 }
23454 NK_API nk_flags
23455 nk_edit_buffer(struct nk_context *ctx, nk_flags flags,
23456     struct nk_text_edit *edit, nk_plugin_filter filter)
23457 {
23458     struct nk_window *win;
23459     struct nk_style *style;
23460     struct nk_input *in;
23461
23462     enum nk_widget_layout_states state;
23463     struct nk_rect bounds;
23464
23465     nk_flags ret_flags = 0;
23466     unsigned char prev_state;
23467     nk_hash hash;
23468
23469     /* make sure correct values */
23470     NK_ASSERT(ctx);
23471     NK_ASSERT(edit);
23472     NK_ASSERT(ctx->current);
23473     NK_ASSERT(ctx->current->layout);
23474     if (!ctx || !ctx->current || !ctx->current->layout)
23475         return 0;
23476
23477     win = ctx->current;
23478     style = &ctx->style;
23479     state = nk_widget(&bounds, ctx);
23480     if (!state) return state;
23481     in = (win->layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
23482
23483     /* check if edit is currently hot item */
23484     hash = win->edit.seq++;
23485     if (win->edit.active && hash == win->edit.name) {
23486         if (flags & NK_EDIT_NO_CURSOR)
23487             edit->cursor = edit->string.len;
23488         if (!(flags & NK_EDIT_SELECTABLE)) {
23489             edit->select_start = edit->cursor;
23490             edit->select_end = edit->cursor;
23491         }
23492         if (flags & NK_EDIT_CLIPBOARD)
23493             edit->clip = ctx->clip;
23494         edit->active = (unsigned char)win->edit.active;
23495     } else edit->active = nk_false;
23496     edit->mode = win->edit.mode;
23497
23498     filter = (!filter) ? nk_filter_default: filter;
23499     prev_state = (unsigned char)edit->active;
```

```
23500     in = (flags & NK_EDIT_READ_ONLY) ? 0: in;
23501     ret_flags = nk_do_edit(&ctx->last_widget_state, &win->buffer, bounds, flags,
23502                     filter, edit, &style->edit, in, style->font);
23503
23504     if (ctx->last_widget_state & NK_WIDGET_STATE_HOVER)
23505         ctx->style.cursor_active = ctx->style.cursors[NK_CURSOR_TEXT];
23506     if (edit->active && prev_state != edit->active) {
23507         /* current edit is now hot */
23508         win->edit.active = nk_true;
23509         win->edit.name = hash;
23510     } else if (prev_state && !edit->active) {
23511         /* current edit is now cold */
23512         win->edit.active = nk_false;
23513     } return ret_flags;
23514 }
23515 NK_API nk_flags
23516 nk_edit_string_zero_terminated(struct nk_context *ctx, nk_flags flags,
23517     char *buffer, int max, nk_plugin_filter filter)
23518 {
23519     nk_flags result;
23520     int len = nk_strlen(buffer);
23521     result = nk_edit_string(ctx, flags, buffer, &len, max, filter);
23522     buffer[NK_MIN(NK_MAX(max-1,0), len)] = '\0';
23523     return result;
23524 }
23525
23526
23527
23528
23529
23530 /* ===============================================================
23531  *
23532  *                              PROPERTY
23533  *
23534  * ===============================================================*/
23535 NK_LIB void
23536 nk_drag_behavior(nk_flags *state, const struct nk_input *in,
23537     struct nk_rect drag, struct nk_property_variant *variant,
23538     float inc_per_pixel)
23539 {
23540     int left_mouse_down = in && in->mouse.buttons[NK_BUTTON_LEFT].down;
23541     int left_mouse_click_in_cursor = in &&
23542         nk_input_has_mouse_click_down_in_rect(in, NK_BUTTON_LEFT, drag, nk_true);
23543
23544     nk_widget_state_reset(state);
23545     if (nk_input_is_mouse_hovering_rect(in, drag))
23546         *state = NK_WIDGET_STATE_HOVERED;
23547
23548     if (left_mouse_down && left_mouse_click_in_cursor) {
23549         float delta, pixels;
23550         pixels = in->mouse.delta.x;
23551         delta = pixels * inc_per_pixel;
23552         switch (variant->kind) {
23553         default: break;
23554         case NK_PROPERTY_INT:
23555             variant->value.i = variant->value.i + (int)delta;
23556             variant->value.i = NK_CLAMP(variant->min_value.i, variant->value.i, variant->max_value.i);
23557             break;
23558         case NK_PROPERTY_FLOAT:
23559             variant->value.f = variant->value.f + (float)delta;
23560             variant->value.f = NK_CLAMP(variant->min_value.f, variant->value.f, variant->max_value.f);
23561             break;
23562         case NK_PROPERTY_DOUBLE:
23563             variant->value.d = variant->value.d + (double)delta;
23564             variant->value.d = NK_CLAMP(variant->min_value.d, variant->value.d, variant->max_value.d);
23565             break;
23566         }
23567         *state = NK_WIDGET_STATE_ACTIVE;
23568     }
23569     if (*state & NK_WIDGET_STATE_HOVER && !nk_input_is_mouse_prev_hovering_rect(in, drag))
23570         *state |= NK_WIDGET_STATE_ENTERED;
23571     else if (nk_input_is_mouse_prev_hovering_rect(in, drag))
23572         *state |= NK_WIDGET_STATE_LEFT;
23573 }
23574 NK_LIB void
23575 nk_property_behavior(nk_flags *ws, const struct nk_input *in,
23576     struct nk_rect property,  struct nk_rect label, struct nk_rect edit,
23577     struct nk_rect empty, int *state, struct nk_property_variant *variant,
23578     float inc_per_pixel)
23579 {
23580     if (in && *state == NK_PROPERTY_DEFAULT) {
23581         if (nk_button_behavior(ws, edit, in, NK_BUTTON_DEFAULT))
23582             *state = NK_PROPERTY_EDIT;
23583         else if (nk_input_is_mouse_click_down_in_rect(in, NK_BUTTON_LEFT, label, nk_true))
23584             *state = NK_PROPERTY_DRAG;
23585         else if (nk_input_is_mouse_click_down_in_rect(in, NK_BUTTON_LEFT, empty, nk_true))
23586             *state = NK_PROPERTY_DRAG;
```

```
23587     }
23588     if (*state == NK_PROPERTY_DRAG) {
23589         nk_drag_behavior(ws, in, property, variant, inc_per_pixel);
23590         if (!(*ws & NK_WIDGET_STATE_ACTIVED)) *state = NK_PROPERTY_DEFAULT;
23591     }
23592 }
23593 NK_LIB void
23594 nk_draw_property(struct nk_command_buffer *out, const struct nk_style_property *style,
23595     const struct nk_rect *bounds, const struct nk_rect *label, nk_flags state,
23596     const char *name, int len, const struct nk_user_font *font)
23597 {
23598     struct nk_text text;
23599     const struct nk_style_item *background;
23600
23601     /* select correct background and text color */
23602     if (state & NK_WIDGET_STATE_ACTIVED) {
23603         background = &style->active;
23604         text.text = style->label_active;
23605     } else if (state & NK_WIDGET_STATE_HOVER) {
23606         background = &style->hover;
23607         text.text = style->label_hover;
23608     } else {
23609         background = &style->normal;
23610         text.text = style->label_normal;
23611     }
23612
23613     /* draw background */
23614     if (background->type == NK_STYLE_ITEM_IMAGE) {
23615         nk_draw_image(out, *bounds, &background->data.image, nk_white);
23616         text.background = nk_rgba(0,0,0,0);
23617     } else {
23618         text.background = background->data.color;
23619         nk_fill_rect(out, *bounds, style->rounding, background->data.color);
23620         nk_stroke_rect(out, *bounds, style->rounding, style->border, background->data.color);
23621     }
23622
23623     /* draw label */
23624     text.padding = nk_vec2(0,0);
23625     nk_widget_text(out, *label, name, len, &text, NK_TEXT_CENTERED, font);
23626 }
23627 NK_LIB void
23628 nk_do_property(nk_flags *ws,
23629     struct nk_command_buffer *out, struct nk_rect property,
23630     const char *name, struct nk_property_variant *variant,
23631     float inc_per_pixel, char *buffer, int *len,
23632     int *state, int *cursor, int *select_begin, int *select_end,
23633     const struct nk_style_property *style,
23634     enum nk_property_filter filter, struct nk_input *in,
23635     const struct nk_user_font *font, struct nk_text_edit *text_edit,
23636     enum nk_button_behavior behavior)
23637 {
23638     const nk_plugin_filter filters[] = {
23639         nk_filter_decimal,
23640         nk_filter_float
23641     };
23642     int active, old;
23643     int num_len, name_len;
23644     char string[NK_MAX_NUMBER_BUFFER];
23645     float size;
23646
23647     char *dst = 0;
23648     int *length;
23649
23650     struct nk_rect left;
23651     struct nk_rect right;
23652     struct nk_rect label;
23653     struct nk_rect edit;
23654     struct nk_rect empty;
23655
23656     /* left decrement button */
23657     left.h = font->height/2;
23658     left.w = left.h;
23659     left.x = property.x + style->border + style->padding.x;
23660     left.y = property.y + style->border + property.h/2.0f - left.h/2;
23661
23662     /* text label */
23663     name_len = nk_strlen(name);
23664     size = font->width(font->userdata, font->height, name, name_len);
23665     label.x = left.x + left.w + style->padding.x;
23666     label.w = (float)size + 2 * style->padding.x;
23667     label.y = property.y + style->border + style->padding.y;
23668     label.h = property.h - (2 * style->border + 2 * style->padding.y);
23669
23670     /* right increment button */
23671     right.y = left.y;
23672     right.w = left.w;
23673     right.h = left.h;
```

```
23674     right.x = property.x + property.w - (right.w + style->padding.x);
23675
23676     /* edit */
23677     if (*state == NK_PROPERTY_EDIT) {
23678         size = font->width(font->userdata, font->height, buffer, *len);
23679         size += style->edit.cursor_size;
23680         length = len;
23681         dst = buffer;
23682     } else {
23683         switch (variant->kind) {
23684         default: break;
23685         case NK_PROPERTY_INT:
23686             nk_itoa(string, variant->value.i);
23687             num_len = nk_strlen(string);
23688             break;
23689         case NK_PROPERTY_FLOAT:
23690             NK_DTOA(string, (double)variant->value.f);
23691             num_len = nk_string_float_limit(string, NK_MAX_FLOAT_PRECISION);
23692             break;
23693         case NK_PROPERTY_DOUBLE:
23694             NK_DTOA(string, variant->value.d);
23695             num_len = nk_string_float_limit(string, NK_MAX_FLOAT_PRECISION);
23696             break;
23697         }
23698         size = font->width(font->userdata, font->height, string, num_len);
23699         dst = string;
23700         length = &num_len;
23701     }
23702
23703     edit.w =  (float)size + 2 * style->padding.x;
23704     edit.w = NK_MIN(edit.w, right.x - (label.x + label.w));
23705     edit.x = right.x - (edit.w + style->padding.x);
23706     edit.y = property.y + style->border;
23707     edit.h = property.h - (2 * style->border);
23708
23709     /* empty left space activator */
23710     empty.w = edit.x - (label.x + label.w);
23711     empty.x = label.x + label.w;
23712     empty.y = property.y;
23713     empty.h = property.h;
23714
23715     /* update property */
23716     old = (*state == NK_PROPERTY_EDIT);
23717     nk_property_behavior(ws, in, property, label, edit, empty, state, variant, inc_per_pixel);
23718
23719     /* draw property */
23720     if (style->draw_begin) style->draw_begin(out, style->userdata);
23721     nk_draw_property(out, style, &property, &label, *ws, name, name_len, font);
23722     if (style->draw_end) style->draw_end(out, style->userdata);
23723
23724     /* execute right button  */
23725     if (nk_do_button_symbol(ws, out, left, style->sym_left, behavior, &style->dec_button, in, font)) {
23726         switch (variant->kind) {
23727         default: break;
23728         case NK_PROPERTY_INT:
23729             variant->value.i = NK_CLAMP(variant->min_value.i, variant->value.i - variant->step.i,
    variant->max_value.i); break;
23730         case NK_PROPERTY_FLOAT:
23731             variant->value.f = NK_CLAMP(variant->min_value.f, variant->value.f - variant->step.f,
    variant->max_value.f); break;
23732         case NK_PROPERTY_DOUBLE:
23733             variant->value.d = NK_CLAMP(variant->min_value.d, variant->value.d - variant->step.d,
    variant->max_value.d); break;
23734         }
23735     }
23736     /* execute left button  */
23737     if (nk_do_button_symbol(ws, out, right, style->sym_right, behavior, &style->inc_button, in, font))
    {
23738         switch (variant->kind) {
23739         default: break;
23740         case NK_PROPERTY_INT:
23741             variant->value.i = NK_CLAMP(variant->min_value.i, variant->value.i + variant->step.i,
    variant->max_value.i); break;
23742         case NK_PROPERTY_FLOAT:
23743             variant->value.f = NK_CLAMP(variant->min_value.f, variant->value.f + variant->step.f,
    variant->max_value.f); break;
23744         case NK_PROPERTY_DOUBLE:
23745             variant->value.d = NK_CLAMP(variant->min_value.d, variant->value.d + variant->step.d,
    variant->max_value.d); break;
23746         }
23747     }
23748     if (old != NK_PROPERTY_EDIT && (*state == NK_PROPERTY_EDIT)) {
23749         /* property has been activated so setup buffer */
23750         NK_MEMCPY(buffer, dst, (nk_size)*length);
23751         *cursor = nk_utf_len(buffer, *length);
23752         *len = *length;
23753         length = len;
```

```
23754         dst = buffer;
23755         active = 0;
23756     } else active = (*state == NK_PROPERTY_EDIT);
23757
23758     /* execute and run text edit field */
23759     nk_textedit_clear_state(text_edit, NK_TEXT_EDIT_SINGLE_LINE, filters[filter]);
23760     text_edit->active = (unsigned char)active;
23761     text_edit->string.len = *length;
23762     text_edit->cursor = NK_CLAMP(0, *cursor, *length);
23763     text_edit->select_start = NK_CLAMP(0,*select_begin, *length);
23764     text_edit->select_end = NK_CLAMP(0,*select_end, *length);
23765     text_edit->string.buffer.allocated = (nk_size)*length;
23766     text_edit->string.buffer.memory.size = NK_MAX_NUMBER_BUFFER;
23767     text_edit->string.buffer.memory.ptr = dst;
23768     text_edit->string.buffer.size = NK_MAX_NUMBER_BUFFER;
23769     text_edit->mode = NK_TEXT_EDIT_MODE_INSERT;
23770     nk_do_edit(ws, out, edit, NK_EDIT_FIELD|NK_EDIT_AUTO_SELECT,
23771         filters[filter], text_edit, &style->edit, (*state == NK_PROPERTY_EDIT) ? in: 0, font);
23772
23773     *length = text_edit->string.len;
23774     *cursor = text_edit->cursor;
23775     *select_begin = text_edit->select_start;
23776     *select_end = text_edit->select_end;
23777     if (text_edit->active && nk_input_is_key_pressed(in, NK_KEY_ENTER))
23778         text_edit->active = nk_false;
23779
23780     if (active && !text_edit->active) {
23781         /* property is now not active so convert edit text to value*/
23782         *state = NK_PROPERTY_DEFAULT;
23783         buffer[*len] = '\0';
23784         switch (variant->kind) {
23785         default: break;
23786         case NK_PROPERTY_INT:
23787             variant->value.i = nk_strtoi(buffer, 0);
23788             variant->value.i = NK_CLAMP(variant->min_value.i, variant->value.i, variant->max_value.i);
23789             break;
23790         case NK_PROPERTY_FLOAT:
23791             nk_string_float_limit(buffer, NK_MAX_FLOAT_PRECISION);
23792             variant->value.f = nk_strtof(buffer, 0);
23793             variant->value.f = NK_CLAMP(variant->min_value.f, variant->value.f, variant->max_value.f);
23794             break;
23795         case NK_PROPERTY_DOUBLE:
23796             nk_string_float_limit(buffer, NK_MAX_FLOAT_PRECISION);
23797             variant->value.d = nk_strtod(buffer, 0);
23798             variant->value.d = NK_CLAMP(variant->min_value.d, variant->value.d, variant->max_value.d);
23799             break;
23800         }
23801     }
23802 }
23803 NK_LIB struct nk_property_variant
23804 nk_property_variant_int(int value, int min_value, int max_value, int step)
23805 {
23806     struct nk_property_variant result;
23807     result.kind = NK_PROPERTY_INT;
23808     result.value.i = value;
23809     result.min_value.i = min_value;
23810     result.max_value.i = max_value;
23811     result.step.i = step;
23812     return result;
23813 }
23814 NK_LIB struct nk_property_variant
23815 nk_property_variant_float(float value, float min_value, float max_value, float step)
23816 {
23817     struct nk_property_variant result;
23818     result.kind = NK_PROPERTY_FLOAT;
23819     result.value.f = value;
23820     result.min_value.f = min_value;
23821     result.max_value.f = max_value;
23822     result.step.f = step;
23823     return result;
23824 }
23825 NK_LIB struct nk_property_variant
23826 nk_property_variant_double(double value, double min_value, double max_value,
23827     double step)
23828 {
23829     struct nk_property_variant result;
23830     result.kind = NK_PROPERTY_DOUBLE;
23831     result.value.d = value;
23832     result.min_value.d = min_value;
23833     result.max_value.d = max_value;
23834     result.step.d = step;
23835     return result;
23836 }
23837 NK_LIB void
23838 nk_property(struct nk_context *ctx, const char *name, struct nk_property_variant *variant,
23839     float inc_per_pixel, const enum nk_property_filter filter)
23840 {
```

```
23841     struct nk_window *win;
23842     struct nk_panel *layout;
23843     struct nk_input *in;
23844     const struct nk_style *style;
23845
23846     struct nk_rect bounds;
23847     enum nk_widget_layout_states s;
23848
23849     int *state = 0;
23850     nk_hash hash = 0;
23851     char *buffer = 0;
23852     int *len = 0;
23853     int *cursor = 0;
23854     int *select_begin = 0;
23855     int *select_end = 0;
23856     int old_state;
23857
23858     char dummy_buffer[NK_MAX_NUMBER_BUFFER];
23859     int dummy_state = NK_PROPERTY_DEFAULT;
23860     int dummy_length = 0;
23861     int dummy_cursor = 0;
23862     int dummy_select_begin = 0;
23863     int dummy_select_end = 0;
23864
23865     NK_ASSERT(ctx);
23866     NK_ASSERT(ctx->current);
23867     NK_ASSERT(ctx->current->layout);
23868     if (!ctx || !ctx->current || !ctx->current->layout)
23869         return;
23870
23871     win = ctx->current;
23872     layout = win->layout;
23873     style = &ctx->style;
23874     s = nk_widget(&bounds, ctx);
23875     if (!s) return;
23876
23877     /* calculate hash from name */
23878     if (name[0] == '#') {
23879         hash = nk_murmur_hash(name, (int)nk_strlen(name), win->property.seq++);
23880         name++; /* special number hash */
23881     } else hash = nk_murmur_hash(name, (int)nk_strlen(name), 42);
23882
23883     /* check if property is currently hot item */
23884     if (win->property.active && hash == win->property.name) {
23885         buffer = win->property.buffer;
23886         len = &win->property.length;
23887         cursor = &win->property.cursor;
23888         state = &win->property.state;
23889         select_begin = &win->property.select_start;
23890         select_end = &win->property.select_end;
23891     } else {
23892         buffer = dummy_buffer;
23893         len = &dummy_length;
23894         cursor = &dummy_cursor;
23895         state = &dummy_state;
23896         select_begin =  &dummy_select_begin;
23897         select_end = &dummy_select_end;
23898     }
23899
23900     /* execute property widget */
23901     old_state = *state;
23902     ctx->text_edit.clip = ctx->clip;
23903     in = ((s == NK_WIDGET_ROM && !win->property.active) ||
23904         layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
23905     nk_do_property(&ctx->last_widget_state, &win->buffer, bounds, name,
23906         variant, inc_per_pixel, buffer, len, state, cursor, select_begin,
23907         select_end, &style->property, filter, in, style->font, &ctx->text_edit,
23908         ctx->button_behavior);
23909
23910     if (in && *state != NK_PROPERTY_DEFAULT && !win->property.active) {
23911         /* current property is now hot */
23912         win->property.active = 1;
23913         NK_MEMCPY(win->property.buffer, buffer, (nk_size)*len);
23914         win->property.length = *len;
23915         win->property.cursor = *cursor;
23916         win->property.state = *state;
23917         win->property.name = hash;
23918         win->property.select_start = *select_begin;
23919         win->property.select_end = *select_end;
23920         if (*state == NK_PROPERTY_DRAG) {
23921             ctx->input.mouse.grab = nk_true;
23922             ctx->input.mouse.grabbed = nk_true;
23923         }
23924     }
23925     /* check if previously active property is now inactive */
23926     if (*state == NK_PROPERTY_DEFAULT && old_state != NK_PROPERTY_DEFAULT) {
23927         if (old_state == NK_PROPERTY_DRAG) {
```

```
23928             ctx->input.mouse.grab = nk_false;
23929             ctx->input.mouse.grabbed = nk_false;
23930             ctx->input.mouse.ungrab = nk_true;
23931         }
23932         win->property.select_start = 0;
23933         win->property.select_end = 0;
23934         win->property.active = 0;
23935     }
23936 }
23937 NK_API void
23938 nk_property_int(struct nk_context *ctx, const char *name,
23939     int min, int *val, int max, int step, float inc_per_pixel)
23940 {
23941     struct nk_property_variant variant;
23942     NK_ASSERT(ctx);
23943     NK_ASSERT(name);
23944     NK_ASSERT(val);
23945
23946     if (!ctx || !ctx->current || !name || !val) return;
23947     variant = nk_property_variant_int(*val, min, max, step);
23948     nk_property(ctx, name, &variant, inc_per_pixel, NK_FILTER_INT);
23949     *val = variant.value.i;
23950 }
23951 NK_API void
23952 nk_property_float(struct nk_context *ctx, const char *name,
23953     float min, float *val, float max, float step, float inc_per_pixel)
23954 {
23955     struct nk_property_variant variant;
23956     NK_ASSERT(ctx);
23957     NK_ASSERT(name);
23958     NK_ASSERT(val);
23959
23960     if (!ctx || !ctx->current || !name || !val) return;
23961     variant = nk_property_variant_float(*val, min, max, step);
23962     nk_property(ctx, name, &variant, inc_per_pixel, NK_FILTER_FLOAT);
23963     *val = variant.value.f;
23964 }
23965 NK_API void
23966 nk_property_double(struct nk_context *ctx, const char *name,
23967     double min, double *val, double max, double step, float inc_per_pixel)
23968 {
23969     struct nk_property_variant variant;
23970     NK_ASSERT(ctx);
23971     NK_ASSERT(name);
23972     NK_ASSERT(val);
23973
23974     if (!ctx || !ctx->current || !name || !val) return;
23975     variant = nk_property_variant_double(*val, min, max, step);
23976     nk_property(ctx, name, &variant, inc_per_pixel, NK_FILTER_FLOAT);
23977     *val = variant.value.d;
23978 }
23979 NK_API int
23980 nk_propertyi(struct nk_context *ctx, const char *name, int min, int val,
23981     int max, int step, float inc_per_pixel)
23982 {
23983     struct nk_property_variant variant;
23984     NK_ASSERT(ctx);
23985     NK_ASSERT(name);
23986
23987     if (!ctx || !ctx->current || !name) return val;
23988     variant = nk_property_variant_int(val, min, max, step);
23989     nk_property(ctx, name, &variant, inc_per_pixel, NK_FILTER_INT);
23990     val = variant.value.i;
23991     return val;
23992 }
23993 NK_API float
23994 nk_propertyf(struct nk_context *ctx, const char *name, float min,
23995     float val, float max, float step, float inc_per_pixel)
23996 {
23997     struct nk_property_variant variant;
23998     NK_ASSERT(ctx);
23999     NK_ASSERT(name);
24000
24001     if (!ctx || !ctx->current || !name) return val;
24002     variant = nk_property_variant_float(val, min, max, step);
24003     nk_property(ctx, name, &variant, inc_per_pixel, NK_FILTER_FLOAT);
24004     val = variant.value.f;
24005     return val;
24006 }
24007 NK_API double
24008 nk_propertyd(struct nk_context *ctx, const char *name, double min,
24009     double val, double max, double step, float inc_per_pixel)
24010 {
24011     struct nk_property_variant variant;
24012     NK_ASSERT(ctx);
24013     NK_ASSERT(name);
24014
```

```
24015     if (!ctx || !ctx->current || !name) return val;
24016     variant = nk_property_variant_double(val, min, max, step);
24017     nk_property(ctx, name, &variant, inc_per_pixel, NK_FILTER_FLOAT);
24018     val = variant.value.d;
24019     return val;
24020 }
24021
24022
24023
24024
24025
24026 /* ==============================================================
24027  *
24028  *                          CHART
24029  *
24030  * ==============================================================*/
24031 NK_API int
24032 nk_chart_begin_colored(struct nk_context *ctx, enum nk_chart_type type,
24033     struct nk_color color, struct nk_color highlight,
24034     int count, float min_value, float max_value)
24035 {
24036     struct nk_window *win;
24037     struct nk_chart *chart;
24038     const struct nk_style *config;
24039     const struct nk_style_chart *style;
24040
24041     const struct nk_style_item *background;
24042     struct nk_rect bounds = {0, 0, 0, 0};
24043
24044     NK_ASSERT(ctx);
24045     NK_ASSERT(ctx->current);
24046     NK_ASSERT(ctx->current->layout);
24047
24048     if (!ctx || !ctx->current || !ctx->current->layout) return 0;
24049     if (!nk_widget(&bounds, ctx)) {
24050         chart = &ctx->current->layout->chart;
24051         nk_zero(chart, sizeof(*chart));
24052         return 0;
24053     }
24054
24055     win = ctx->current;
24056     config = &ctx->style;
24057     chart = &win->layout->chart;
24058     style = &config->chart;
24059
24060     /* setup basic generic chart  */
24061     nk_zero(chart, sizeof(*chart));
24062     chart->x = bounds.x + style->padding.x;
24063     chart->y = bounds.y + style->padding.y;
24064     chart->w = bounds.w - 2 * style->padding.x;
24065     chart->h = bounds.h - 2 * style->padding.y;
24066     chart->w = NK_MAX(chart->w, 2 * style->padding.x);
24067     chart->h = NK_MAX(chart->h, 2 * style->padding.y);
24068
24069     /* add first slot into chart */
24070     {struct nk_chart_slot *slot = &chart->slots[chart->slot++];
24071     slot->type = type;
24072     slot->count = count;
24073     slot->color = color;
24074     slot->highlight = highlight;
24075     slot->min = NK_MIN(min_value, max_value);
24076     slot->max = NK_MAX(min_value, max_value);
24077     slot->range = slot->max - slot->min;}
24078
24079     /* draw chart background */
24080     background = &style->background;
24081     if (background->type == NK_STYLE_ITEM_IMAGE) {
24082         nk_draw_image(&win->buffer, bounds, &background->data.image, nk_white);
24083     } else {
24084         nk_fill_rect(&win->buffer, bounds, style->rounding, style->border_color);
24085         nk_fill_rect(&win->buffer, nk_shrink_rect(bounds, style->border),
24086             style->rounding, style->background.data.color);
24087     }
24088     return 1;
24089 }
24090 NK_API int
24091 nk_chart_begin(struct nk_context *ctx, const enum nk_chart_type type,
24092     int count, float min_value, float max_value)
24093 {
24094     return nk_chart_begin_colored(ctx, type, ctx->style.chart.color,
24095                 ctx->style.chart.selected_color, count, min_value, max_value);
24096 }
24097 NK_API void
24098 nk_chart_add_slot_colored(struct nk_context *ctx, const enum nk_chart_type type,
24099     struct nk_color color, struct nk_color highlight,
24100     int count, float min_value, float max_value)
24101 {
```

```
24102     NK_ASSERT(ctx);
24103     NK_ASSERT(ctx->current);
24104     NK_ASSERT(ctx->current->layout);
24105     NK_ASSERT(ctx->current->layout->chart.slot < NK_CHART_MAX_SLOT);
24106     if (!ctx || !ctx->current || !ctx->current->layout) return;
24107     if (ctx->current->layout->chart.slot >= NK_CHART_MAX_SLOT) return;
24108
24109     /* add another slot into the graph */
24110     {struct nk_chart *chart = &ctx->current->layout->chart;
24111     struct nk_chart_slot *slot = &chart->slots[chart->slot++];
24112     slot->type = type;
24113     slot->count = count;
24114     slot->color = color;
24115     slot->highlight = highlight;
24116     slot->min = NK_MIN(min_value, max_value);
24117     slot->max = NK_MAX(min_value, max_value);
24118     slot->range = slot->max - slot->min;}
24119 }
24120 NK_API void
24121 nk_chart_add_slot(struct nk_context *ctx, const enum nk_chart_type type,
24122     int count, float min_value, float max_value)
24123 {
24124     nk_chart_add_slot_colored(ctx, type, ctx->style.chart.color,
24125         ctx->style.chart.selected_color, count, min_value, max_value);
24126 }
24127 NK_INTERN nk_flags
24128 nk_chart_push_line(struct nk_context *ctx, struct nk_window *win,
24129     struct nk_chart *g, float value, int slot)
24130 {
24131     struct nk_panel *layout = win->layout;
24132     const struct nk_input *i = &ctx->input;
24133     struct nk_command_buffer *out = &win->buffer;
24134
24135     nk_flags ret = 0;
24136     struct nk_vec2 cur;
24137     struct nk_rect bounds;
24138     struct nk_color color;
24139     float step;
24140     float range;
24141     float ratio;
24142
24143     NK_ASSERT(slot >= 0 && slot < NK_CHART_MAX_SLOT);
24144     step = g->w / (float)g->slots[slot].count;
24145     range = g->slots[slot].max - g->slots[slot].min;
24146     ratio = (value - g->slots[slot].min) / range;
24147
24148     if (g->slots[slot].index == 0) {
24149         /* first data point does not have a connection */
24150         g->slots[slot].last.x = g->x;
24151         g->slots[slot].last.y = (g->y + g->h) - ratio * (float)g->h;
24152
24153         bounds.x = g->slots[slot].last.x - 2;
24154         bounds.y = g->slots[slot].last.y - 2;
24155         bounds.w = bounds.h = 4;
24156
24157         color = g->slots[slot].color;
24158         if (!(layout->flags & NK_WINDOW_ROM) &&
24159             NK_INBOX(i->mouse.pos.x,i->mouse.pos.y, g->slots[slot].last.x-3, g->slots[slot].last.y-3,
      6, 6)){
24160             ret = nk_input_is_mouse_hovering_rect(i, bounds) ? NK_CHART_HOVERING : 0;
24161             ret |= (i->mouse.buttons[NK_BUTTON_LEFT].down &&
24162                 i->mouse.buttons[NK_BUTTON_LEFT].clicked) ? NK_CHART_CLICKED: 0;
24163             color = g->slots[slot].highlight;
24164         }
24165         nk_fill_rect(out, bounds, 0, color);
24166         g->slots[slot].index += 1;
24167         return ret;
24168     }
24169
24170     /* draw a line between the last data point and the new one */
24171     color = g->slots[slot].color;
24172     cur.x = g->x + (float)(step * (float)g->slots[slot].index);
24173     cur.y = (g->y + g->h) - (ratio * (float)g->h);
24174     nk_stroke_line(out, g->slots[slot].last.x, g->slots[slot].last.y, cur.x, cur.y, 1.0f, color);
24175
24176     bounds.x = cur.x - 3;
24177     bounds.y = cur.y - 3;
24178     bounds.w = bounds.h = 6;
24179
24180     /* user selection of current data point */
24181     if (!(layout->flags & NK_WINDOW_ROM)) {
24182         if (nk_input_is_mouse_hovering_rect(i, bounds)) {
24183             ret = NK_CHART_HOVERING;
24184             ret |= (!i->mouse.buttons[NK_BUTTON_LEFT].down &&
24185                 i->mouse.buttons[NK_BUTTON_LEFT].clicked) ? NK_CHART_CLICKED: 0;
24186             color = g->slots[slot].highlight;
24187         }
```

```
24188     }
24189     nk_fill_rect(out, nk_rect(cur.x - 2, cur.y - 2, 4, 4), 0, color);
24190
24191     /* save current data point position */
24192     g->slots[slot].last.x = cur.x;
24193     g->slots[slot].last.y = cur.y;
24194     g->slots[slot].index  += 1;
24195     return ret;
24196 }
24197 NK_INTERN nk_flags
24198 nk_chart_push_column(const struct nk_context *ctx, struct nk_window *win,
24199     struct nk_chart *chart, float value, int slot)
24200 {
24201     struct nk_command_buffer *out = &win->buffer;
24202     const struct nk_input *in = &ctx->input;
24203     struct nk_panel *layout = win->layout;
24204
24205     float ratio;
24206     nk_flags ret = 0;
24207     struct nk_color color;
24208     struct nk_rect item = {0,0,0,0};
24209
24210     NK_ASSERT(slot >= 0 && slot < NK_CHART_MAX_SLOT);
24211     if (chart->slots[slot].index  >= chart->slots[slot].count)
24212         return nk_false;
24213     if (chart->slots[slot].count) {
24214         float padding = (float)(chart->slots[slot].count-1);
24215         item.w = (chart->w - padding) / (float)(chart->slots[slot].count);
24216     }
24217
24218     /* calculate bounds of current bar chart entry */
24219     color = chart->slots[slot].color;;
24220     item.h = chart->h * NK_ABS((value/chart->slots[slot].range));
24221     if (value >= 0) {
24222         ratio = (value + NK_ABS(chart->slots[slot].min)) / NK_ABS(chart->slots[slot].range);
24223         item.y = (chart->y + chart->h) - chart->h * ratio;
24224     } else {
24225         ratio = (value - chart->slots[slot].max) / chart->slots[slot].range;
24226         item.y = chart->y + (chart->h * NK_ABS(ratio)) - item.h;
24227     }
24228     item.x = chart->x + ((float)chart->slots[slot].index * item.w);
24229     item.x = item.x + ((float)chart->slots[slot].index);
24230
24231     /* user chart bar selection */
24232     if (!(layout->flags & NK_WINDOW_ROM) &&
24233         NK_INBOX(in->mouse.pos.x,in->mouse.pos.y,item.x,item.y,item.w,item.h)) {
24234         ret = NK_CHART_HOVERING;
24235         ret |= (!in->mouse.buttons[NK_BUTTON_LEFT].down &&
24236                 in->mouse.buttons[NK_BUTTON_LEFT].clicked) ? NK_CHART_CLICKED: 0;
24237         color = chart->slots[slot].highlight;
24238     }
24239     nk_fill_rect(out, item, 0, color);
24240     chart->slots[slot].index += 1;
24241     return ret;
24242 }
24243 NK_API nk_flags
24244 nk_chart_push_slot(struct nk_context *ctx, float value, int slot)
24245 {
24246     nk_flags flags;
24247     struct nk_window *win;
24248
24249     NK_ASSERT(ctx);
24250     NK_ASSERT(ctx->current);
24251     NK_ASSERT(slot >= 0 && slot < NK_CHART_MAX_SLOT);
24252     NK_ASSERT(slot < ctx->current->layout->chart.slot);
24253     if (!ctx || !ctx->current || slot >= NK_CHART_MAX_SLOT) return nk_false;
24254     if (slot >= ctx->current->layout->chart.slot) return nk_false;
24255
24256     win = ctx->current;
24257     if (win->layout->chart.slot < slot) return nk_false;
24258     switch (win->layout->chart.slots[slot].type) {
24259     case NK_CHART_LINES:
24260         flags = nk_chart_push_line(ctx, win, &win->layout->chart, value, slot); break;
24261     case NK_CHART_COLUMN:
24262         flags = nk_chart_push_column(ctx, win, &win->layout->chart, value, slot); break;
24263     default:
24264     case NK_CHART_MAX:
24265         flags = 0;
24266     }
24267     return flags;
24268 }
24269 NK_API nk_flags
24270 nk_chart_push(struct nk_context *ctx, float value)
24271 {
24272     return nk_chart_push_slot(ctx, value, 0);
24273 }
24274 NK_API void
```

```
24275 nk_chart_end(struct nk_context *ctx)
24276 {
24277     struct nk_window *win;
24278     struct nk_chart *chart;
24279
24280     NK_ASSERT(ctx);
24281     NK_ASSERT(ctx->current);
24282     if (!ctx || !ctx->current)
24283         return;
24284
24285     win = ctx->current;
24286     chart = &win->layout->chart;
24287     NK_MEMSET(chart, 0, sizeof(*chart));
24288     return;
24289 }
24290 NK_API void
24291 nk_plot(struct nk_context *ctx, enum nk_chart_type type, const float *values,
24292     int count, int offset)
24293 {
24294     int i = 0;
24295     float min_value;
24296     float max_value;
24297
24298     NK_ASSERT(ctx);
24299     NK_ASSERT(values);
24300     if (!ctx || !values || !count) return;
24301
24302     min_value = values[offset];
24303     max_value = values[offset];
24304     for (i = 0; i < count; ++i) {
24305         min_value = NK_MIN(values[i + offset], min_value);
24306         max_value = NK_MAX(values[i + offset], max_value);
24307     }
24308
24309     if (nk_chart_begin(ctx, type, count, min_value, max_value)) {
24310         for (i = 0; i < count; ++i)
24311             nk_chart_push(ctx, values[i + offset]);
24312         nk_chart_end(ctx);
24313     }
24314 }
24315 NK_API void
24316 nk_plot_function(struct nk_context *ctx, enum nk_chart_type type, void *userdata,
24317     float(*value_getter)(void* user, int index), int count, int offset)
24318 {
24319     int i = 0;
24320     float min_value;
24321     float max_value;
24322
24323     NK_ASSERT(ctx);
24324     NK_ASSERT(value_getter);
24325     if (!ctx || !value_getter || !count) return;
24326
24327     max_value = min_value = value_getter(userdata, offset);
24328     for (i = 0; i < count; ++i) {
24329         float value = value_getter(userdata, i + offset);
24330         min_value = NK_MIN(value, min_value);
24331         max_value = NK_MAX(value, max_value);
24332     }
24333
24334     if (nk_chart_begin(ctx, type, count, min_value, max_value)) {
24335         for (i = 0; i < count; ++i)
24336             nk_chart_push(ctx, value_getter(userdata, i + offset));
24337         nk_chart_end(ctx);
24338     }
24339 }
24340
24341
24342
24343
24344
24345 /* ===============================================================
24346  *
24347  *                          COLOR PICKER
24348  *
24349  * ===============================================================*/
24350 NK_LIB int
24351 nk_color_picker_behavior(nk_flags *state,
24352     const struct nk_rect *bounds, const struct nk_rect *matrix,
24353     const struct nk_rect *hue_bar, const struct nk_rect *alpha_bar,
24354     struct nk_colorf *color, const struct nk_input *in)
24355 {
24356     float hsva[4];
24357     int value_changed = 0;
24358     int hsv_changed = 0;
24359
24360     NK_ASSERT(state);
24361     NK_ASSERT(matrix);
```

```
24362     NK_ASSERT(hue_bar);
24363     NK_ASSERT(color);
24364
24365     /* color matrix */
24366     nk_colorf_hsva_fv(hsva, *color);
24367     if (nk_button_behavior(state, *matrix, in, NK_BUTTON_REPEATER)) {
24368         hsva[1] = NK_SATURATE((in->mouse.pos.x - matrix->x) / (matrix->w-1));
24369         hsva[2] = 1.0f - NK_SATURATE((in->mouse.pos.y - matrix->y) / (matrix->h-1));
24370         value_changed = hsv_changed = 1;
24371     }
24372     /* hue bar */
24373     if (nk_button_behavior(state, *hue_bar, in, NK_BUTTON_REPEATER)) {
24374         hsva[0] = NK_SATURATE((in->mouse.pos.y - hue_bar->y) / (hue_bar->h-1));
24375         value_changed = hsv_changed = 1;
24376     }
24377     /* alpha bar */
24378     if (alpha_bar) {
24379         if (nk_button_behavior(state, *alpha_bar, in, NK_BUTTON_REPEATER)) {
24380             hsva[3] = 1.0f - NK_SATURATE((in->mouse.pos.y - alpha_bar->y) / (alpha_bar->h-1));
24381             value_changed = 1;
24382         }
24383     }
24384     nk_widget_state_reset(state);
24385     if (hsv_changed) {
24386         *color = nk_hsva_colorfv(hsva);
24387         *state = NK_WIDGET_STATE_ACTIVE;
24388     }
24389     if (value_changed) {
24390         color->a = hsva[3];
24391         *state = NK_WIDGET_STATE_ACTIVE;
24392     }
24393     /* set color picker widget state */
24394     if (nk_input_is_mouse_hovering_rect(in, *bounds))
24395         *state = NK_WIDGET_STATE_HOVERED;
24396     if (*state & NK_WIDGET_STATE_HOVER && !nk_input_is_mouse_prev_hovering_rect(in, *bounds))
24397         *state |= NK_WIDGET_STATE_ENTERED;
24398     else if (nk_input_is_mouse_prev_hovering_rect(in, *bounds))
24399         *state |= NK_WIDGET_STATE_LEFT;
24400     return value_changed;
24401 }
24402 NK_LIB void
24403 nk_draw_color_picker(struct nk_command_buffer *o, const struct nk_rect *matrix,
24404     const struct nk_rect *hue_bar, const struct nk_rect *alpha_bar,
24405     struct nk_colorf col)
24406 {
24407     NK_STORAGE const struct nk_color black = {0,0,0,255};
24408     NK_STORAGE const struct nk_color white = {255, 255, 255, 255};
24409     NK_STORAGE const struct nk_color black_trans = {0,0,0,0};
24410
24411     const float crosshair_size = 7.0f;
24412     struct nk_color temp;
24413     float hsva[4];
24414     float line_y;
24415     int i;
24416
24417     NK_ASSERT(o);
24418     NK_ASSERT(matrix);
24419     NK_ASSERT(hue_bar);
24420
24421     /* draw hue bar */
24422     nk_colorf_hsva_fv(hsva, col);
24423     for (i = 0; i < 6; ++i) {
24424         NK_GLOBAL const struct nk_color hue_colors[] = {
24425             {255, 0, 0, 255}, {255,255,0,255}, {0,255,0,255}, {0, 255,255,255},
24426             {0,0,255,255}, {255, 0, 255, 255}, {255, 0, 0, 255}
24427         };
24428         nk_fill_rect_multi_color(o,
24429             nk_rect(hue_bar->x, hue_bar->y + (float)i * (hue_bar->h/6.0f) + 0.5f,
24430                 hue_bar->w, (hue_bar->h/6.0f) + 0.5f), hue_colors[i], hue_colors[i],
24431                 hue_colors[i+1], hue_colors[i+1]);
24432     }
24433     line_y = (float)(int)(hue_bar->y + hsva[0] * matrix->h + 0.5f);
24434     nk_stroke_line(o, hue_bar->x-1, line_y, hue_bar->x + hue_bar->w + 2,
24435         line_y, 1, nk_rgb(255,255,255));
24436
24437     /* draw alpha bar */
24438     if (alpha_bar) {
24439         float alpha = NK_SATURATE(col.a);
24440         line_y = (float)(int)(alpha_bar->y +  (1.0f - alpha) * matrix->h + 0.5f);
24441
24442         nk_fill_rect_multi_color(o, *alpha_bar, white, white, black, black);
24443         nk_stroke_line(o, alpha_bar->x-1, line_y, alpha_bar->x + alpha_bar->w + 2,
24444             line_y, 1, nk_rgb(255,255,255));
24445     }
24446
24447     /* draw color matrix */
24448     temp = nk_hsv_f(hsva[0], 1.0f, 1.0f);
```

```
24449     nk_fill_rect_multi_color(o, *matrix, white, temp, temp, white);
24450     nk_fill_rect_multi_color(o, *matrix, black_trans, black_trans, black, black);
24451
24452     /* draw cross-hair */
24453     {struct nk_vec2 p; float S = hsva[1]; float V = hsva[2];
24454     p.x = (float)(int)(matrix->x + S * matrix->w);
24455     p.y = (float)(int)(matrix->y + (1.0f - V) * matrix->h);
24456     nk_stroke_line(o, p.x - crosshair_size, p.y, p.x-2, p.y, 1.0f, white);
24457     nk_stroke_line(o, p.x + crosshair_size + 1, p.y, p.x+3, p.y, 1.0f, white);
24458     nk_stroke_line(o, p.x, p.y + crosshair_size + 1, p.x, p.y+3, 1.0f, white);
24459     nk_stroke_line(o, p.x, p.y - crosshair_size, p.x, p.y-2, 1.0f, white);}
24460 }
24461 NK_LIB int
24462 nk_do_color_picker(nk_flags *state,
24463     struct nk_command_buffer *out, struct nk_colorf *col,
24464     enum nk_color_format fmt, struct nk_rect bounds,
24465     struct nk_vec2 padding, const struct nk_input *in,
24466     const struct nk_user_font *font)
24467 {
24468     int ret = 0;
24469     struct nk_rect matrix;
24470     struct nk_rect hue_bar;
24471     struct nk_rect alpha_bar;
24472     float bar_w;
24473
24474     NK_ASSERT(out);
24475     NK_ASSERT(col);
24476     NK_ASSERT(state);
24477     NK_ASSERT(font);
24478     if (!out || !col || !state || !font)
24479         return ret;
24480
24481     bar_w = font->height;
24482     bounds.x += padding.x;
24483     bounds.y += padding.x;
24484     bounds.w -= 2 * padding.x;
24485     bounds.h -= 2 * padding.y;
24486
24487     matrix.x = bounds.x;
24488     matrix.y = bounds.y;
24489     matrix.h = bounds.h;
24490     matrix.w = bounds.w - (3 * padding.x + 2 * bar_w);
24491
24492     hue_bar.w = bar_w;
24493     hue_bar.y = bounds.y;
24494     hue_bar.h = matrix.h;
24495     hue_bar.x = matrix.x + matrix.w + padding.x;
24496
24497     alpha_bar.x = hue_bar.x + hue_bar.w + padding.x;
24498     alpha_bar.y = bounds.y;
24499     alpha_bar.w = bar_w;
24500     alpha_bar.h = matrix.h;
24501
24502     ret = nk_color_picker_behavior(state, &bounds, &matrix, &hue_bar,
24503         (fmt == NK_RGBA) ? &alpha_bar:0, col, in);
24504     nk_draw_color_picker(out, &matrix, &hue_bar, (fmt == NK_RGBA) ? &alpha_bar:0, *col);
24505     return ret;
24506 }
24507 NK_API int
24508 nk_color_pick(struct nk_context * ctx, struct nk_colorf *color,
24509     enum nk_color_format fmt)
24510 {
24511     struct nk_window *win;
24512     struct nk_panel *layout;
24513     const struct nk_style *config;
24514     const struct nk_input *in;
24515
24516     enum nk_widget_layout_states state;
24517     struct nk_rect bounds;
24518
24519     NK_ASSERT(ctx);
24520     NK_ASSERT(color);
24521     NK_ASSERT(ctx->current);
24522     NK_ASSERT(ctx->current->layout);
24523     if (!ctx || !ctx->current || !ctx->current->layout || !color)
24524         return 0;
24525
24526     win = ctx->current;
24527     config = &ctx->style;
24528     layout = win->layout;
24529     state = nk_widget(&bounds, ctx);
24530     if (!state) return 0;
24531     in = (state == NK_WIDGET_ROM || layout->flags & NK_WINDOW_ROM) ? 0 : &ctx->input;
24532     return nk_do_color_picker(&ctx->last_widget_state, &win->buffer, color, fmt, bounds,
24533                 nk_vec2(0,0), in, config->font);
24534 }
24535 NK_API struct nk_colorf
```

```
24536 nk_color_picker(struct nk_context *ctx, struct nk_colorf color,
24537     enum nk_color_format fmt)
24538 {
24539     nk_color_pick(ctx, &color, fmt);
24540     return color;
24541 }
24542
24543
24544
24545
24546
24547 /* ==============================================================
24548  *
24549  *                          COMBO
24550  *
24551  * ==============================================================*/
24552 NK_INTERN int
24553 nk_combo_begin(struct nk_context *ctx, struct nk_window *win,
24554     struct nk_vec2 size, int is_clicked, struct nk_rect header)
24555 {
24556     struct nk_window *popup;
24557     int is_open = 0;
24558     int is_active = 0;
24559     struct nk_rect body;
24560     nk_hash hash;
24561
24562     NK_ASSERT(ctx);
24563     NK_ASSERT(ctx->current);
24564     NK_ASSERT(ctx->current->layout);
24565     if (!ctx || !ctx->current || !ctx->current->layout)
24566         return 0;
24567
24568     popup = win->popup.win;
24569     body.x = header.x;
24570     body.w = size.x;
24571     body.y = header.y + header.h-ctx->style.window.combo_border;
24572     body.h = size.y;
24573
24574     hash = win->popup.combo_count++;
24575     is_open = (popup) ? nk_true:nk_false;
24576     is_active = (popup && (win->popup.name == hash) && win->popup.type == NK_PANEL_COMBO);
24577     if ((is_clicked && is_open && !is_active) || (is_open && !is_active) ||
24578         (!is_open && !is_active && !is_clicked)) return 0;
24579     if (!nk_nonblock_begin(ctx, 0, body,
24580         (is_clicked && is_open)?nk_rect(0,0,0,0):header, NK_PANEL_COMBO)) return 0;
24581
24582     win->popup.type = NK_PANEL_COMBO;
24583     win->popup.name = hash;
24584     return 1;
24585 }
24586 NK_API int
24587 nk_combo_begin_text(struct nk_context *ctx, const char *selected, int len,
24588     struct nk_vec2 size)
24589 {
24590     const struct nk_input *in;
24591     struct nk_window *win;
24592     struct nk_style *style;
24593
24594     enum nk_widget_layout_states s;
24595     int is_clicked = nk_false;
24596     struct nk_rect header;
24597     const struct nk_style_item *background;
24598     struct nk_text text;
24599
24600     NK_ASSERT(ctx);
24601     NK_ASSERT(selected);
24602     NK_ASSERT(ctx->current);
24603     NK_ASSERT(ctx->current->layout);
24604     if (!ctx || !ctx->current || !ctx->current->layout || !selected)
24605         return 0;
24606
24607     win = ctx->current;
24608     style = &ctx->style;
24609     s = nk_widget(&header, ctx);
24610     if (s == NK_WIDGET_INVALID)
24611         return 0;
24612
24613     in = (win->layout->flags & NK_WINDOW_ROM || s == NK_WIDGET_ROM)? 0: &ctx->input;
24614     if (nk_button_behavior(&ctx->last_widget_state, header, in, NK_BUTTON_DEFAULT))
24615         is_clicked = nk_true;
24616
24617     /* draw combo box header background and border */
24618     if (ctx->last_widget_state & NK_WIDGET_STATE_ACTIVED) {
24619         background = &style->combo.active;
24620         text.text = style->combo.label_active;
24621     } else if (ctx->last_widget_state & NK_WIDGET_STATE_HOVER) {
24622         background = &style->combo.hover;
```

```
24623         text.text = style->combo.label_hover;
24624     } else {
24625         background = &style->combo.normal;
24626         text.text = style->combo.label_normal;
24627     }
24628     if (background->type == NK_STYLE_ITEM_IMAGE) {
24629         text.background = nk_rgba(0,0,0,0);
24630         nk_draw_image(&win->buffer, header, &background->data.image, nk_white);
24631     } else {
24632         text.background = background->data.color;
24633         nk_fill_rect(&win->buffer, header, style->combo.rounding, background->data.color);
24634         nk_stroke_rect(&win->buffer, header, style->combo.rounding, style->combo.border,
      style->combo.border_color);
24635     }
24636     {
24637         /* print currently selected text item */
24638         struct nk_rect label;
24639         struct nk_rect button;
24640         struct nk_rect content;
24641
24642         enum nk_symbol_type sym;
24643         if (ctx->last_widget_state & NK_WIDGET_STATE_HOVER)
24644             sym = style->combo.sym_hover;
24645         else if (is_clicked)
24646             sym = style->combo.sym_active;
24647         else sym = style->combo.sym_normal;
24648
24649         /* calculate button */
24650         button.w = header.h - 2 * style->combo.button_padding.y;
24651         button.x = (header.x + header.w - header.h) - style->combo.button_padding.x;
24652         button.y = header.y + style->combo.button_padding.y;
24653         button.h = button.w;
24654
24655         content.x = button.x + style->combo.button.padding.x;
24656         content.y = button.y + style->combo.button.padding.y;
24657         content.w = button.w - 2 * style->combo.button.padding.x;
24658         content.h = button.h - 2 * style->combo.button.padding.y;
24659
24660         /* draw selected label */
24661         text.padding = nk_vec2(0,0);
24662         label.x = header.x + style->combo.content_padding.x;
24663         label.y = header.y + style->combo.content_padding.y;
24664         label.w = button.x - (style->combo.content_padding.x + style->combo.spacing.x) - label.x;;
24665         label.h = header.h - 2 * style->combo.content_padding.y;
24666         nk_widget_text(&win->buffer, label, selected, len, &text,
24667             NK_TEXT_LEFT, ctx->style.font);
24668
24669         /* draw open/close button */
24670         nk_draw_button_symbol(&win->buffer, &button, &content, ctx->last_widget_state,
24671             &ctx->style.combo.button, sym, style->font);
24672     }
24673     return nk_combo_begin(ctx, win, size, is_clicked, header);
24674 }
24675 NK_API int
24676 nk_combo_begin_label(struct nk_context *ctx, const char *selected, struct nk_vec2 size)
24677 {
24678     return nk_combo_begin_text(ctx, selected, nk_strlen(selected), size);
24679 }
24680 NK_API int
24681 nk_combo_begin_color(struct nk_context *ctx, struct nk_color color, struct nk_vec2 size)
24682 {
24683     struct nk_window *win;
24684     struct nk_style *style;
24685     const struct nk_input *in;
24686
24687     struct nk_rect header;
24688     int is_clicked = nk_false;
24689     enum nk_widget_layout_states s;
24690     const struct nk_style_item *background;
24691
24692     NK_ASSERT(ctx);
24693     NK_ASSERT(ctx->current);
24694     NK_ASSERT(ctx->current->layout);
24695     if (!ctx || !ctx->current || !ctx->current->layout)
24696         return 0;
24697
24698     win = ctx->current;
24699     style = &ctx->style;
24700     s = nk_widget(&header, ctx);
24701     if (s == NK_WIDGET_INVALID)
24702         return 0;
24703
24704     in = (win->layout->flags & NK_WINDOW_ROM || s == NK_WIDGET_ROM)? 0: &ctx->input;
24705     if (nk_button_behavior(&ctx->last_widget_state, header, in, NK_BUTTON_DEFAULT))
24706         is_clicked = nk_true;
24707
24708     /* draw combo box header background and border */
```

```
24709     if (ctx->last_widget_state & NK_WIDGET_STATE_ACTIVED)
24710         background = &style->combo.active;
24711     else if (ctx->last_widget_state & NK_WIDGET_STATE_HOVER)
24712         background = &style->combo.hover;
24713     else background = &style->combo.normal;
24714
24715     if (background->type == NK_STYLE_ITEM_IMAGE) {
24716         nk_draw_image(&win->buffer, header, &background->data.image,nk_white);
24717     } else {
24718         nk_fill_rect(&win->buffer, header, style->combo.rounding, background->data.color);
24719         nk_stroke_rect(&win->buffer, header, style->combo.rounding, style->combo.border,
       style->combo.border_color);
24720     }
24721     {
24722         struct nk_rect content;
24723         struct nk_rect button;
24724         struct nk_rect bounds;
24725
24726         enum nk_symbol_type sym;
24727         if (ctx->last_widget_state & NK_WIDGET_STATE_HOVER)
24728             sym = style->combo.sym_hover;
24729         else if (is_clicked)
24730             sym = style->combo.sym_active;
24731         else sym = style->combo.sym_normal;
24732
24733         /* calculate button */
24734         button.w = header.h - 2 * style->combo.button_padding.y;
24735         button.x = (header.x + header.w - header.h) - style->combo.button_padding.x;
24736         button.y = header.y + style->combo.button_padding.y;
24737         button.h = button.w;
24738
24739         content.x = button.x + style->combo.button.padding.x;
24740         content.y = button.y + style->combo.button.padding.y;
24741         content.w = button.w - 2 * style->combo.button.padding.x;
24742         content.h = button.h - 2 * style->combo.button.padding.y;
24743
24744         /* draw color */
24745         bounds.h = header.h - 4 * style->combo.content_padding.y;
24746         bounds.y = header.y + 2 * style->combo.content_padding.y;
24747         bounds.x = header.x + 2 * style->combo.content_padding.x;
24748         bounds.w = (button.x - (style->combo.content_padding.x + style->combo.spacing.x)) - bounds.x;
24749         nk_fill_rect(&win->buffer, bounds, 0, color);
24750
24751         /* draw open/close button */
24752         nk_draw_button_symbol(&win->buffer, &button, &content, ctx->last_widget_state,
24753             &ctx->style.combo.button, sym, style->font);
24754     }
24755     return nk_combo_begin(ctx, win, size, is_clicked, header);
24756 }
24757 NK_API int
24758 nk_combo_begin_symbol(struct nk_context *ctx, enum nk_symbol_type symbol, struct nk_vec2 size)
24759 {
24760     struct nk_window *win;
24761     struct nk_style *style;
24762     const struct nk_input *in;
24763
24764     struct nk_rect header;
24765     int is_clicked = nk_false;
24766     enum nk_widget_layout_states s;
24767     const struct nk_style_item *background;
24768     struct nk_color sym_background;
24769     struct nk_color symbol_color;
24770
24771     NK_ASSERT(ctx);
24772     NK_ASSERT(ctx->current);
24773     NK_ASSERT(ctx->current->layout);
24774     if (!ctx || !ctx->current || !ctx->current->layout)
24775         return 0;
24776
24777     win = ctx->current;
24778     style = &ctx->style;
24779     s = nk_widget(&header, ctx);
24780     if (s == NK_WIDGET_INVALID)
24781         return 0;
24782
24783     in = (win->layout->flags & NK_WINDOW_ROM || s == NK_WIDGET_ROM)? 0: &ctx->input;
24784     if (nk_button_behavior(&ctx->last_widget_state, header, in, NK_BUTTON_DEFAULT))
24785         is_clicked = nk_true;
24786
24787     /* draw combo box header background and border */
24788     if (ctx->last_widget_state & NK_WIDGET_STATE_ACTIVED) {
24789         background = &style->combo.active;
24790         symbol_color = style->combo.symbol_active;
24791     } else if (ctx->last_widget_state & NK_WIDGET_STATE_HOVER) {
24792         background = &style->combo.hover;
24793         symbol_color = style->combo.symbol_hover;
24794     } else {
```

```
24795         background = &style->combo.normal;
24796         symbol_color = style->combo.symbol_hover;
24797     }
24798
24799     if (background->type == NK_STYLE_ITEM_IMAGE) {
24800         sym_background = nk_rgba(0,0,0,0);
24801         nk_draw_image(&win->buffer, header, &background->data.image, nk_white);
24802     } else {
24803         sym_background = background->data.color;
24804         nk_fill_rect(&win->buffer, header, style->combo.rounding, background->data.color);
24805         nk_stroke_rect(&win->buffer, header, style->combo.rounding, style->combo.border,
       style->combo.border_color);
24806     }
24807     {
24808         struct nk_rect bounds = {0,0,0,0};
24809         struct nk_rect content;
24810         struct nk_rect button;
24811
24812         enum nk_symbol_type sym;
24813         if (ctx->last_widget_state & NK_WIDGET_STATE_HOVER)
24814             sym = style->combo.sym_hover;
24815         else if (is_clicked)
24816             sym = style->combo.sym_active;
24817         else sym = style->combo.sym_normal;
24818
24819         /* calculate button */
24820         button.w = header.h - 2 * style->combo.button_padding.y;
24821         button.x = (header.x + header.w - header.h) - style->combo.button_padding.y;
24822         button.y = header.y + style->combo.button_padding.y;
24823         button.h = button.w;
24824
24825         content.x = button.x + style->combo.button.padding.x;
24826         content.y = button.y + style->combo.button.padding.y;
24827         content.w = button.w - 2 * style->combo.button.padding.x;
24828         content.h = button.h - 2 * style->combo.button.padding.y;
24829
24830         /* draw symbol */
24831         bounds.h = header.h - 2 * style->combo.content_padding.y;
24832         bounds.y = header.y + style->combo.content_padding.y;
24833         bounds.x = header.x + style->combo.content_padding.x;
24834         bounds.w = (button.x - style->combo.content_padding.y) - bounds.x;
24835         nk_draw_symbol(&win->buffer, symbol, bounds, sym_background, symbol_color,
24836             1.0f, style->font);
24837
24838         /* draw open/close button */
24839         nk_draw_button_symbol(&win->buffer, &bounds, &content, ctx->last_widget_state,
24840             &ctx->style.combo.button, sym, style->font);
24841     }
24842     return nk_combo_begin(ctx, win, size, is_clicked, header);
24843 }
24844 NK_API int
24845 nk_combo_begin_symbol_text(struct nk_context *ctx, const char *selected, int len,
24846     enum nk_symbol_type symbol, struct nk_vec2 size)
24847 {
24848     struct nk_window *win;
24849     struct nk_style *style;
24850     struct nk_input *in;
24851
24852     struct nk_rect header;
24853     int is_clicked = nk_false;
24854     enum nk_widget_layout_states s;
24855     const struct nk_style_item *background;
24856     struct nk_color symbol_color;
24857     struct nk_text text;
24858
24859     NK_ASSERT(ctx);
24860     NK_ASSERT(ctx->current);
24861     NK_ASSERT(ctx->current->layout);
24862     if (!ctx || !ctx->current || !ctx->current->layout)
24863         return 0;
24864
24865     win = ctx->current;
24866     style = &ctx->style;
24867     s = nk_widget(&header, ctx);
24868     if (!s) return 0;
24869
24870     in = (win->layout->flags & NK_WINDOW_ROM || s == NK_WIDGET_ROM)? 0: &ctx->input;
24871     if (nk_button_behavior(&ctx->last_widget_state, header, in, NK_BUTTON_DEFAULT))
24872         is_clicked = nk_true;
24873
24874     /* draw combo box header background and border */
24875     if (ctx->last_widget_state & NK_WIDGET_STATE_ACTIVED) {
24876         background = &style->combo.active;
24877         symbol_color = style->combo.symbol_active;
24878         text.text = style->combo.label_active;
24879     } else if (ctx->last_widget_state & NK_WIDGET_STATE_HOVER) {
24880         background = &style->combo.hover;
```

```
24881         symbol_color = style->combo.symbol_hover;
24882         text.text = style->combo.label_hover;
24883     } else {
24884         background = &style->combo.normal;
24885         symbol_color = style->combo.symbol_normal;
24886         text.text = style->combo.label_normal;
24887     }
24888     if (background->type == NK_STYLE_ITEM_IMAGE) {
24889         text.background = nk_rgba(0,0,0,0);
24890         nk_draw_image(&win->buffer, header, &background->data.image, nk_white);
24891     } else {
24892         text.background = background->data.color;
24893         nk_fill_rect(&win->buffer, header, style->combo.rounding, background->data.color);
24894         nk_stroke_rect(&win->buffer, header, style->combo.rounding, style->combo.border,
       style->combo.border_color);
24895     }
24896     {
24897         struct nk_rect content;
24898         struct nk_rect button;
24899         struct nk_rect label;
24900         struct nk_rect image;
24901
24902         enum nk_symbol_type sym;
24903         if (ctx->last_widget_state & NK_WIDGET_STATE_HOVER)
24904             sym = style->combo.sym_hover;
24905         else if (is_clicked)
24906             sym = style->combo.sym_active;
24907         else sym = style->combo.sym_normal;
24908
24909         /* calculate button */
24910         button.w = header.h - 2 * style->combo.button_padding.y;
24911         button.x = (header.x + header.w - header.h) - style->combo.button_padding.x;
24912         button.y = header.y + style->combo.button_padding.y;
24913         button.h = button.w;
24914
24915         content.x = button.x + style->combo.button.padding.x;
24916         content.y = button.y + style->combo.button.padding.y;
24917         content.w = button.w - 2 * style->combo.button.padding.x;
24918         content.h = button.h - 2 * style->combo.button.padding.y;
24919         nk_draw_button_symbol(&win->buffer, &button, &content, ctx->last_widget_state,
24920             &ctx->style.combo.button, sym, style->font);
24921
24922         /* draw symbol */
24923         image.x = header.x + style->combo.content_padding.x;
24924         image.y = header.y + style->combo.content_padding.y;
24925         image.h = header.h - 2 * style->combo.content_padding.y;
24926         image.w = image.h;
24927         nk_draw_symbol(&win->buffer, symbol, image, text.background, symbol_color,
24928             1.0f, style->font);
24929
24930         /* draw label */
24931         text.padding = nk_vec2(0,0);
24932         label.x = image.x + image.w + style->combo.spacing.x + style->combo.content_padding.x;
24933         label.y = header.y + style->combo.content_padding.y;
24934         label.w = (button.x - style->combo.content_padding.x) - label.x;
24935         label.h = header.h - 2 * style->combo.content_padding.y;
24936         nk_widget_text(&win->buffer, label, selected, len, &text, NK_TEXT_LEFT, style->font);
24937     }
24938     return nk_combo_begin(ctx, win, size, is_clicked, header);
24939 }
24940 NK_API int
24941 nk_combo_begin_image(struct nk_context *ctx, struct nk_image img, struct nk_vec2 size)
24942 {
24943     struct nk_window *win;
24944     struct nk_style *style;
24945     const struct nk_input *in;
24946
24947     struct nk_rect header;
24948     int is_clicked = nk_false;
24949     enum nk_widget_layout_states s;
24950     const struct nk_style_item *background;
24951
24952     NK_ASSERT(ctx);
24953     NK_ASSERT(ctx->current);
24954     NK_ASSERT(ctx->current->layout);
24955     if (!ctx || !ctx->current || !ctx->current->layout)
24956         return 0;
24957
24958     win = ctx->current;
24959     style = &ctx->style;
24960     s = nk_widget(&header, ctx);
24961     if (s == NK_WIDGET_INVALID)
24962         return 0;
24963
24964     in = (win->layout->flags & NK_WINDOW_ROM || s == NK_WIDGET_ROM)? 0: &ctx->input;
24965     if (nk_button_behavior(&ctx->last_widget_state, header, in, NK_BUTTON_DEFAULT))
24966         is_clicked = nk_true;
```

```
24967
24968     /* draw combo box header background and border */
24969     if (ctx->last_widget_state & NK_WIDGET_STATE_ACTIVED)
24970         background = &style->combo.active;
24971     else if (ctx->last_widget_state & NK_WIDGET_STATE_HOVER)
24972         background = &style->combo.hover;
24973     else background = &style->combo.normal;
24974
24975     if (background->type == NK_STYLE_ITEM_IMAGE) {
24976         nk_draw_image(&win->buffer, header, &background->data.image, nk_white);
24977     } else {
24978         nk_fill_rect(&win->buffer, header, style->combo.rounding, background->data.color);
24979         nk_stroke_rect(&win->buffer, header, style->combo.rounding, style->combo.border,
      style->combo.border_color);
24980     }
24981     {
24982         struct nk_rect bounds = {0,0,0,0};
24983         struct nk_rect content;
24984         struct nk_rect button;
24985
24986         enum nk_symbol_type sym;
24987         if (ctx->last_widget_state & NK_WIDGET_STATE_HOVER)
24988             sym = style->combo.sym_hover;
24989         else if (is_clicked)
24990             sym = style->combo.sym_active;
24991         else sym = style->combo.sym_normal;
24992
24993         /* calculate button */
24994         button.w = header.h - 2 * style->combo.button_padding.y;
24995         button.x = (header.x + header.w - header.h) - style->combo.button_padding.y;
24996         button.y = header.y + style->combo.button_padding.y;
24997         button.h = button.w;
24998
24999         content.x = button.x + style->combo.button.padding.x;
25000         content.y = button.y + style->combo.button.padding.y;
25001         content.w = button.w - 2 * style->combo.button.padding.x;
25002         content.h = button.h - 2 * style->combo.button.padding.y;
25003
25004         /* draw image */
25005         bounds.h = header.h - 2 * style->combo.content_padding.y;
25006         bounds.y = header.y + style->combo.content_padding.y;
25007         bounds.x = header.x + style->combo.content_padding.x;
25008         bounds.w = (button.x - style->combo.content_padding.y) - bounds.x;
25009         nk_draw_image(&win->buffer, bounds, &img, nk_white);
25010
25011         /* draw open/close button */
25012         nk_draw_button_symbol(&win->buffer, &bounds, &content, ctx->last_widget_state,
25013             &ctx->style.combo.button, sym, style->font);
25014     }
25015     return nk_combo_begin(ctx, win, size, is_clicked, header);
25016 }
25017 NK_API int
25018 nk_combo_begin_image_text(struct nk_context *ctx, const char *selected, int len,
25019     struct nk_image img, struct nk_vec2 size)
25020 {
25021     struct nk_window *win;
25022     struct nk_style *style;
25023     struct nk_input *in;
25024
25025     struct nk_rect header;
25026     int is_clicked = nk_false;
25027     enum nk_widget_layout_states s;
25028     const struct nk_style_item *background;
25029     struct nk_text text;
25030
25031     NK_ASSERT(ctx);
25032     NK_ASSERT(ctx->current);
25033     NK_ASSERT(ctx->current->layout);
25034     if (!ctx || !ctx->current || !ctx->current->layout)
25035         return 0;
25036
25037     win = ctx->current;
25038     style = &ctx->style;
25039     s = nk_widget(&header, ctx);
25040     if (!s) return 0;
25041
25042     in = (win->layout->flags & NK_WINDOW_ROM || s == NK_WIDGET_ROM)? 0: &ctx->input;
25043     if (nk_button_behavior(&ctx->last_widget_state, header, in, NK_BUTTON_DEFAULT))
25044         is_clicked = nk_true;
25045
25046     /* draw combo box header background and border */
25047     if (ctx->last_widget_state & NK_WIDGET_STATE_ACTIVED) {
25048         background = &style->combo.active;
25049         text.text = style->combo.label_active;
25050     } else if (ctx->last_widget_state & NK_WIDGET_STATE_HOVER) {
25051         background = &style->combo.hover;
25052         text.text = style->combo.label_hover;
```

```
25053     } else {
25054         background = &style->combo.normal;
25055         text.text = style->combo.label_normal;
25056     }
25057     if (background->type == NK_STYLE_ITEM_IMAGE) {
25058         text.background = nk_rgba(0,0,0,0);
25059         nk_draw_image(&win->buffer, header, &background->data.image, nk_white);
25060     } else {
25061         text.background = background->data.color;
25062         nk_fill_rect(&win->buffer, header, style->combo.rounding, background->data.color);
25063         nk_stroke_rect(&win->buffer, header, style->combo.rounding, style->combo.border,
      style->combo.border_color);
25064     }
25065     {
25066         struct nk_rect content;
25067         struct nk_rect button;
25068         struct nk_rect label;
25069         struct nk_rect image;
25070
25071         enum nk_symbol_type sym;
25072         if (ctx->last_widget_state & NK_WIDGET_STATE_HOVER)
25073             sym = style->combo.sym_hover;
25074         else if (is_clicked)
25075             sym = style->combo.sym_active;
25076         else sym = style->combo.sym_normal;
25077
25078         /* calculate button */
25079         button.w = header.h - 2 * style->combo.button_padding.y;
25080         button.x = (header.x + header.w - header.h) - style->combo.button_padding.x;
25081         button.y = header.y + style->combo.button_padding.y;
25082         button.h = button.w;
25083
25084         content.x = button.x + style->combo.button.padding.x;
25085         content.y = button.y + style->combo.button.padding.y;
25086         content.w = button.w - 2 * style->combo.button.padding.x;
25087         content.h = button.h - 2 * style->combo.button.padding.y;
25088         nk_draw_button_symbol(&win->buffer, &button, &content, ctx->last_widget_state,
25089             &ctx->style.combo.button, sym, style->font);
25090
25091         /* draw image */
25092         image.x = header.x + style->combo.content_padding.x;
25093         image.y = header.y + style->combo.content_padding.y;
25094         image.h = header.h - 2 * style->combo.content_padding.y;
25095         image.w = image.h;
25096         nk_draw_image(&win->buffer, image, &img, nk_white);
25097
25098         /* draw label */
25099         text.padding = nk_vec2(0,0);
25100         label.x = image.x + image.w + style->combo.spacing.x + style->combo.content_padding.x;
25101         label.y = header.y + style->combo.content_padding.y;
25102         label.w = (button.x - style->combo.content_padding.x) - label.x;
25103         label.h = header.h - 2 * style->combo.content_padding.y;
25104         nk_widget_text(&win->buffer, label, selected, len, &text, NK_TEXT_LEFT, style->font);
25105     }
25106     return nk_combo_begin(ctx, win, size, is_clicked, header);
25107 }
25108 NK_API int
25109 nk_combo_begin_symbol_label(struct nk_context *ctx,
25110     const char *selected, enum nk_symbol_type type, struct nk_vec2 size)
25111 {
25112     return nk_combo_begin_symbol_text(ctx, selected, nk_strlen(selected), type, size);
25113 }
25114 NK_API int
25115 nk_combo_begin_image_label(struct nk_context *ctx,
25116     const char *selected, struct nk_image img, struct nk_vec2 size)
25117 {
25118     return nk_combo_begin_image_text(ctx, selected, nk_strlen(selected), img, size);
25119 }
25120 NK_API int
25121 nk_combo_item_text(struct nk_context *ctx, const char *text, int len,nk_flags align)
25122 {
25123     return nk_contextual_item_text(ctx, text, len, align);
25124 }
25125 NK_API int
25126 nk_combo_item_label(struct nk_context *ctx, const char *label, nk_flags align)
25127 {
25128     return nk_contextual_item_label(ctx, label, align);
25129 }
25130 NK_API int
25131 nk_combo_item_image_text(struct nk_context *ctx, struct nk_image img, const char *text,
25132     int len, nk_flags alignment)
25133 {
25134     return nk_contextual_item_image_text(ctx, img, text, len, alignment);
25135 }
25136 NK_API int
25137 nk_combo_item_image_label(struct nk_context *ctx, struct nk_image img,
25138     const char *text, nk_flags alignment)
```

```
25139 {
25140     return nk_contextual_item_image_label(ctx, img, text, alignment);
25141 }
25142 NK_API int
25143 nk_combo_item_symbol_text(struct nk_context *ctx, enum nk_symbol_type sym,
25144     const char *text, int len, nk_flags alignment)
25145 {
25146     return nk_contextual_item_symbol_text(ctx, sym, text, len, alignment);
25147 }
25148 NK_API int
25149 nk_combo_item_symbol_label(struct nk_context *ctx, enum nk_symbol_type sym,
25150     const char *label, nk_flags alignment)
25151 {
25152     return nk_contextual_item_symbol_label(ctx, sym, label, alignment);
25153 }
25154 NK_API void nk_combo_end(struct nk_context *ctx)
25155 {
25156     nk_contextual_end(ctx);
25157 }
25158 NK_API void nk_combo_close(struct nk_context *ctx)
25159 {
25160     nk_contextual_close(ctx);
25161 }
25162 NK_API int
25163 nk_combo(struct nk_context *ctx, const char **items, int count,
25164     int selected, int item_height, struct nk_vec2 size)
25165 {
25166     int i = 0;
25167     int max_height;
25168     struct nk_vec2 item_spacing;
25169     struct nk_vec2 window_padding;
25170
25171     NK_ASSERT(ctx);
25172     NK_ASSERT(items);
25173     NK_ASSERT(ctx->current);
25174     if (!ctx || !items ||!count)
25175         return selected;
25176
25177     item_spacing = ctx->style.window.spacing;
25178     window_padding = nk_panel_get_padding(&ctx->style, ctx->current->layout->type);
25179     max_height = count * item_height + count * (int)item_spacing.y;
25180     max_height += (int)item_spacing.y * 2 + (int)window_padding.y * 2;
25181     size.y = NK_MIN(size.y, (float)max_height);
25182     if (nk_combo_begin_label(ctx, items[selected], size)) {
25183         nk_layout_row_dynamic(ctx, (float)item_height, 1);
25184         for (i = 0; i < count; ++i) {
25185             if (nk_combo_item_label(ctx, items[i], NK_TEXT_LEFT))
25186                 selected = i;
25187         }
25188         nk_combo_end(ctx);
25189     }
25190     return selected;
25191 }
25192 NK_API int
25193 nk_combo_separator(struct nk_context *ctx, const char *items_separated_by_separator,
25194     int separator, int selected, int count, int item_height, struct nk_vec2 size)
25195 {
25196     int i;
25197     int max_height;
25198     struct nk_vec2 item_spacing;
25199     struct nk_vec2 window_padding;
25200     const char *current_item;
25201     const char *iter;
25202     int length = 0;
25203
25204     NK_ASSERT(ctx);
25205     NK_ASSERT(items_separated_by_separator);
25206     if (!ctx || !items_separated_by_separator)
25207         return selected;
25208
25209     /* calculate popup window */
25210     item_spacing = ctx->style.window.spacing;
25211     window_padding = nk_panel_get_padding(&ctx->style, ctx->current->layout->type);
25212     max_height = count * item_height + count * (int)item_spacing.y;
25213     max_height += (int)item_spacing.y * 2 + (int)window_padding.y * 2;
25214     size.y = NK_MIN(size.y, (float)max_height);
25215
25216     /* find selected item */
25217     current_item = items_separated_by_separator;
25218     for (i = 0; i < count; ++i) {
25219         iter = current_item;
25220         while (*iter && *iter != separator) iter++;
25221         length = (int)(iter - current_item);
25222         if (i == selected) break;
25223         current_item = iter + 1;
25224     }
25225
```

```
25226     if (nk_combo_begin_text(ctx, current_item, length, size)) {
25227         current_item = items_separated_by_separator;
25228         nk_layout_row_dynamic(ctx, (float)item_height, 1);
25229         for (i = 0; i < count; ++i) {
25230             iter = current_item;
25231             while (*iter && *iter != separator) iter++;
25232             length = (int)(iter - current_item);
25233             if (nk_combo_item_text(ctx, current_item, length, NK_TEXT_LEFT))
25234                 selected = i;
25235             current_item = current_item + length + 1;
25236         }
25237         nk_combo_end(ctx);
25238     }
25239     return selected;
25240 }
25241 NK_API int
25242 nk_combo_string(struct nk_context *ctx, const char *items_separated_by_zeros,
25243     int selected, int count, int item_height, struct nk_vec2 size)
25244 {
25245     return nk_combo_separator(ctx, items_separated_by_zeros, '\0', selected, count, item_height,
       size);
25246 }
25247 NK_API int
25248 nk_combo_callback(struct nk_context *ctx, void(*item_getter)(void*, int, const char**),
25249     void *userdata, int selected, int count, int item_height, struct nk_vec2 size)
25250 {
25251     int i;
25252     int max_height;
25253     struct nk_vec2 item_spacing;
25254     struct nk_vec2 window_padding;
25255     const char *item;
25256
25257     NK_ASSERT(ctx);
25258     NK_ASSERT(item_getter);
25259     if (!ctx || !item_getter)
25260         return selected;
25261
25262     /* calculate popup window */
25263     item_spacing = ctx->style.window.spacing;
25264     window_padding = nk_panel_get_padding(&ctx->style, ctx->current->layout->type);
25265     max_height = count * item_height + count * (int)item_spacing.y;
25266     max_height += (int)item_spacing.y * 2 + (int)window_padding.y * 2;
25267     size.y = NK_MIN(size.y, (float)max_height);
25268
25269     item_getter(userdata, selected, &item);
25270     if (nk_combo_begin_label(ctx, item, size)) {
25271         nk_layout_row_dynamic(ctx, (float)item_height, 1);
25272         for (i = 0; i < count; ++i) {
25273             item_getter(userdata, i, &item);
25274             if (nk_combo_item_label(ctx, item, NK_TEXT_LEFT))
25275                 selected = i;
25276         }
25277         nk_combo_end(ctx);
25278     } return selected;
25279 }
25280 NK_API void
25281 nk_combobox(struct nk_context *ctx, const char **items, int count,
25282     int *selected, int item_height, struct nk_vec2 size)
25283 {
25284     *selected = nk_combo(ctx, items, count, *selected, item_height, size);
25285 }
25286 NK_API void
25287 nk_combobox_string(struct nk_context *ctx, const char *items_separated_by_zeros,
25288     int *selected, int count, int item_height, struct nk_vec2 size)
25289 {
25290     *selected = nk_combo_string(ctx, items_separated_by_zeros, *selected, count, item_height, size);
25291 }
25292 NK_API void
25293 nk_combobox_separator(struct nk_context *ctx, const char *items_separated_by_separator,
25294     int separator,int *selected, int count, int item_height, struct nk_vec2 size)
25295 {
25296     *selected = nk_combo_separator(ctx, items_separated_by_separator, separator,
25297                                    *selected, count, item_height, size);
25298 }
25299 NK_API void
25300 nk_combobox_callback(struct nk_context *ctx,
25301     void(*item_getter)(void* data, int id, const char **out_text),
25302     void *userdata, int *selected, int count, int item_height, struct nk_vec2 size)
25303 {
25304     *selected = nk_combo_callback(ctx, item_getter, userdata,  *selected, count, item_height, size);
25305 }
25306
25307
25308
25309
25310
25311 /* ===============================================================
```

```
25312  *
25313  *                              TOOLTIP
25314  *
25315  * ==============================================================*/
25316 NK_API int
25317 nk_tooltip_begin(struct nk_context *ctx, float width)
25318 {
25319     int x,y,w,h;
25320     struct nk_window *win;
25321     const struct nk_input *in;
25322     struct nk_rect bounds;
25323     int ret;
25324
25325     NK_ASSERT(ctx);
25326     NK_ASSERT(ctx->current);
25327     NK_ASSERT(ctx->current->layout);
25328     if (!ctx || !ctx->current || !ctx->current->layout)
25329         return 0;
25330
25331     /* make sure that no nonblocking popup is currently active */
25332     win = ctx->current;
25333     in = &ctx->input;
25334     if (win->popup.win && (win->popup.type & NK_PANEL_SET_NONBLOCK))
25335         return 0;
25336
25337     w = nk_iceilf(width);
25338     h = nk_iceilf(nk_null_rect.h);
25339     x = nk_ifloorf(in->mouse.pos.x + 1) - (int)win->layout->clip.x;
25340     y = nk_ifloorf(in->mouse.pos.y + 1) - (int)win->layout->clip.y;
25341
25342     bounds.x = (float)x;
25343     bounds.y = (float)y;
25344     bounds.w = (float)w;
25345     bounds.h = (float)h;
25346
25347     ret = nk_popup_begin(ctx, NK_POPUP_DYNAMIC,
25348         "__##Tooltip##__", NK_WINDOW_NO_SCROLLBAR|NK_WINDOW_BORDER, bounds);
25349     if (ret) win->layout->flags &= ~(nk_flags)NK_WINDOW_ROM;
25350     win->popup.type = NK_PANEL_TOOLTIP;
25351     ctx->current->layout->type = NK_PANEL_TOOLTIP;
25352     return ret;
25353 }
25354
25355 NK_API void
25356 nk_tooltip_end(struct nk_context *ctx)
25357 {
25358     NK_ASSERT(ctx);
25359     NK_ASSERT(ctx->current);
25360     if (!ctx || !ctx->current) return;
25361     ctx->current->seq--;
25362     nk_popup_close(ctx);
25363     nk_popup_end(ctx);
25364 }
25365 NK_API void
25366 nk_tooltip(struct nk_context *ctx, const char *text)
25367 {
25368     const struct nk_style *style;
25369     struct nk_vec2 padding;
25370
25371     int text_len;
25372     float text_width;
25373     float text_height;
25374
25375     NK_ASSERT(ctx);
25376     NK_ASSERT(ctx->current);
25377     NK_ASSERT(ctx->current->layout);
25378     NK_ASSERT(text);
25379     if (!ctx || !ctx->current || !ctx->current->layout || !text)
25380         return;
25381
25382     /* fetch configuration data */
25383     style = &ctx->style;
25384     padding = style->window.padding;
25385
25386     /* calculate size of the text and tooltip */
25387     text_len = nk_strlen(text);
25388     text_width = style->font->width(style->font->userdata,
25389                     style->font->height, text, text_len);
25390     text_width += (4 * padding.x);
25391     text_height = (style->font->height + 2 * padding.y);
25392
25393     /* execute tooltip and fill with text */
25394     if (nk_tooltip_begin(ctx, (float)text_width)) {
25395         nk_layout_row_dynamic(ctx, (float)text_height, 1);
25396         nk_text(ctx, text, text_len, NK_TEXT_LEFT);
25397         nk_tooltip_end(ctx);
25398     }
```

```
25399 }
25400 #ifdef NK_INCLUDE_STANDARD_VARARGS
25401 NK_API void
25402 nk_tooltipf(struct nk_context *ctx, const char *fmt, ...)
25403 {
25404     va_list args;
25405     va_start(args, fmt);
25406     nk_tooltipfv(ctx, fmt, args);
25407     va_end(args);
25408 }
25409 NK_API void
25410 nk_tooltipfv(struct nk_context *ctx, const char *fmt, va_list args)
25411 {
25412     char buf[256];
25413     nk_strfmt(buf, NK_LEN(buf), fmt, args);
25414     nk_tooltip(ctx, buf);
25415 }
25416 #endif
25417
25418
25419
25420 #endif /* NK_IMPLEMENTATION */
25421
25422 /*
25465
25777 */
25778
```

27.10 nuklear_glfw_gl2.h

```
1 /*
2  * Nuklear - v1.32.0 - public domain
3  * no warrenty implied; use at your own risk.
4  * authored from 2015-2017 by Micha Mettke
5  */
6 /*
7  * ==============================================================
8  *
9  *                              API
10  *
11  * ===============================================================
12  */
13 #ifndef NK_GLFW_GL2_H_
14 #define NK_GLFW_GL2_H_
15
16 #include <GLFW/glfw3.h>
17
18 enum nk_glfw_init_state{
19     NK_GLFW3_DEFAULT = 0,
20     NK_GLFW3_INSTALL_CALLBACKS
21 };
22 NK_API struct nk_context*   nk_glfw3_init(GLFWwindow *win, enum nk_glfw_init_state);
23 NK_API void                 nk_glfw3_font_stash_begin(struct nk_font_atlas **atlas);
24 NK_API void                 nk_glfw3_font_stash_end(void);
25
26 NK_API void                 nk_glfw3_new_frame(void);
27 NK_API void                 nk_glfw3_render(enum nk_anti_aliasing);
28 NK_API void                 nk_glfw3_shutdown(void);
29
30 NK_API void                 nk_glfw3_char_callback(GLFWwindow *win, unsigned int codepoint);
31 NK_API void                 nk_gflw3_scroll_callback(GLFWwindow *win, double xoff, double yoff);
32
33 #endif
34
35 /*
36  * ==============================================================
37  *
38  *                          IMPLEMENTATION
39  *
40  * ===============================================================
41  */
42 #ifdef NK_GLFW_GL2_IMPLEMENTATION
43
44 #ifndef NK_GLFW_TEXT_MAX
45 #define NK_GLFW_TEXT_MAX 256
46 #endif
47 #ifndef NK_GLFW_DOUBLE_CLICK_LO
48 #define NK_GLFW_DOUBLE_CLICK_LO 0.02
49 #endif
50 #ifndef NK_GLFW_DOUBLE_CLICK_HI
51 #define NK_GLFW_DOUBLE_CLICK_HI 0.2
52 #endif
53
```

```
54 struct nk_glfw_device {
55     struct nk_buffer cmds;
56     struct nk_draw_null_texture null;
57     GLuint font_tex;
58 };
59
60 struct nk_glfw_vertex {
61     float position[2];
62     float uv[2];
63     nk_byte col[4];
64 };
65
66 static struct nk_glfw {
67     GLFWwindow *win;
68     int width, height;
69     int display_width, display_height;
70     struct nk_glfw_device ogl;
71     struct nk_context ctx;
72     struct nk_font_atlas atlas;
73     struct nk_vec2 fb_scale;
74     unsigned int text[NK_GLFW_TEXT_MAX];
75     int text_len;
76     struct nk_vec2 scroll;
77     double last_button_click;
78     int is_double_click_down;
79     struct nk_vec2 double_click_pos;
80 } glfw;
81
82 NK_INTERN void
83 nk_glfw3_device_upload_atlas(const void *image, int width, int height)
84 {
85     struct nk_glfw_device *dev = &glfw.ogl;
86     glGenTextures(1, &dev->font_tex);
87     glBindTexture(GL_TEXTURE_2D, dev->font_tex);
88     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
89     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
90     glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, (GLsizei)width, (GLsizei)height, 0,
91                 GL_RGBA, GL_UNSIGNED_BYTE, image);
92 }
93
94 NK_API void
95 nk_glfw3_render(enum nk_anti_aliasing AA)
96 {
97     /* setup global state */
98     struct nk_glfw_device *dev = &glfw.ogl;
99     glPushAttrib(GL_ENABLE_BIT|GL_COLOR_BUFFER_BIT|GL_TRANSFORM_BIT);
100     glDisable(GL_CULL_FACE);
101     glDisable(GL_DEPTH_TEST);
102     glEnable(GL_SCISSOR_TEST);
103     glEnable(GL_BLEND);
104     glEnable(GL_TEXTURE_2D);
105     glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
106
107     /* setup viewport/project */
108     glViewport(0,0,(GLsizei)glfw.display_width,(GLsizei)glfw.display_height);
109     glMatrixMode(GL_PROJECTION);
110     glPushMatrix();
111     glLoadIdentity();
112     glOrtho(0.0f, glfw.width, glfw.height, 0.0f, -1.0f, 1.0f);
113     glMatrixMode(GL_MODELVIEW);
114     glPushMatrix();
115     glLoadIdentity();
116
117     glEnableClientState(GL_VERTEX_ARRAY);
118     glEnableClientState(GL_TEXTURE_COORD_ARRAY);
119     glEnableClientState(GL_COLOR_ARRAY);
120     {
121         GLsizei vs = sizeof(struct nk_glfw_vertex);
122         size_t vp = offsetof(struct nk_glfw_vertex, position);
123         size_t vt = offsetof(struct nk_glfw_vertex, uv);
124         size_t vc = offsetof(struct nk_glfw_vertex, col);
125
126         /* convert from command queue into draw list and draw to screen */
127         const struct nk_draw_command *cmd;
128         const nk_draw_index *offset = NULL;
129         struct nk_buffer vbuf, ebuf;
130
131         /* fill convert configuration */
132         struct nk_convert_config config;
133         static const struct nk_draw_vertex_layout_element vertex_layout[] = {
134             {NK_VERTEX_POSITION, NK_FORMAT_FLOAT, NK_OFFSETOF(struct nk_glfw_vertex, position)},
135             {NK_VERTEX_TEXCOORD, NK_FORMAT_FLOAT, NK_OFFSETOF(struct nk_glfw_vertex, uv)},
136             {NK_VERTEX_COLOR, NK_FORMAT_R8G8B8A8, NK_OFFSETOF(struct nk_glfw_vertex, col)},
137             {NK_VERTEX_LAYOUT_END}
138         };
139         NK_MEMSET(&config, 0, sizeof(config));
140         config.vertex_layout = vertex_layout;
```

```
141         config.vertex_size = sizeof(struct nk_glfw_vertex);
142         config.vertex_alignment = NK_ALIGNOF(struct nk_glfw_vertex);
143         config.null = dev->null;
144         config.circle_segment_count = 22;
145         config.curve_segment_count = 22;
146         config.arc_segment_count = 22;
147         config.global_alpha = 1.0f;
148         config.shape_AA = AA;
149         config.line_AA = AA;
150 
151         /* convert shapes into vertexes */
152         nk_buffer_init_default(&vbuf);
153         nk_buffer_init_default(&ebuf);
154         nk_convert(&glfw.ctx, &dev->cmds, &vbuf, &ebuf, &config);
155 
156         /* setup vertex buffer pointer */
157         {const void *vertices = nk_buffer_memory_const(&vbuf);
158         glVertexPointer(2, GL_FLOAT, vs, (const void*)((const nk_byte*)vertices + vp));
159         glTexCoordPointer(2, GL_FLOAT, vs, (const void*)((const nk_byte*)vertices + vt));
160         glColorPointer(4, GL_UNSIGNED_BYTE, vs, (const void*)((const nk_byte*)vertices + vc));}
161 
162         /* iterate over and execute each draw command */
163         offset = (const nk_draw_index*)nk_buffer_memory_const(&ebuf);
164         nk_draw_foreach(cmd, &glfw.ctx, &dev->cmds)
165         {
166             if (!cmd->elem_count) continue;
167             glBindTexture(GL_TEXTURE_2D, (GLuint)cmd->texture.id);
168             glScissor(
169                 (GLint)(cmd->clip_rect.x * glfw.fb_scale.x),
170                 (GLint)((glfw.height - (GLint)(cmd->clip_rect.y + cmd->clip_rect.h)) * glfw.fb_scale.y),
171                 (GLint)(cmd->clip_rect.w * glfw.fb_scale.x),
172                 (GLint)(cmd->clip_rect.h * glfw.fb_scale.y));
173             glDrawElements(GL_TRIANGLES, (GLsizei)cmd->elem_count, GL_UNSIGNED_SHORT, offset);
174             offset += cmd->elem_count;
175         }
176         nk_clear(&glfw.ctx);
177         nk_buffer_free(&vbuf);
178         nk_buffer_free(&ebuf);
179     }
180 
181     /* default OpenGL state */
182     glDisableClientState(GL_VERTEX_ARRAY);
183     glDisableClientState(GL_TEXTURE_COORD_ARRAY);
184     glDisableClientState(GL_COLOR_ARRAY);
185 
186     glDisable(GL_CULL_FACE);
187     glDisable(GL_DEPTH_TEST);
188     glDisable(GL_SCISSOR_TEST);
189     glDisable(GL_BLEND);
190     glDisable(GL_TEXTURE_2D);
191 
192     glBindTexture(GL_TEXTURE_2D, 0);
193     glMatrixMode(GL_MODELVIEW);
194     glPopMatrix();
195     glMatrixMode(GL_PROJECTION);
196     glPopMatrix();
197     glPopAttrib();
198 }
199 
200 NK_API void
201 nk_glfw3_char_callback(GLFWwindow *win, unsigned int codepoint)
202 {
203     (void)win;
204     if (glfw.text_len < NK_GLFW_TEXT_MAX)
205         glfw.text[glfw.text_len++] = codepoint;
206 }
207 
208 NK_API void
209 nk_gflw3_scroll_callback(GLFWwindow *win, double xoff, double yoff)
210 {
211     (void)win; (void)xoff;
212     glfw.scroll.x += (float)xoff;
213     glfw.scroll.y += (float)yoff;
214 }
215 
216 NK_API void
217 nk_glfw3_mouse_button_callback(GLFWwindow* window, int button, int action, int mods)
218 {
219     double x, y;
220     if (button != GLFW_MOUSE_BUTTON_LEFT) return;
221     glfwGetCursorPos(window, &x, &y);
222     if (action == GLFW_PRESS)  {
223         double dt = glfwGetTime() - glfw.last_button_click;
224         if (dt > NK_GLFW_DOUBLE_CLICK_LO && dt < NK_GLFW_DOUBLE_CLICK_HI) {
225             glfw.is_double_click_down = nk_true;
226             glfw.double_click_pos = nk_vec2((float)x, (float)y);
227         }
```

```
228         glfw.last_button_click = glfwGetTime();
229     } else glfw.is_double_click_down = nk_false;
230 }
231
232 NK_INTERN void
233 nk_glfw3_clipboard_paste(nk_handle usr, struct nk_text_edit *edit)
234 {
235     const char *text = glfwGetClipboardString(glfw.win);
236     if (text) nk_textedit_paste(edit, text, nk_strlen(text));
237     (void)usr;
238 }
239
240 NK_INTERN void
241 nk_glfw3_clipboard_copy(nk_handle usr, const char *text, int len)
242 {
243     char *str = 0;
244     (void)usr;
245     if (!len) return;
246     str = (char*)malloc((size_t)len+1);
247     if (!str) return;
248     NK_MEMCPY(str, text, (size_t)len);
249     str[len] = '\0';
250     glfwSetClipboardString(glfw.win, str);
251     free(str);
252 }
253
254 NK_API struct nk_context*
255 nk_glfw3_init(GLFWwindow *win, enum nk_glfw_init_state init_state)
256 {
257     glfw.win = win;
258     if (init_state == NK_GLFW3_INSTALL_CALLBACKS) {
259         glfwSetScrollCallback(win, nk_gflw3_scroll_callback);
260         glfwSetCharCallback(win, nk_glfw3_char_callback);
261         glfwSetMouseButtonCallback(win, nk_glfw3_mouse_button_callback);
262     }
263     nk_init_default(&glfw.ctx, 0);
264     glfw.ctx.clip.copy = nk_glfw3_clipboard_copy;
265     glfw.ctx.clip.paste = nk_glfw3_clipboard_paste;
266     glfw.ctx.clip.userdata = nk_handle_ptr(0);
267     nk_buffer_init_default(&glfw.ogl.cmds);
268
269     glfw.is_double_click_down = nk_false;
270     glfw.double_click_pos = nk_vec2(0, 0);
271
272     return &glfw.ctx;
273 }
274
275 NK_API void
276 nk_glfw3_font_stash_begin(struct nk_font_atlas **atlas)
277 {
278     nk_font_atlas_init_default(&glfw.atlas);
279     nk_font_atlas_begin(&glfw.atlas);
280     *atlas = &glfw.atlas;
281 }
282
283 NK_API void
284 nk_glfw3_font_stash_end(void)
285 {
286     const void *image; int w, h;
287     image = nk_font_atlas_bake(&glfw.atlas, &w, &h, NK_FONT_ATLAS_RGBA32);
288     nk_glfw3_device_upload_atlas(image, w, h);
289     nk_font_atlas_end(&glfw.atlas, nk_handle_id((int)glfw.ogl.font_tex), &glfw.ogl.null);
290     if (glfw.atlas.default_font)
291         nk_style_set_font(&glfw.ctx, &glfw.atlas.default_font->handle);
292 }
293
294 NK_API void
295 nk_glfw3_new_frame(void)
296 {
297     int i;
298     double x, y;
299     struct nk_context *ctx = &glfw.ctx;
300     struct GLFWwindow *win = glfw.win;
301
302     glfwGetWindowSize(win, &glfw.width, &glfw.height);
303     glfwGetFramebufferSize(win, &glfw.display_width, &glfw.display_height);
304     glfw.fb_scale.x = (float)glfw.display_width/(float)glfw.width;
305     glfw.fb_scale.y = (float)glfw.display_height/(float)glfw.height;
306
307     nk_input_begin(ctx);
308     for (i = 0; i < glfw.text_len; ++i)
309         nk_input_unicode(ctx, glfw.text[i]);
310
311     /* optional grabbing behavior */
312     if (ctx->input.mouse.grab)
313         glfwSetInputMode(glfw.win, GLFW_CURSOR, GLFW_CURSOR_HIDDEN);
314     else if (ctx->input.mouse.ungrab)
```

```
315         glfwSetInputMode(glfw.win, GLFW_CURSOR, GLFW_CURSOR_NORMAL);
316
317     nk_input_key(ctx, NK_KEY_DEL, glfwGetKey(win, GLFW_KEY_DELETE) == GLFW_PRESS);
318     nk_input_key(ctx, NK_KEY_ENTER, glfwGetKey(win, GLFW_KEY_ENTER) == GLFW_PRESS);
319     nk_input_key(ctx, NK_KEY_TAB, glfwGetKey(win, GLFW_KEY_TAB) == GLFW_PRESS);
320     nk_input_key(ctx, NK_KEY_BACKSPACE, glfwGetKey(win, GLFW_KEY_BACKSPACE) == GLFW_PRESS);
321     nk_input_key(ctx, NK_KEY_UP, glfwGetKey(win, GLFW_KEY_UP) == GLFW_PRESS);
322     nk_input_key(ctx, NK_KEY_DOWN, glfwGetKey(win, GLFW_KEY_DOWN) == GLFW_PRESS);
323     nk_input_key(ctx, NK_KEY_TEXT_START, glfwGetKey(win, GLFW_KEY_HOME) == GLFW_PRESS);
324     nk_input_key(ctx, NK_KEY_TEXT_END, glfwGetKey(win, GLFW_KEY_END) == GLFW_PRESS);
325     nk_input_key(ctx, NK_KEY_SCROLL_START, glfwGetKey(win, GLFW_KEY_HOME) == GLFW_PRESS);
326     nk_input_key(ctx, NK_KEY_SCROLL_END, glfwGetKey(win, GLFW_KEY_END) == GLFW_PRESS);
327     nk_input_key(ctx, NK_KEY_SCROLL_DOWN, glfwGetKey(win, GLFW_KEY_PAGE_DOWN) == GLFW_PRESS);
328     nk_input_key(ctx, NK_KEY_SCROLL_UP, glfwGetKey(win, GLFW_KEY_PAGE_UP) == GLFW_PRESS);
329     nk_input_key(ctx, NK_KEY_SHIFT, glfwGetKey(win, GLFW_KEY_LEFT_SHIFT) == GLFW_PRESS||
330                                     glfwGetKey(win, GLFW_KEY_RIGHT_SHIFT) == GLFW_PRESS);
331
332     if (glfwGetKey(win, GLFW_KEY_LEFT_CONTROL) == GLFW_PRESS ||
333         glfwGetKey(win, GLFW_KEY_RIGHT_CONTROL) == GLFW_PRESS) {
334         nk_input_key(ctx, NK_KEY_COPY, glfwGetKey(win, GLFW_KEY_C) == GLFW_PRESS);
335         nk_input_key(ctx, NK_KEY_PASTE, glfwGetKey(win, GLFW_KEY_V) == GLFW_PRESS);
336         nk_input_key(ctx, NK_KEY_CUT, glfwGetKey(win, GLFW_KEY_X) == GLFW_PRESS);
337         nk_input_key(ctx, NK_KEY_TEXT_UNDO, glfwGetKey(win, GLFW_KEY_Z) == GLFW_PRESS);
338         nk_input_key(ctx, NK_KEY_TEXT_REDO, glfwGetKey(win, GLFW_KEY_R) == GLFW_PRESS);
339         nk_input_key(ctx, NK_KEY_TEXT_WORD_LEFT, glfwGetKey(win, GLFW_KEY_LEFT) == GLFW_PRESS);
340         nk_input_key(ctx, NK_KEY_TEXT_WORD_RIGHT, glfwGetKey(win, GLFW_KEY_RIGHT) == GLFW_PRESS);
341         nk_input_key(ctx, NK_KEY_TEXT_LINE_START, glfwGetKey(win, GLFW_KEY_B) == GLFW_PRESS);
342         nk_input_key(ctx, NK_KEY_TEXT_LINE_END, glfwGetKey(win, GLFW_KEY_E) == GLFW_PRESS);
343     } else {
344         nk_input_key(ctx, NK_KEY_LEFT, glfwGetKey(win, GLFW_KEY_LEFT) == GLFW_PRESS);
345         nk_input_key(ctx, NK_KEY_RIGHT, glfwGetKey(win, GLFW_KEY_RIGHT) == GLFW_PRESS);
346         nk_input_key(ctx, NK_KEY_COPY, 0);
347         nk_input_key(ctx, NK_KEY_PASTE, 0);
348         nk_input_key(ctx, NK_KEY_CUT, 0);
349         nk_input_key(ctx, NK_KEY_SHIFT, 0);
350     }
351
352     glfwGetCursorPos(win, &x, &y);
353     nk_input_motion(ctx, (int)x, (int)y);
354     if (ctx->input.mouse.grabbed) {
355         glfwSetCursorPos(glfw.win, (double)ctx->input.mouse.prev.x, (double)ctx->input.mouse.prev.y);
356         ctx->input.mouse.pos.x = ctx->input.mouse.prev.x;
357         ctx->input.mouse.pos.y = ctx->input.mouse.prev.y;
358     }
359
360     nk_input_button(ctx, NK_BUTTON_LEFT, (int)x, (int)y, glfwGetMouseButton(win, GLFW_MOUSE_BUTTON_LEFT)
    == GLFW_PRESS);
361     nk_input_button(ctx, NK_BUTTON_MIDDLE, (int)x, (int)y, glfwGetMouseButton(win,
    GLFW_MOUSE_BUTTON_MIDDLE) == GLFW_PRESS);
362     nk_input_button(ctx, NK_BUTTON_RIGHT, (int)x, (int)y, glfwGetMouseButton(win,
    GLFW_MOUSE_BUTTON_RIGHT) == GLFW_PRESS);
363     nk_input_button(ctx, NK_BUTTON_DOUBLE, (int)glfw.double_click_pos.x, (int)glfw.double_click_pos.y,
    glfw.is_double_click_down);
364     nk_input_scroll(ctx, glfw.scroll);
365     nk_input_end(&glfw.ctx);
366     glfw.text_len = 0;
367     glfw.scroll = nk_vec2(0,0);
368 }
369
370 NK_API
371 void nk_glfw3_shutdown(void)
372 {
373     struct nk_glfw_device *dev = &glfw.ogl;
374     nk_font_atlas_clear(&glfw.atlas);
375     nk_free(&glfw.ctx);
376     glDeleteTextures(1, &dev->font_tex);
377     nk_buffer_free(&dev->cmds);
378     NK_MEMSET(&glfw, 0, sizeof(glfw));
379 }
380
381 #endif
```

27.11 stb_image_write.h

```
1 /* stb_image_write - v1.02 - public domain - http://nothings.org/stb/stb_image_write.h
2    writes out PNG/BMP/TGA images to C stdio - Sean Barrett 2010-2015
3                                      no warranty implied; use at your own risk
4
5    Before #including,
6
7        #define STB_IMAGE_WRITE_IMPLEMENTATION
8
9    in the file that you want to have the implementation.
```

```
10
11    Will probably not work correctly with strict-aliasing optimizations.
12
13 ABOUT:
14
15    This header file is a library for writing images to C stdio. It could be
16    adapted to write to memory or a general streaming interface; let me know.
17
18    The PNG output is not optimal; it is 20-50% larger than the file
19    written by a decent optimizing implementation. This library is designed
20    for source code compactness and simplicity, not optimal image file size
21    or run-time performance.
22
23 BUILDING:
24
25    You can #define STBIW_ASSERT(x) before the #include to avoid using assert.h.
26    You can #define STBIW_MALLOC(), STBIW_REALLOC(), and STBIW_FREE() to replace
27    malloc,realloc,free.
28    You can define STBIW_MEMMOVE() to replace memmove()
29
30 USAGE:
31
32    There are four functions, one for each image file format:
33
34      int stbi_write_png(char const *filename, int w, int h, int comp, const void *data, int
       stride_in_bytes);
35      int stbi_write_bmp(char const *filename, int w, int h, int comp, const void *data);
36      int stbi_write_tga(char const *filename, int w, int h, int comp, const void *data);
37      int stbi_write_hdr(char const *filename, int w, int h, int comp, const float *data);
38
39    There are also four equivalent functions that use an arbitrary write function. You are
40    expected to open/close your file-equivalent before and after calling these:
41
42      int stbi_write_png_to_func(stbi_write_func *func, void *context, int w, int h, int comp, const void
       *data, int stride_in_bytes);
43      int stbi_write_bmp_to_func(stbi_write_func *func, void *context, int w, int h, int comp, const void
       *data);
44      int stbi_write_tga_to_func(stbi_write_func *func, void *context, int w, int h, int comp, const void
       *data);
45      int stbi_write_hdr_to_func(stbi_write_func *func, void *context, int w, int h, int comp, const float
       *data);
46
47    where the callback is:
48       void stbi_write_func(void *context, void *data, int size);
49
50    You can define STBI_WRITE_NO_STDIO to disable the file variant of these
51    functions, so the library will not use stdio.h at all. However, this will
52    also disable HDR writing, because it requires stdio for formatted output.
53
54    Each function returns 0 on failure and non-0 on success.
55
56    The functions create an image file defined by the parameters. The image
57    is a rectangle of pixels stored from left-to-right, top-to-bottom.
58    Each pixel contains 'comp' channels of data stored interleaved with 8-bits
59    per channel, in the following order: 1=Y, 2=YA, 3=RGB, 4=RGBA. (Y is
60    monochrome color.) The rectangle is 'w' pixels wide and 'h' pixels tall.
61    The *data pointer points to the first byte of the top-left-most pixel.
62    For PNG, "stride_in_bytes" is the distance in bytes from the first byte of
63    a row of pixels to the first byte of the next row of pixels.
64
65    PNG creates output files with the same number of components as the input.
66    The BMP format expands Y to RGB in the file format and does not
67    output alpha.
68
69    PNG supports writing rectangles of data even when the bytes storing rows of
70    data are not consecutive in memory (e.g. sub-rectangles of a larger image),
71    by supplying the stride between the beginning of adjacent rows. The other
72    formats do not. (Thus you cannot write a native-format BMP through the BMP
73    writer, both because it is in BGR order and because it may have padding
74    at the end of the line.)
75
76    HDR expects linear float data. Since the format is always 32-bit rgb(e)
77    data, alpha (if provided) is discarded, and for monochrome data it is
78    replicated across all three channels.
79
80    TGA supports RLE or non-RLE compressed data. To use non-RLE-compressed
81    data, set the global variable 'stbi_write_tga_with_rle' to 0.
82
83 CREDITS:
84
85    PNG/BMP/TGA
86       Sean Barrett
87    HDR
88       Baldur Karlsson
89    TGA monochrome:
90       Jean-Sebastien Guay
91    misc enhancements:
```

```
92       Tim Kelsey
93    TGA RLE
94       Alan Hickman
95    initial file IO callback implementation
96       Emmanuel Julien
97    bugfixes:
98       github:Chribba
99       Guillaume Chereau
100       github:jry2
101       github:romigrou
102       Sergio Gonzalez
103       Jonas Karlsson
104       Filip Wasil
105       Thatcher Ulrich
106
107 LICENSE
108
109 This software is dual-licensed to the public domain and under the following
110 license: you are granted a perpetual, irrevocable license to copy, modify,
111 publish, and distribute this file as you see fit.
112
113 */
114
115 #ifndef INCLUDE_STB_IMAGE_WRITE_H
116 #define INCLUDE_STB_IMAGE_WRITE_H
117
118 #ifdef __cplusplus
119 extern "C" {
120 #endif
121
122 #ifdef STB_IMAGE_WRITE_STATIC
123 #define STBIWDEF static
124 #else
125 #define STBIWDEF extern
126 extern int stbi_write_tga_with_rle;
127 #endif
128
129 #ifndef STBI_WRITE_NO_STDIO
130 STBIWDEF int stbi_write_png(char const *filename, int w, int h, int comp, const void  *data, int
       stride_in_bytes);
131 STBIWDEF int stbi_write_bmp(char const *filename, int w, int h, int comp, const void  *data);
132 STBIWDEF int stbi_write_tga(char const *filename, int w, int h, int comp, const void  *data);
133 STBIWDEF int stbi_write_hdr(char const *filename, int w, int h, int comp, const float *data);
134 #endif
135
136 typedef void stbi_write_func(void *context, void *data, int size);
137
138 STBIWDEF int stbi_write_png_to_func(stbi_write_func *func, void *context, int w, int h, int comp, const
       void  *data, int stride_in_bytes);
139 STBIWDEF int stbi_write_bmp_to_func(stbi_write_func *func, void *context, int w, int h, int comp, const
       void  *data);
140 STBIWDEF int stbi_write_tga_to_func(stbi_write_func *func, void *context, int w, int h, int comp, const
       void  *data);
141 STBIWDEF int stbi_write_hdr_to_func(stbi_write_func *func, void *context, int w, int h, int comp, const
       float *data);
142
143 #ifdef __cplusplus
144 }
145 #endif
146
147 #endif//INCLUDE_STB_IMAGE_WRITE_H
148
149 #ifdef STB_IMAGE_WRITE_IMPLEMENTATION
150
151 #ifdef _WIN32
152    #ifndef _CRT_SECURE_NO_WARNINGS
153    #define _CRT_SECURE_NO_WARNINGS
154    #endif
155    #ifndef _CRT_NONSTDC_NO_DEPRECATE
156    #define _CRT_NONSTDC_NO_DEPRECATE
157    #endif
158 #endif
159
160 #ifndef STBI_WRITE_NO_STDIO
161 #include <stdio.h>
162 #endif // STBI_WRITE_NO_STDIO
163
164 #include <stdarg.h>
165 #include <stdlib.h>
166 #include <string.h>
167 #include <math.h>
168
169 #if defined(STBIW_MALLOC) && defined(STBIW_FREE) && (defined(STBIW_REALLOC) ||
       defined(STBIW_REALLOC_SIZED))
170 // ok
171 #elif !defined(STBIW_MALLOC) && !defined(STBIW_FREE) && !defined(STBIW_REALLOC) &&
       !defined(STBIW_REALLOC_SIZED)
```

```
172 // ok
173 #else
174 #error "Must define all or none of STBIW_MALLOC, STBIW_FREE, and STBIW_REALLOC (or
       STBIW_REALLOC_SIZED)."
175 #endif
176
177 #ifndef STBIW_MALLOC
178 #define STBIW_MALLOC(sz)        malloc(sz)
179 #define STBIW_REALLOC(p,newsz)  realloc(p,newsz)
180 #define STBIW_FREE(p)           free(p)
181 #endif
182
183 #ifndef STBIW_REALLOC_SIZED
184 #define STBIW_REALLOC_SIZED(p,oldsz,newsz) STBIW_REALLOC(p,newsz)
185 #endif
186
187
188 #ifndef STBIW_MEMMOVE
189 #define STBIW_MEMMOVE(a,b,sz) memmove(a,b,sz)
190 #endif
191
192
193 #ifndef STBIW_ASSERT
194 #include <assert.h>
195 #define STBIW_ASSERT(x) assert(x)
196 #endif
197
198 #define STBIW_UCHAR(x) (unsigned char) ((x) & 0xff)
199
200 typedef struct
201 {
202    stbi_write_func *func;
203    void *context;
204 } stbi__write_context;
205
206 // initialize a callback-based context
207 static void stbi__start_write_callbacks(stbi__write_context *s, stbi_write_func *c, void *context)
208 {
209    s->func    = c;
210    s->context = context;
211 }
212
213 #ifndef STBI_WRITE_NO_STDIO
214
215 static void stbi__stdio_write(void *context, void *data, int size)
216 {
217    fwrite(data,1,size,(FILE*) context);
218 }
219
220 static int stbi__start_write_file(stbi__write_context *s, const char *filename)
221 {
222    FILE *f = fopen(filename, "wb");
223    stbi__start_write_callbacks(s, stbi__stdio_write, (void *) f);
224    return f != NULL;
225 }
226
227 static void stbi__end_write_file(stbi__write_context *s)
228 {
229    fclose((FILE *)s->context);
230 }
231
232 #endif // !STBI_WRITE_NO_STDIO
233
234 typedef unsigned int stbiw_uint32;
235 typedef int stb_image_write_test[sizeof(stbiw_uint32)==4 ? 1 : -1];
236
237 #ifdef STB_IMAGE_WRITE_STATIC
238 static int stbi_write_tga_with_rle = 1;
239 #else
240 int stbi_write_tga_with_rle = 1;
241 #endif
242
243 static void stbiw__writefv(stbi__write_context *s, const char *fmt, va_list v)
244 {
245    while (*fmt) {
246       switch (*fmt++) {
247          case ' ': break;
248          case '1': { unsigned char x = STBIW_UCHAR(va_arg(v, int));
249                      s->func(s->context,&x,1);
250                      break; }
251          case '2': { int x = va_arg(v,int);
252                      unsigned char b[2];
253                      b[0] = STBIW_UCHAR(x);
254                      b[1] = STBIW_UCHAR(x>>8);
255                      s->func(s->context,b,2);
256                      break; }
257          case '4': { stbiw_uint32 x = va_arg(v,int);
```

```
258                     unsigned char b[4];
259                     b[0]=STBIW_UCHAR(x);
260                     b[1]=STBIW_UCHAR(x>>8);
261                     b[2]=STBIW_UCHAR(x>>16);
262                     b[3]=STBIW_UCHAR(x>>24);
263                     s->func(s->context,b,4);
264                     break; }
265          default:
266             STBIW_ASSERT(0);
267             return;
268       }
269    }
270 }
271
272 static void stbiw__writef(stbi__write_context *s, const char *fmt, ...)
273 {
274    va_list v;
275    va_start(v, fmt);
276    stbiw__writefv(s, fmt, v);
277    va_end(v);
278 }
279
280 static void stbiw__write3(stbi__write_context *s, unsigned char a, unsigned char b, unsigned char c)
281 {
282    unsigned char arr[3];
283    arr[0] = a, arr[1] = b, arr[2] = c;
284    s->func(s->context, arr, 3);
285 }
286
287 static void stbiw__write_pixel(stbi__write_context *s, int rgb_dir, int comp, int write_alpha, int
       expand_mono, unsigned char *d)
288 {
289    unsigned char bg[3] = { 255, 0, 255}, px[3];
290    int k;
291
292    if (write_alpha < 0)
293       s->func(s->context, &d[comp - 1], 1);
294
295    switch (comp) {
296       case 1:
297          s->func(s->context,d,1);
298          break;
299       case 2:
300          if (expand_mono)
301             stbiw__write3(s, d[0], d[0], d[0]); // monochrome bmp
302          else
303             s->func(s->context, d, 1);  // monochrome TGA
304          break;
305       case 4:
306          if (!write_alpha) {
307             // composite against pink background
308             for (k = 0; k < 3; ++k)
309                px[k] = bg[k] + ((d[k] - bg[k]) * d[3]) / 255;
310             stbiw__write3(s, px[1 - rgb_dir], px[1], px[1 + rgb_dir]);
311             break;
312          }
313          /* FALLTHROUGH */
314       case 3:
315          stbiw__write3(s, d[1 - rgb_dir], d[1], d[1 + rgb_dir]);
316          break;
317    }
318    if (write_alpha > 0)
319       s->func(s->context, &d[comp - 1], 1);
320 }
321
322 static void stbiw__write_pixels(stbi__write_context *s, int rgb_dir, int vdir, int x, int y, int comp,
       void *data, int write_alpha, int scanline_pad, int expand_mono)
323 {
324    stbiw_uint32 zero = 0;
325    int i,j, j_end;
326
327    if (y <= 0)
328       return;
329
330    if (vdir < 0)
331       j_end = -1, j = y-1;
332    else
333       j_end =  y, j = 0;
334
335    for (; j != j_end; j += vdir) {
336       for (i=0; i < x; ++i) {
337          unsigned char *d = (unsigned char *) data + (j*x+i)*comp;
338          stbiw__write_pixel(s, rgb_dir, comp, write_alpha, expand_mono, d);
339       }
340       s->func(s->context, &zero, scanline_pad);
341    }
342 }
```

```
343
344 static int stbiw__outfile(stbi__write_context *s, int rgb_dir, int vdir, int x, int y, int comp, int
    expand_mono, void *data, int alpha, int pad, const char *fmt, ...)
345 {
346    if (y < 0 || x < 0) {
347       return 0;
348    } else {
349       va_list v;
350       va_start(v, fmt);
351       stbiw__writefv(s, fmt, v);
352       va_end(v);
353       stbiw__write_pixels(s,rgb_dir,vdir,x,y,comp,data,alpha,pad, expand_mono);
354       return 1;
355    }
356 }
357
358 static int stbi_write_bmp_core(stbi__write_context *s, int x, int y, int comp, const void *data)
359 {
360    int pad = (-x*3) & 3;
361    return stbiw__outfile(s,-1,-1,x,y,comp,1,(void *) data,0,pad,
362            "11 4 22 4" "4 44 22 444444",
363            'B', 'M', 14+40+(x*3+pad)*y, 0,0, 14+40,  // file header
364             40, x,y, 1,24, 0,0,0,0,0,0);             // bitmap header
365 }
366
367 STBIWDEF int stbi_write_bmp_to_func(stbi_write_func *func, void *context, int x, int y, int comp, const
    void *data)
368 {
369    stbi__write_context s;
370    stbi__start_write_callbacks(&s, func, context);
371    return stbi_write_bmp_core(&s, x, y, comp, data);
372 }
373
374 #ifndef STBI_WRITE_NO_STDIO
375 STBIWDEF int stbi_write_bmp(char const *filename, int x, int y, int comp, const void *data)
376 {
377    stbi__write_context s;
378    if (stbi__start_write_file(&s,filename)) {
379       int r = stbi_write_bmp_core(&s, x, y, comp, data);
380       stbi__end_write_file(&s);
381       return r;
382    } else
383       return 0;
384 }
385 #endif
386
387 static int stbi_write_tga_core(stbi__write_context *s, int x, int y, int comp, void *data)
388 {
389    int has_alpha = (comp == 2 || comp == 4);
390    int colorbytes = has_alpha ? comp-1 : comp;
391    int format = colorbytes < 2 ? 3 : 2; // 3 color channels (RGB/RGBA) = 2, 1 color channel (Y/YA) = 3
392
393    if (y < 0 || x < 0)
394       return 0;
395
396    if (!stbi_write_tga_with_rle) {
397       return stbiw__outfile(s, -1, -1, x, y, comp, 0, (void *) data, has_alpha, 0,
398          "111 221 2222 11", 0, 0, format, 0, 0, 0, 0, 0, x, y, (colorbytes + has_alpha) * 8, has_alpha *
    8);
399    } else {
400       int i,j,k;
401
402       stbiw__writef(s, "111 221 2222 11", 0,0,format+8, 0,0,0, 0,0,x,y, (colorbytes + has_alpha) * 8,
    has_alpha * 8);
403
404       for (j = y - 1; j >= 0; --j) {
405           unsigned char *row = (unsigned char *) data + j * x * comp;
406          int len;
407
408          for (i = 0; i < x; i += len) {
409             unsigned char *begin = row + i * comp;
410             int diff = 1;
411             len = 1;
412
413             if (i < x - 1) {
414                ++len;
415                diff = memcmp(begin, row + (i + 1) * comp, comp);
416                if (diff) {
417                   const unsigned char *prev = begin;
418                   for (k = i + 2; k < x && len < 128; ++k) {
419                      if (memcmp(prev, row + k * comp, comp)) {
420                         prev += comp;
421                         ++len;
422                      } else {
423                         --len;
424                         break;
425                      }
```

```
426                  }
427               } else {
428                  for (k = i + 2; k < x && len < 128; ++k) {
429                     if (!memcmp(begin, row + k * comp, comp)) {
430                        ++len;
431                     } else {
432                        break;
433                     }
434                  }
435               }
436            }
437
438            if (diff) {
439               unsigned char header = STBIW_UCHAR(len - 1);
440               s->func(s->context, &header, 1);
441               for (k = 0; k < len; ++k) {
442                  stbiw__write_pixel(s, -1, comp, has_alpha, 0, begin + k * comp);
443               }
444            } else {
445               unsigned char header = STBIW_UCHAR(len - 129);
446               s->func(s->context, &header, 1);
447               stbiw__write_pixel(s, -1, comp, has_alpha, 0, begin);
448            }
449         }
450      }
451   }
452   return 1;
453 }
454
455 int stbi_write_tga_to_func(stbi_write_func *func, void *context, int x, int y, int comp, const void
       *data)
456 {
457    stbi__write_context s;
458    stbi__start_write_callbacks(&s, func, context);
459    return stbi_write_tga_core(&s, x, y, comp, (void *) data);
460 }
461
462 #ifndef STBI_WRITE_NO_STDIO
463 int stbi_write_tga(char const *filename, int x, int y, int comp, const void *data)
464 {
465    stbi__write_context s;
466    if (stbi__start_write_file(&s,filename)) {
467       int r = stbi_write_tga_core(&s, x, y, comp, (void *) data);
468       stbi__end_write_file(&s);
469       return r;
470    } else
471       return 0;
472 }
473 #endif
474
475 // *************************************************************************************************
476 // Radiance RGBE HDR writer
477 // by Baldur Karlsson
478 #ifndef STBI_WRITE_NO_STDIO
479
480 #define stbiw__max(a, b)  ((a) > (b) ? (a) : (b))
481
482 void stbiw__linear_to_rgbe(unsigned char *rgbe, float *linear)
483 {
484    int exponent;
485    float maxcomp = stbiw__max(linear[0], stbiw__max(linear[1], linear[2]));
486
487    if (maxcomp < 1e-32f) {
488       rgbe[0] = rgbe[1] = rgbe[2] = rgbe[3] = 0;
489    } else {
490       float normalize = (float) frexp(maxcomp, &exponent) * 256.0f/maxcomp;
491
492       rgbe[0] = (unsigned char)(linear[0] * normalize);
493       rgbe[1] = (unsigned char)(linear[1] * normalize);
494       rgbe[2] = (unsigned char)(linear[2] * normalize);
495       rgbe[3] = (unsigned char)(exponent + 128);
496    }
497 }
498
499 void stbiw__write_run_data(stbi__write_context *s, int length, unsigned char databyte)
500 {
501    unsigned char lengthbyte = STBIW_UCHAR(length+128);
502    STBIW_ASSERT(length+128 <= 255);
503    s->func(s->context, &lengthbyte, 1);
504    s->func(s->context, &databyte, 1);
505 }
506
507 void stbiw__write_dump_data(stbi__write_context *s, int length, unsigned char *data)
508 {
509    unsigned char lengthbyte = STBIW_UCHAR(length);
510    STBIW_ASSERT(length <= 128); // inconsistent with spec but consistent with official code
511    s->func(s->context, &lengthbyte, 1);
```

```
512    s->func(s->context, data, length);
513 }
514
515 void stbiw__write_hdr_scanline(stbi__write_context *s, int width, int ncomp, unsigned char *scratch,
    float *scanline)
516 {
517    unsigned char scanlineheader[4] = { 2, 2, 0, 0 };
518    unsigned char rgbe[4];
519    float linear[3];
520    int x;
521
522    scanlineheader[2] = (width&0xff00)»8;
523    scanlineheader[3] = (width&0x00ff);
524
525    /* skip RLE for images too small or large */
526    if (width < 8 || width >= 32768) {
527       for (x=0; x < width; x++) {
528          switch (ncomp) {
529             case 4: /* fallthrough */
530             case 3: linear[2] = scanline[x*ncomp + 2];
531                     linear[1] = scanline[x*ncomp + 1];
532                     linear[0] = scanline[x*ncomp + 0];
533                     break;
534             default:
535                     linear[0] = linear[1] = linear[2] = scanline[x*ncomp + 0];
536                     break;
537          }
538          stbiw__linear_to_rgbe(rgbe, linear);
539          s->func(s->context, rgbe, 4);
540       }
541    } else {
542       int c,r;
543       /* encode into scratch buffer */
544       for (x=0; x < width; x++) {
545          switch(ncomp) {
546             case 4: /* fallthrough */
547             case 3: linear[2] = scanline[x*ncomp + 2];
548                     linear[1] = scanline[x*ncomp + 1];
549                     linear[0] = scanline[x*ncomp + 0];
550                     break;
551             default:
552                     linear[0] = linear[1] = linear[2] = scanline[x*ncomp + 0];
553                     break;
554          }
555          stbiw__linear_to_rgbe(rgbe, linear);
556          scratch[x + width*0] = rgbe[0];
557          scratch[x + width*1] = rgbe[1];
558          scratch[x + width*2] = rgbe[2];
559          scratch[x + width*3] = rgbe[3];
560       }
561
562       s->func(s->context, scanlineheader, 4);
563
564       /* RLE each component separately */
565       for (c=0; c < 4; c++) {
566          unsigned char *comp = &scratch[width*c];
567
568          x = 0;
569          while (x < width) {
570             // find first run
571             r = x;
572             while (r+2 < width) {
573                if (comp[r] == comp[r+1] && comp[r] == comp[r+2])
574                   break;
575                ++r;
576             }
577             if (r+2 >= width)
578                r = width;
579             // dump up to first run
580             while (x < r) {
581                int len = r-x;
582                if (len > 128) len = 128;
583                stbiw__write_dump_data(s, len, &comp[x]);
584                x += len;
585             }
586             // if there's a run, output it
587             if (r+2 < width) { // same test as what we break out of in search loop, so only true if we
    break'd
588                // find next byte after run
589                while (r < width && comp[r] == comp[x])
590                   ++r;
591                // output run up to r
592                while (x < r) {
593                   int len = r-x;
594                   if (len > 127) len = 127;
595                   stbiw__write_run_data(s, len, comp[x]);
596                   x += len;
```

```
597                }
598             }
599          }
600       }
601    }
602 }
603
604 static int stbi_write_hdr_core(stbi__write_context *s, int x, int y, int comp, float *data)
605 {
606    if (y <= 0 || x <= 0 || data == NULL)
607       return 0;
608    else {
609       // Each component is stored separately. Allocate scratch space for full output scanline.
610       unsigned char *scratch = (unsigned char *) STBIW_MALLOC(x*4);
611       int i, len;
612       char buffer[128];
613       char header[] = "#?RADIANCE\n# Written by stb_image_write.h\nFORMAT=32-bit_rle_rgbe\n";
614       s->func(s->context, header, sizeof(header)-1);
615
616       len = sprintf(buffer, "EXPOSURE=          1.0000000000000\n\n-Y %d +X %d\n", y, x);
617       s->func(s->context, buffer, len);
618
619       for(i=0; i < y; i++)
620          stbiw__write_hdr_scanline(s, x, comp, scratch, data + comp*i*x);
621       STBIW_FREE(scratch);
622       return 1;
623    }
624 }
625
626 int stbi_write_hdr_to_func(stbi_write_func *func, void *context, int x, int y, int comp, const float
      *data)
627 {
628    stbi__write_context s;
629    stbi__start_write_callbacks(&s, func, context);
630    return stbi_write_hdr_core(&s, x, y, comp, (float *) data);
631 }
632
633 int stbi_write_hdr(char const *filename, int x, int y, int comp, const float *data)
634 {
635    stbi__write_context s;
636    if (stbi__start_write_file(&s,filename)) {
637       int r = stbi_write_hdr_core(&s, x, y, comp, (float *) data);
638       stbi__end_write_file(&s);
639       return r;
640    } else
641       return 0;
642 }
643 #endif // STBI_WRITE_NO_STDIO
644
645
647 //
648 // PNG writer
649 //
650
651 // stretchy buffer; stbiw__sbpush() == vector<>::push_back() -- stbiw__sbcount() == vector<>::size()
652 #define stbiw__sbraw(a) ((int *) (a) - 2)
653 #define stbiw__sbm(a)   stbiw__sbraw(a)[0]
654 #define stbiw__sbn(a)   stbiw__sbraw(a)[1]
655
656 #define stbiw__sbneedgrow(a,n)  ((a)==0 || stbiw__sbn(a)+n >= stbiw__sbm(a))
657 #define stbiw__sbmaybegrow(a,n) (stbiw__sbneedgrow(a,(n)) ? stbiw__sbgrow(a,n) : 0)
658 #define stbiw__sbgrow(a,n)  stbiw__sbgrowf((void **) &(a), (n), sizeof(*(a)))
659
660 #define stbiw__sbpush(a, v)      (stbiw__sbmaybegrow(a,1), (a)[stbiw__sbn(a)++] = (v))
661 #define stbiw__sbcount(a)        ((a) ? stbiw__sbn(a) : 0)
662 #define stbiw__sbfree(a)         ((a) ? STBIW_FREE(stbiw__sbraw(a)),0 : 0)
663
664 static void *stbiw__sbgrowf(void **arr, int increment, int itemsize)
665 {
666    int m = *arr ? 2*stbiw__sbm(*arr)+increment : increment+1;
667    void *p = STBIW_REALLOC_SIZED(*arr ? stbiw__sbraw(*arr) : 0, *arr ? (stbiw__sbm(*arr)*itemsize +
      sizeof(int)*2) : 0, itemsize * m + sizeof(int)*2);
668    STBIW_ASSERT(p);
669    if (p) {
670       if (!*arr) ((int *) p)[1] = 0;
671       *arr = (void *) ((int *) p + 2);
672       stbiw__sbm(*arr) = m;
673    }
674    return *arr;
675 }
676
677 static unsigned char *stbiw__zlib_flushf(unsigned char *data, unsigned int *bitbuffer, int *bitcount)
678 {
679    while (*bitcount >= 8) {
680       stbiw__sbpush(data, STBIW_UCHAR(*bitbuffer));
681       *bitbuffer >>= 8;
682       *bitcount -= 8;
```

```
683    }
684    return data;
685 }
686
687 static int stbiw__zlib_bitrev(int code, int codebits)
688 {
689    int res=0;
690    while (codebits--) {
691       res = (res « 1) | (code & 1);
692       code »= 1;
693    }
694    return res;
695 }
696
697 static unsigned int stbiw__zlib_countm(unsigned char *a, unsigned char *b, int limit)
698 {
699    int i;
700    for (i=0; i < limit && i < 258; ++i)
701       if (a[i] != b[i]) break;
702    return i;
703 }
704
705 static unsigned int stbiw__zhash(unsigned char *data)
706 {
707    stbiw_uint32 hash = data[0] + (data[1] « 8) + (data[2] « 16);
708    hash ^= hash « 3;
709    hash += hash » 5;
710    hash ^= hash « 4;
711    hash += hash » 17;
712    hash ^= hash « 25;
713    hash += hash » 6;
714    return hash;
715 }
716
717 #define stbiw__zlib_flush() (out = stbiw__zlib_flushf(out, &bitbuf, &bitcount))
718 #define stbiw__zlib_add(code,codebits) \
719       (bitbuf |= (code) « bitcount, bitcount += (codebits), stbiw__zlib_flush())
720 #define stbiw__zlib_huffa(b,c)  stbiw__zlib_add(stbiw__zlib_bitrev(b,c),c)
721 // default huffman tables
722 #define stbiw__zlib_huff1(n)  stbiw__zlib_huffa(0x30 + (n), 8)
723 #define stbiw__zlib_huff2(n)  stbiw__zlib_huffa(0x190 + (n)-144, 9)
724 #define stbiw__zlib_huff3(n)  stbiw__zlib_huffa(0 + (n)-256,7)
725 #define stbiw__zlib_huff4(n)  stbiw__zlib_huffa(0xc0 + (n)-280,8)
726 #define stbiw__zlib_huff(n)  ((n) <= 143 ? stbiw__zlib_huff1(n) : (n) <= 255 ? stbiw__zlib_huff2(n) :
      (n) <= 279 ? stbiw__zlib_huff3(n) : stbiw__zlib_huff4(n))
727 #define stbiw__zlib_huffb(n) ((n) <= 143 ? stbiw__zlib_huff1(n) : stbiw__zlib_huff2(n))
728
729 #define stbiw__ZHASH   16384
730
731 unsigned char * stbi_zlib_compress(unsigned char *data, int data_len, int *out_len, int quality)
732 {
733    static unsigned short lengthc[] = {
      3,4,5,6,7,8,9,10,11,13,15,17,19,23,27,31,35,43,51,59,67,83,99,115,131,163,195,227,258, 259 };
734    static unsigned char  lengtheb[]= { 0,0,0,0,0,0,0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4,
      4,  5,  5,  5,  5,  0 };
735    static unsigned short distc[]   = {
      1,2,3,4,5,7,9,13,17,25,33,49,65,97,129,193,257,385,513,769,1025,1537,2049,3073,4097,6145,8193,12289,16385,24577,
      32768 };
736    static unsigned char  disteb[]  = {
      0,0,0,0,1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,8,9,9,10,10,11,11,12,12,13,13 };
737    unsigned int bitbuf=0;
738    int i,j, bitcount=0;
739    unsigned char *out = NULL;
740    unsigned char ***hash_table = (unsigned char***) STBIW_MALLOC(stbiw__ZHASH * sizeof(char**));
741    if (quality < 5) quality = 5;
742
743    stbiw__sbpush(out, 0x78);   // DEFLATE 32K window
744    stbiw__sbpush(out, 0x5e);   // FLEVEL = 1
745    stbiw__zlib_add(1,1);  // BFINAL = 1
746    stbiw__zlib_add(1,2);  // BTYPE = 1 -- fixed huffman
747
748    for (i=0; i < stbiw__ZHASH; ++i)
749       hash_table[i] = NULL;
750
751    i=0;
752    while (i < data_len-3) {
753       // hash next 3 bytes of data to be compressed
754       int h = stbiw__zhash(data+i)&(stbiw__ZHASH-1), best=3;
755       unsigned char *bestloc = 0;
756       unsigned char **hlist = hash_table[h];
757       int n = stbiw__sbcount(hlist);
758       for (j=0; j < n; ++j) {
759          if (hlist[j]-data > i-32768) { // if entry lies within window
760             int d = stbiw__zlib_countm(hlist[j], data+i, data_len-i);
761             if (d >= best) best=d,bestloc=hlist[j];
762          }
763       }
```

```
764         // when hash table entry is too long, delete half the entries
765         if (hash_table[h] && stbiw__sbn(hash_table[h]) == 2*quality) {
766            STBIW_MEMMOVE(hash_table[h], hash_table[h]+quality, sizeof(hash_table[h][0])*quality);
767            stbiw__sbn(hash_table[h]) = quality;
768         }
769         stbiw__sbpush(hash_table[h],data+i);
770
771         if (bestloc) {
772            // "lazy matching" - check match at *next* byte, and if it's better, do cur byte as literal
773            h = stbiw__zhash(data+i+1)&(stbiw__ZHASH-1);
774            hlist = hash_table[h];
775            n = stbiw__sbcount(hlist);
776            for (j=0; j < n; ++j) {
777               if (hlist[j]-data > i-32767) {
778                  int e = stbiw__zlib_countm(hlist[j], data+i+1, data_len-i-1);
779                  if (e > best) { // if next match is better, bail on current match
780                     bestloc = NULL;
781                     break;
782                  }
783               }
784            }
785         }
786
787         if (bestloc) {
788            int d = (int) (data+i - bestloc); // distance back
789            STBIW_ASSERT(d <= 32767 && best <= 258);
790            for (j=0; best > lengthc[j+1]-1; ++j);
791            stbiw__zlib_huff(j+257);
792            if (lengtheb[j]) stbiw__zlib_add(best - lengthc[j], lengtheb[j]);
793            for (j=0; d > distc[j+1]-1; ++j);
794            stbiw__zlib_add(stbiw__zlib_bitrev(j,5),5);
795            if (disteb[j]) stbiw__zlib_add(d - distc[j], disteb[j]);
796            i += best;
797         } else {
798            stbiw__zlib_huffb(data[i]);
799            ++i;
800         }
801      }
802      // write out final bytes
803      for (;i < data_len; ++i)
804         stbiw__zlib_huffb(data[i]);
805      stbiw__zlib_huff(256); // end of block
806      // pad with 0 bits to byte boundary
807      while (bitcount)
808         stbiw__zlib_add(0,1);
809
810      for (i=0; i < stbiw__ZHASH; ++i)
811         (void) stbiw__sbfree(hash_table[i]);
812      STBIW_FREE(hash_table);
813
814      {
815         // compute adler32 on input
816         unsigned int s1=1, s2=0;
817         int blocklen = (int) (data_len % 5552);
818         j=0;
819         while (j < data_len) {
820            for (i=0; i < blocklen; ++i) s1 += data[j+i], s2 += s1;
821            s1 %= 65521, s2 %= 65521;
822            j += blocklen;
823            blocklen = 5552;
824         }
825         stbiw__sbpush(out, STBIW_UCHAR(s2 » 8));
826         stbiw__sbpush(out, STBIW_UCHAR(s2));
827         stbiw__sbpush(out, STBIW_UCHAR(s1 » 8));
828         stbiw__sbpush(out, STBIW_UCHAR(s1));
829      }
830      *out_len = stbiw__sbn(out);
831      // make returned pointer freeable
832      STBIW_MEMMOVE(stbiw__sbraw(out), out, *out_len);
833      return (unsigned char *) stbiw__sbraw(out);
834   }
835
836   static unsigned int stbiw__crc32(unsigned char *buffer, int len)
837   {
838      static unsigned int crc_table[256] =
839      {
840         0x00000000, 0x77073096, 0xEE0E612C, 0x990951BA, 0x076DC419, 0x706AF48F, 0xE963A535, 0x9E6495A3,
841         0x0eDB8832, 0x79DCB8A4, 0xE0D5E91E, 0x97D2D988, 0x09B64C2B, 0x7EB17CBD, 0xE7B82D07, 0x90BF1D91,
842         0x1DB71064, 0x6AB020F2, 0xF3B97148, 0x84BE41DE, 0x1ADAD47D, 0x6DDDE4EB, 0xF4D4B551, 0x83D385C7,
843         0x136C9856, 0x646BA8C0, 0xFD62F97A, 0x8A65C9EC, 0x14015C4F, 0x63066CD9, 0xFA0F3D63, 0x8D080DF5,
844         0x3B6E20C8, 0x4C69105E, 0xD56041E4, 0xA2677172, 0x3C03E4D1, 0x4B04D447, 0xD20D85FD, 0xA50AB56B,
845         0x35B5A8FA, 0x42B2986C, 0xDBBBC9D6, 0xACBCF940, 0x32D86CE3, 0x45DF5C75, 0xDCD60DCF, 0xABD13D59,
846         0x26D930AC, 0x51DE003A, 0xC8D75180, 0xBFD06116, 0x21B4F4B5, 0x56B3C423, 0xCFBA9599, 0xB8BDA50F,
847         0x2802B89E, 0x5F058808, 0xC60CD9B2, 0xB10BE924, 0x2F6F7C87, 0x58684C11, 0xC1611DAB, 0xB6662D3D,
848         0x76DC4190, 0x01DB7106, 0x98D220BC, 0xEFD5102A, 0x71B18589, 0x06B6B51F, 0x9FBFE4A5, 0xE8B8D433,
849         0x7807C9A2, 0x0F00F934, 0x9609A88E, 0xE10E9818, 0x7F6A0DBB, 0x086D3D2D, 0x91646C97, 0xE6635C01,
850         0x6B6B51F4, 0x1C6C6162, 0x856530D8, 0xF262004E, 0x6C0695ED, 0x1B01A57B, 0x8208F4C1, 0xF50FC457,
```

```
851       0x65B0D9C6, 0x12B7E950, 0x8BBEB8EA, 0xFCB9887C, 0x62DD1DDF, 0x15DA2D49, 0x8CD37CF3, 0xFBD44C65,
852       0x4DB26158, 0x3AB551CE, 0xA3BC0074, 0xD4BB30E2, 0x4ADFA541, 0x3DD895D7, 0xA4D1C46D, 0xD3D6F4FB,
853       0x4369E96A, 0x346ED9FC, 0xAD678846, 0xDA60B8D0, 0x44042D73, 0x33031DE5, 0xAA0A4C5F, 0xDD0D7CC9,
854       0x5005713C, 0x270241AA, 0xBE0B1010, 0xC90C2086, 0x5768B525, 0x206F85B3, 0xB966D409, 0xCE61E49F,
855       0x5EDEF90E, 0x29D9C998, 0xB0D09822, 0xC7D7A8B4, 0x59B33D17, 0x2EB40D81, 0xB7BD5C3B, 0xC0BA6CAD,
856       0xEDB88320, 0x9ABFB3B6, 0x03B6E20C, 0x74B1D29A, 0xEAD54739, 0x9DD277AF, 0x04DB2615, 0x73DC1683,
857       0xE3630B12, 0x94643B84, 0x0D6D6A3E, 0x7A6A5AA8, 0xE40ECF0B, 0x9309FF9D, 0x0A00AE27, 0x7D079EB1,
858       0xF00F9344, 0x8708A3D2, 0x1E01F268, 0x6906C2FE, 0xF762575D, 0x806567CB, 0x196C3671, 0x6E6B06E7,
859       0xFED41B76, 0x89D32BE0, 0x10DA7A5A, 0x67DD4ACC, 0xF9B9DF6F, 0x8EBEEFF9, 0x17B7BE43, 0x60B08ED5,
860       0xD6D6A3E8, 0xA1D1937E, 0x38D8C2C4, 0x4FDFF252, 0xD1BB67F1, 0xA6BC5767, 0x3FB506DD, 0x48B2364B,
861       0xD80D2BDA, 0xAF0A1B4C, 0x36034AF6, 0x41047A60, 0xDF60EFC3, 0xA867DF55, 0x316E8EEF, 0x4669BE79,
862       0xCB61B38C, 0xBC66831A, 0x256FD2A0, 0x5268E236, 0xCC0C7795, 0xBB0B4703, 0x220216B9, 0x5505262F,
863       0xC5BA3BBE, 0xB2BD0B28, 0x2BB45A92, 0x5CB36A04, 0xC2D7FFA7, 0xB5D0CF31, 0x2CD99E8B, 0x5BDEAE1D,
864       0x9B64C2B0, 0xEC63F226, 0x756AA39C, 0x026D930A, 0x9C0906A9, 0xEB0E363F, 0x72076785, 0x05005713,
865       0x95BF4A82, 0xE2B87A14, 0x7BB12BAE, 0x0CB61B38, 0x92D28E9B, 0xE5D5BE0D, 0x7CDCEFB7, 0x0BDBDF21,
866       0x86D3D2D4, 0xF1D4E242, 0x68DDB3F8, 0x1FDA836E, 0x81BE16CD, 0xF6B9265B, 0x6FB077E1, 0x18B74777,
867       0x88085AE6, 0xFF0F6A70, 0x66063BCA, 0x11010B5C, 0x8F659EFF, 0xF862AE69, 0x616BFFD3, 0x166CCF45,
868       0xA00AE278, 0xD70DD2EE, 0x4E048354, 0x3903B3C2, 0xA7672661, 0xD06016F7, 0x4969474D, 0x3E6E77DB,
869       0xAED16A4A, 0xD9D65ADC, 0x40DF0B66, 0x37D83BF0, 0xA9BCAE53, 0xDEBB9EC5, 0x47B2CF7F, 0x30B5FFE9,
870       0xBDBDF21C, 0xCABAC28A, 0x53B39330, 0x24B4A3A6, 0xBAD03605, 0xCDD70693, 0x54DE5729, 0x23D967BF,
871       0xB3667A2E, 0xC4614AB8, 0x5D681B02, 0x2A6F2B94, 0xB40BBE37, 0xC30C8EA1, 0x5A05DF1B, 0x2D02EF8D
872    };
873
874    unsigned int crc = ~0u;
875    int i;
876    for (i=0; i < len; ++i)
877       crc = (crc >> 8) ^ crc_table[buffer[i] ^ (crc & 0xff)];
878    return ~crc;
879 }
880
881 #define stbiw__wpng4(o,a,b,c,d)
       ((o)[0]=STBIW_UCHAR(a),(o)[1]=STBIW_UCHAR(b),(o)[2]=STBIW_UCHAR(c),(o)[3]=STBIW_UCHAR(d),(o)+=4)
882 #define stbiw__wp32(data,v) stbiw__wpng4(data, (v)>>24,(v)>>16,(v)>>8,(v));
883 #define stbiw__wptag(data,s) stbiw__wpng4(data, s[0],s[1],s[2],s[3])
884
885 static void stbiw__wpcrc(unsigned char **data, int len)
886 {
887    unsigned int crc = stbiw__crc32(*data - len - 4, len+4);
888    stbiw__wp32(*data, crc);
889 }
890
891 static unsigned char stbiw__paeth(int a, int b, int c)
892 {
893    int p = a + b - c, pa = abs(p-a), pb = abs(p-b), pc = abs(p-c);
894    if (pa <= pb && pa <= pc) return STBIW_UCHAR(a);
895    if (pb <= pc) return STBIW_UCHAR(b);
896    return STBIW_UCHAR(c);
897 }
898
899 unsigned char *stbi_write_png_to_mem(unsigned char *pixels, int stride_bytes, int x, int y, int n, int
       *out_len)
900 {
901    int ctype[5] = { -1, 0, 4, 2, 6 };
902    unsigned char sig[8] = { 137,80,78,71,13,10,26,10 };
903    unsigned char *out,*o, *filt, *zlib;
904    signed char *line_buffer;
905    int i,j,k,p,zlen;
906
907    if (stride_bytes == 0)
908       stride_bytes = x * n;
909
910    filt = (unsigned char *) STBIW_MALLOC((x*n+1) * y); if (!filt) return 0;
911    line_buffer = (signed char *) STBIW_MALLOC(x * n); if (!line_buffer) { STBIW_FREE(filt); return 0; }
912    for (j=0; j < y; ++j) {
913       static int mapping[] = { 0,1,2,3,4 };
914       static int firstmap[] = { 0,1,0,5,6 };
915       int *mymap = j ? mapping : firstmap;
916       int best = 0, bestval = 0x7fffffff;
917       for (p=0; p < 2; ++p) {
918          for (k= p?best:0; k < 5; ++k) {
919             int type = mymap[k],est=0;
920             unsigned char *z = pixels + stride_bytes*j;
921             for (i=0; i < n; ++i)
922                switch (type) {
923                   case 0: line_buffer[i] = z[i]; break;
924                   case 1: line_buffer[i] = z[i]; break;
925                   case 2: line_buffer[i] = z[i] - z[i-stride_bytes]; break;
926                   case 3: line_buffer[i] = z[i] - (z[i-stride_bytes]>>1); break;
927                   case 4: line_buffer[i] = (signed char) (z[i] - stbiw__paeth(0,z[i-stride_bytes],0));
       break;
928                   case 5: line_buffer[i] = z[i]; break;
929                   case 6: line_buffer[i] = z[i]; break;
930                }
931             for (i=n; i < x*n; ++i) {
932                switch (type) {
933                   case 0: line_buffer[i] = z[i]; break;
934                   case 1: line_buffer[i] = z[i] - z[i-n]; break;
```

```
935                     case 2: line_buffer[i] = z[i] - z[i-stride_bytes]; break;
936                     case 3: line_buffer[i] = z[i] - ((z[i-n] + z[i-stride_bytes])>>1); break;
937                     case 4: line_buffer[i] = z[i] - stbiw__paeth(z[i-n], z[i-stride_bytes],
      z[i-stride_bytes-n]); break;
938                     case 5: line_buffer[i] = z[i] - (z[i-n]>>1); break;
939                     case 6: line_buffer[i] = z[i] - stbiw__paeth(z[i-n], 0,0); break;
940                  }
941               }
942               if (p) break;
943               for (i=0; i < x*n; ++i)
944                  est += abs((signed char) line_buffer[i]);
945               if (est < bestval) { bestval = est; best = k; }
946            }
947         }
948         // when we get here, best contains the filter type, and line_buffer contains the data
949         filt[j*(x*n+1)] = (unsigned char) best;
950         STBIW_MEMMOVE(filt+j*(x*n+1)+1, line_buffer, x*n);
951      }
952      STBIW_FREE(line_buffer);
953      zlib = stbi_zlib_compress(filt, y*( x*n+1), &zlen, 8); // increase 8 to get smaller but use more
      memory
954      STBIW_FREE(filt);
955      if (!zlib) return 0;
956
957      // each tag requires 12 bytes of overhead
958      out = (unsigned char *) STBIW_MALLOC(8 + 12+13 + 12+zlen + 12);
959      if (!out) return 0;
960      *out_len = 8 + 12+13 + 12+zlen + 12;
961
962      o=out;
963      STBIW_MEMMOVE(o,sig,8); o+= 8;
964      stbiw__wp32(o, 13); // header length
965      stbiw__wptag(o, "IHDR");
966      stbiw__wp32(o, x);
967      stbiw__wp32(o, y);
968      *o++ = 8;
969      *o++ = STBIW_UCHAR(ctype[n]);
970      *o++ = 0;
971      *o++ = 0;
972      *o++ = 0;
973      stbiw__wpcrc(&o,13);
974
975      stbiw__wp32(o, zlen);
976      stbiw__wptag(o, "IDAT");
977      STBIW_MEMMOVE(o, zlib, zlen);
978      o += zlen;
979      STBIW_FREE(zlib);
980      stbiw__wpcrc(&o, zlen);
981
982      stbiw__wp32(o,0);
983      stbiw__wptag(o, "IEND");
984      stbiw__wpcrc(&o,0);
985
986      STBIW_ASSERT(o == out + *out_len);
987
988      return out;
989 }
990
991 #ifndef STBI_WRITE_NO_STDIO
992 STBIWDEF int stbi_write_png(char const *filename, int x, int y, int comp, const void *data, int
      stride_bytes)
993 {
994      FILE *f;
995      int len;
996      unsigned char *png = stbi_write_png_to_mem((unsigned char *) data, stride_bytes, x, y, comp, &len);
997      if (png == NULL) return 0;
998      f = fopen(filename, "wb");
999      if (!f) { STBIW_FREE(png); return 0; }
1000      fwrite(png, 1, len, f);
1001      fclose(f);
1002      STBIW_FREE(png);
1003      return 1;
1004 }
1005 #endif
1006
1007 STBIWDEF int stbi_write_png_to_func(stbi_write_func *func, void *context, int x, int y, int comp, const
       void *data, int stride_bytes)
1008 {
1009      int len;
1010      unsigned char *png = stbi_write_png_to_mem((unsigned char *) data, stride_bytes, x, y, comp, &len);
1011      if (png == NULL) return 0;
1012      func(context, png, len);
1013      STBIW_FREE(png);
1014      return 1;
1015 }
1016
1017 #endif // STB_IMAGE_WRITE_IMPLEMENTATION
```

```
1018
1019 /* Revision history
1020       1.02 (2016-04-02)
1021              avoid allocating large structures on the stack
1022       1.01 (2016-01-16)
1023              STBIW_REALLOC_SIZED: support allocators with no realloc support
1024              avoid race-condition in crc initialization
1025              minor compile issues
1026       1.00 (2015-09-14)
1027              installable file IO function
1028       0.99 (2015-09-13)
1029              warning fixes; TGA rle support
1030       0.98 (2015-04-08)
1031              added STBIW_MALLOC, STBIW_ASSERT etc
1032       0.97 (2015-01-18)
1033              fixed HDR asserts, rewrote HDR rle logic
1034       0.96 (2015-01-17)
1035              add HDR output
1036              fix monochrome BMP
1037       0.95 (2014-08-17)
1038                add monochrome TGA output
1039       0.94 (2014-05-31)
1040              rename private functions to avoid conflicts with stb_image.h
1041       0.93 (2014-05-27)
1042              warning fixes
1043       0.92 (2010-08-01)
1044              casts to unsigned char to fix warnings
1045       0.91 (2010-07-17)
1046              first public release
1047       0.90   first internal release
1048 */
```

27.12 lib/glfw/deps/tinycthread.h File Reference

```
#include <time.h>
#include <sys/time.h>
#include <pthread.h>
```

Macros

- #define **_TTHREAD_POSIX_**
- #define **_TTHREAD_PLATFORM_DEFINED_**
- #define **_GNU_SOURCE**
- #define **_POSIX_C_SOURCE** 199309L
- #define **_XOPEN_SOURCE** 500
- #define **TIME_UTC** 0
- #define TINYCTHREAD_VERSION_MAJOR 1
- #define TINYCTHREAD_VERSION_MINOR 1
- #define TINYCTHREAD_VERSION (TINYCTHREAD_VERSION_MAJOR * 100 + TINYCTHREAD_VERSION_MINOR)
- #define _Thread_local
- #define **TSS_DTOR_ITERATIONS** 0
- #define thrd_error 0
- #define thrd_success 1
- #define thrd_timeout 2
- #define thrd_busy 3
- #define thrd_nomem 4
- #define **mtx_plain** 1
- #define **mtx_timed** 2
- #define **mtx_try** 4
- #define **mtx_recursive** 8

Typedefs

- typedef pthread_mutex_t **mtx_t**
- typedef pthread_cond_t **cnd_t**
- typedef pthread_t **thrd_t**
- typedef int(∗ thrd_start_t) (void ∗arg)
- typedef pthread_key_t **tss_t**
- typedef void(∗ tss_dtor_t) (void ∗val)

Functions

- int mtx_init (mtx_t ∗mtx, int type)
- void mtx_destroy (mtx_t ∗mtx)
- int mtx_lock (mtx_t ∗mtx)
- int mtx_timedlock (mtx_t ∗mtx, const struct timespec ∗ts)
- int mtx_trylock (mtx_t ∗mtx)
- int mtx_unlock (mtx_t ∗mtx)
- int cnd_init (cnd_t ∗cond)
- void cnd_destroy (cnd_t ∗cond)
- int cnd_signal (cnd_t ∗cond)
- int cnd_broadcast (cnd_t ∗cond)
- int cnd_wait (cnd_t ∗cond, mtx_t ∗mtx)
- int cnd_timedwait (cnd_t ∗cond, mtx_t ∗mtx, const struct timespec ∗ts)
- int thrd_create (thrd_t ∗thr, thrd_start_t func, void ∗arg)
- thrd_t thrd_current (void)
- int thrd_detach (thrd_t thr)
- int thrd_equal (thrd_t thr0, thrd_t thr1)
- void thrd_exit (int res)
- int thrd_join (thrd_t thr, int ∗res)
- int thrd_sleep (const struct timespec ∗time_point, struct timespec ∗remaining)
- void thrd_yield (void)
- int tss_create (tss_t ∗key, tss_dtor_t dtor)
- void tss_delete (tss_t key)
- void ∗ tss_get (tss_t key)
- int tss_set (tss_t key, void ∗val)

27.12.1 Macro Definition Documentation

27.12.1.1 _Thread_local

```
#define _Thread_local
```

Thread local storage keyword. A variable that is declared with the `_Thread_local` keyword makes the value of the variable local to each thread (known as thread-local storage, or TLS). Example usage:

```
// This variable is local to each thread.
_Thread_local int variable;
```

Note

The `_Thread_local` keyword is a macro that maps to the corresponding compiler directive (e.g. `__declspec(thread)`).

This directive is currently not supported on Mac OS X (it will give a compiler error), since compile-time TLS is not supported in the Mac OS X executable format. Also, some older versions of MinGW (before GCC 4.x) do not support this directive.

27.12.1.2 thrd_busy

```
#define thrd_busy 3
```

The requested operation failed because a tesource requested by a test and return function is already in use

27.12.1.3 thrd_error

```
#define thrd_error 0
```

The requested operation failed

27.12.1.4 thrd_nomem

```
#define thrd_nomem 4
```

The requested operation failed because it was unable to allocate memory

27.12.1.5 thrd_success

```
#define thrd_success 1
```

The requested operation succeeded

27.12.1.6 thrd_timeout

```
#define thrd_timeout 2
```

The time specified in the call was reached without acquiring the requested resource

27.12.1.7 TINYCTHREAD_VERSION

```
#define TINYCTHREAD_VERSION (TINYCTHREAD_VERSION_MAJOR * 100 + TINYCTHREAD_VERSION_MINOR)
```

TinyCThread version (full version).

27.12.1.8 TINYCTHREAD_VERSION_MAJOR

```
#define TINYCTHREAD_VERSION_MAJOR 1
```

TinyCThread version (major number).

27.12.1.9 TINYCTHREAD_VERSION_MINOR

```
#define TINYCTHREAD_VERSION_MINOR 1
```

TinyCThread version (minor number).

27.12.2 Typedef Documentation

27.12.2.1 thrd_start_t

```
typedef int(* thrd_start_t) (void *arg)
```

Thread start function. Any thread that is started with the thrd_create() function must be started through a function of this type.

Parameters

arg	The thread argument (the `arg` argument of the corresponding thrd_create() call).

Returns

The thread return value, which can be obtained by another thread by using the thrd_join() function.

27.12.2.2 tss_dtor_t

```
typedef void(* tss_dtor_t) (void *val)
```

Destructor function for a thread-specific storage.

Parameters

val	The value of the destructed thread-specific storage.

27.12.3 Function Documentation

27.12.3.1 cnd_broadcast()

```
int cnd_broadcast (
            cnd_t * cond )
```

Broadcast a condition variable. Unblocks all of the threads that are blocked on the given condition variable at the time of the call. If no threads are blocked on the condition variable at the time of the call, the function does nothing and return success.

Parameters

cond	A condition variable object.

Returns

thrd_success on success, or thrd_error if the request could not be honored.

27.12.3.2 cnd_destroy()

```
void cnd_destroy (
            cnd_t * cond )
```

Release any resources used by the given condition variable.

Parameters

cond	A condition variable object.

27.12.3.3 cnd_init()

```
int cnd_init (
            cnd_t * cond )
```

Create a condition variable object.

Parameters

cond	A condition variable object.

Returns

thrd_success on success, or thrd_error if the request could not be honored.

27.12.3.4 cnd_signal()

```
int cnd_signal (
            cnd_t * cond )
```

Signal a condition variable. Unblocks one of the threads that are blocked on the given condition variable at the time of the call. If no threads are blocked on the condition variable at the time of the call, the function does nothing and return success.

Parameters

cond	A condition variable object.

Returns

thrd_success on success, or thrd_error if the request could not be honored.

27.12.3.5 cnd_timedwait()

```
int cnd_timedwait (
            cnd_t * cond,
```

```
            mtx_t * mtx,
            const struct timespec * ts )
```

Wait for a condition variable to become signaled. The function atomically unlocks the given mutex and endeavors to block until the given condition variable is signaled by a call to cnd_signal or to cnd_broadcast, or until after the specified time. When the calling thread becomes unblocked it locks the mutex before it returns.

Parameters

cond	A condition variable object.
mtx	A mutex object.
xt	A point in time at which the request will time out (absolute time).

Returns

thrd_success upon success, or thrd_timeout if the time specified in the call was reached without acquiring the requested resource, or thrd_error if the request could not be honored.

27.12.3.6 cnd_wait()

```
int cnd_wait (
            cnd_t * cond,
            mtx_t * mtx )
```

Wait for a condition variable to become signaled. The function atomically unlocks the given mutex and endeavors to block until the given condition variable is signaled by a call to cnd_signal or to cnd_broadcast. When the calling thread becomes unblocked it locks the mutex before it returns.

Parameters

cond	A condition variable object.
mtx	A mutex object.

Returns

thrd_success on success, or thrd_error if the request could not be honored.

27.12.3.7 mtx_destroy()

```
void mtx_destroy (
            mtx_t * mtx )
```

Release any resources used by the given mutex.

Parameters

mtx	A mutex object.

27.12.3.8 mtx_init()

```
int mtx_init (
            mtx_t * mtx,
            int type )
```

Create a mutex object.

Parameters

mtx	A mutex object.
type	Bit-mask that must have one of the following six values: • `mtx_plain` for a simple non-recursive mutex • `mtx_timed` for a non-recursive mutex that supports timeout • `mtx_try` for a non-recursive mutex that supports test and return • `mtx_plain` | `mtx_recursive` (same as `mtx_plain`, but recursive) • `mtx_timed` | `mtx_recursive` (same as `mtx_timed`, but recursive) • `mtx_try` | `mtx_recursive` (same as `mtx_try`, but recursive)

Returns

thrd_success on success, or thrd_error if the request could not be honored.

27.12.3.9 mtx_lock()

```
int mtx_lock (
            mtx_t * mtx )
```

Lock the given mutex. Blocks until the given mutex can be locked. If the mutex is non-recursive, and the calling thread already has a lock on the mutex, this call will block forever.

Parameters

mtx	A mutex object.

Returns

thrd_success on success, or thrd_error if the request could not be honored.

27.12.3.10 mtx_timedlock()

```
int mtx_timedlock (
            mtx_t * mtx,
            const struct timespec * ts )
```

NOT YET IMPLEMENTED.

27.12.3.11 mtx_trylock()

```
int mtx_trylock (
            mtx_t * mtx )
```

Try to lock the given mutex. The specified mutex shall support either test and return or timeout. If the mutex is already locked, the function returns without blocking.

Parameters

mtx	A mutex object.

Returns

thrd_success on success, or thrd_busy if the resource requested is already in use, or thrd_error if the request could not be honored.

27.12.3.12 mtx_unlock()

```
int mtx_unlock (
            mtx_t * mtx )
```

Unlock the given mutex.

Parameters

mtx	A mutex object.

Returns

thrd_success on success, or thrd_error if the request could not be honored.

27.12.3.13 thrd_create()

```
int thrd_create (
            thrd_t * thr,
            thrd_start_t func,
            void * arg )
```

Create a new thread.

Parameters

thr	Identifier of the newly created thread.
func	A function pointer to the function that will be executed in the new thread.
arg	An argument to the thread function.

Returns

thrd_success on success, or thrd_nomem if no memory could be allocated for the thread requested, or thrd_error if the request could not be honored.

Note

A thread's identifier may be reused for a different thread once the original thread has exited and either been detached or joined to another thread.

27.12.3.14 thrd_current()

```
thrd_t thrd_current (
            void  )
```

Identify the calling thread.

Returns

The identifier of the calling thread.

27.12.3.15 thrd_detach()

```
int thrd_detach (
            thrd_t thr )
```

NOT YET IMPLEMENTED.

27.12.3.16 thrd_equal()

```
int thrd_equal (
            thrd_t thr0,
            thrd_t thr1 )
```

Compare two thread identifiers. The function determines if two thread identifiers refer to the same thread.

Returns

Zero if the two thread identifiers refer to different threads. Otherwise a nonzero value is returned.

27.12.3.17 thrd_exit()

```
void thrd_exit (
            int res )
```

Terminate execution of the calling thread.

Parameters

res	Result code of the calling thread.

27.12.3.18 thrd_join()

```
int thrd_join (
            thrd_t thr,
            int * res )
```

Wait for a thread to terminate. The function joins the given thread with the current thread by blocking until the other thread has terminated.

Parameters

thr	The thread to join with.
res	If this pointer is not NULL, the function will store the result code of the given thread in the integer pointed to by `res`.

Returns

thrd_success on success, or thrd_error if the request could not be honored.

27.12.3.19 thrd_sleep()

```
int thrd_sleep (
            const struct timespec * time_point,
            struct timespec * remaining )
```

Put the calling thread to sleep. Suspend execution of the calling thread.

Parameters

time_point	A point in time at which the thread will resume (absolute time).
remaining	If non-NULL, this parameter will hold the remaining time until time_point upon return. This will typically be zero, but if the thread was woken up by a signal that is not ignored before time_point was reached `remaining` will hold a positive time.

Returns

0 (zero) on successful sleep, or -1 if an interrupt occurred.

27.12.3.20 thrd_yield()

```
void thrd_yield (
            void  )
```

Yield execution to another thread. Permit other threads to run, even if the current thread would ordinarily continue to run.

27.12.3.21 tss_create()

```
int tss_create (
            tss_t * key,
            tss_dtor_t dtor )
```

Create a thread-specific storage.

Parameters

key	The unique key identifier that will be set if the function is successful.
dtor	Destructor function. This can be NULL.

Returns

thrd_success on success, or thrd_error if the request could not be honored.

Note

The destructor function is not supported under Windows. If `dtor` is not NULL when calling this function under Windows, the function will fail and return thrd_error.

27.12.3.22 tss_delete()

```
void tss_delete (
            tss_t key )
```

Delete a thread-specific storage. The function releases any resources used by the given thread-specific storage.

Parameters

key	The key that shall be deleted.

27.12.3.23 tss_get()

```
void * tss_get (
            tss_t key )
```

Get the value for a thread-specific storage.

Parameters

key	The thread-specific storage identifier.

Returns

The value for the current thread held in the given thread-specific storage.

27.12.3.24 tss_set()

```
int tss_set (
            tss_t key,
            void * val )
```

Set the value for a thread-specific storage.

Parameters

key	The thread-specific storage identifier.
val	The value of the thread-specific storage to set for the current thread.

Returns

thrd_success on success, or thrd_error if the request could not be honored.

27.13 tinycthread.h

Go to the documentation of this file.

```
1 /* -*- mode: c; tab-width: 2; indent-tabs-mode: nil; -*-
2 Copyright (c) 2012 Marcus Geelnard
3
4 This software is provided 'as-is', without any express or implied
5 warranty. In no event will the authors be held liable for any damages
6 arising from the use of this software.
7
8 Permission is granted to anyone to use this software for any purpose,
9 including commercial applications, and to alter it and redistribute it
10 freely, subject to the following restrictions:
11
12     1. The origin of this software must not be misrepresented; you must not
13     claim that you wrote the original software. If you use this software
14     in a product, an acknowledgment in the product documentation would be
15     appreciated but is not required.
16
17     2. Altered source versions must be plainly marked as such, and must not be
18     misrepresented as being the original software.
19
20     3. This notice may not be removed or altered from any source
21     distribution.
22 */
23
24 #ifndef _TINYCTHREAD_H_
25 #define _TINYCTHREAD_H_
26
50 /* Which platform are we on? */
51 #if !defined(_TTHREAD_PLATFORM_DEFINED_)
52   #if defined(_WIN32) || defined(__WIN32__) || defined(__WINDOWS__)
53     #define _TTHREAD_WIN32_
54   #else
55     #define _TTHREAD_POSIX_
56   #endif
57   #define _TTHREAD_PLATFORM_DEFINED_
58 #endif
59
60 /* Activate some POSIX functionality (e.g. clock_gettime and recursive mutexes) */
61 #if defined(_TTHREAD_POSIX_)
62   #undef _FEATURES_H
63   #if !defined(_GNU_SOURCE)
64     #define _GNU_SOURCE
65   #endif
66   #if !defined(_POSIX_C_SOURCE) || ((_POSIX_C_SOURCE - 0) < 199309L)
67     #undef _POSIX_C_SOURCE
68     #define _POSIX_C_SOURCE 199309L
69   #endif
70   #if !defined(_XOPEN_SOURCE) || ((_XOPEN_SOURCE - 0) < 500)
71     #undef _XOPEN_SOURCE
72     #define _XOPEN_SOURCE 500
73   #endif
74 #endif
75
76 /* Generic includes */
77 #include <time.h>
78
79 /* Platform specific includes */
80 #if defined(_TTHREAD_POSIX_)
81   #include <sys/time.h>
82   #include <pthread.h>
83 #elif defined(_TTHREAD_WIN32_)
84   #ifndef WIN32_LEAN_AND_MEAN
85     #define WIN32_LEAN_AND_MEAN
86     #define __UNDEF_LEAN_AND_MEAN
87   #endif
88   #include <windows.h>
89   #ifdef __UNDEF_LEAN_AND_MEAN
90     #undef WIN32_LEAN_AND_MEAN
91     #undef __UNDEF_LEAN_AND_MEAN
92   #endif
93 #endif
94
95 /* Workaround for missing TIME_UTC: If time.h doesn't provide TIME_UTC,
96    it's quite likely that libc does not support it either. Hence, fall back to
97    the only other supported time specifier: CLOCK_REALTIME (and if that fails,
```

```
98    we're probably emulating clock_gettime anyway, so anything goes). */
99 #ifndef TIME_UTC
100   #ifdef CLOCK_REALTIME
101     #define TIME_UTC CLOCK_REALTIME
102   #else
103     #define TIME_UTC 0
104   #endif
105 #endif
106
107 /* Workaround for missing clock_gettime (most Windows compilers, afaik) */
108 #if defined(_TTHREAD_WIN32_) || defined(__APPLE_CC__)
109 #define _TTHREAD_EMULATE_CLOCK_GETTIME_
110 /* Emulate struct timespec */
111 #if defined(_TTHREAD_WIN32_)
112 struct _ttherad_timespec {
113   time_t tv_sec;
114   long   tv_nsec;
115 };
116 #define timespec _ttherad_timespec
117 #endif
118
119 /* Emulate clockid_t */
120 typedef int _tthread_clockid_t;
121 #define clockid_t _tthread_clockid_t
122
123 /* Emulate clock_gettime */
124 int _tthread_clock_gettime(clockid_t clk_id, struct timespec *ts);
125 #define clock_gettime _tthread_clock_gettime
126 #ifndef CLOCK_REALTIME
127   #define CLOCK_REALTIME 0
128 #endif
129 #endif
130
131
133 #define TINYCTHREAD_VERSION_MAJOR 1
135 #define TINYCTHREAD_VERSION_MINOR 1
137 #define TINYCTHREAD_VERSION (TINYCTHREAD_VERSION_MAJOR * 100 + TINYCTHREAD_VERSION_MINOR)
138
158 /* FIXME: Check for a PROPER value of __STDC_VERSION__ to know if we have C11 */
159 #if !(defined(__STDC_VERSION__) && (__STDC_VERSION__ >= 201102L)) && !defined(_Thread_local)
160  #if defined(__GNUC__) || defined(__INTEL_COMPILER) || defined(__SUNPRO_CC) || defined(__IBMCPP__)
161   #define _Thread_local __thread
162  #else
163   #define _Thread_local __declspec(thread)
164  #endif
165 #endif
166
167 /* Macros */
168 #define TSS_DTOR_ITERATIONS 0
169
170 /* Function return values */
171 #define thrd_error    0
172 #define thrd_success  1
173 #define thrd_timeout  2
174 #define thrd_busy     3
175 #define thrd_nomem    4
177 /* Mutex types */
178 #define mtx_plain     1
179 #define mtx_timed     2
180 #define mtx_try       4
181 #define mtx_recursive 8
182
183 /* Mutex */
184 #if defined(_TTHREAD_WIN32_)
185 typedef struct {
186   CRITICAL_SECTION mHandle;   /* Critical section handle */
187   int mAlreadyLocked;         /* TRUE if the mutex is already locked */
188   int mRecursive;             /* TRUE if the mutex is recursive */
189 } mtx_t;
190 #else
191 typedef pthread_mutex_t mtx_t;
192 #endif
193
206 int mtx_init(mtx_t *mtx, int type);
207
211 void mtx_destroy(mtx_t *mtx);
212
221 int mtx_lock(mtx_t *mtx);
222
225 int mtx_timedlock(mtx_t *mtx, const struct timespec *ts);
226
235 int mtx_trylock(mtx_t *mtx);
236
242 int mtx_unlock(mtx_t *mtx);
243
244 /* Condition variable */
245 #if defined(_TTHREAD_WIN32_)
```

```
246 typedef struct {
247   HANDLE mEvents[2];                  /* Signal and broadcast event HANDLEs. */
248   unsigned int mWaitersCount;         /* Count of the number of waiters. */
249   CRITICAL_SECTION mWaitersCountLock; /* Serialize access to mWaitersCount. */
250 } cnd_t;
251 #else
252 typedef pthread_cond_t cnd_t;
253 #endif
254 
260 int cnd_init(cnd_t *cond);
261 
265 void cnd_destroy(cnd_t *cond);
266 
275 int cnd_signal(cnd_t *cond);
276 
285 int cnd_broadcast(cnd_t *cond);
286 
297 int cnd_wait(cnd_t *cond, mtx_t *mtx);
298 
311 int cnd_timedwait(cnd_t *cond, mtx_t *mtx, const struct timespec *ts);
312 
313 /* Thread */
314 #if defined(_TTHREAD_WIN32_)
315 typedef HANDLE thrd_t;
316 #else
317 typedef pthread_t thrd_t;
318 #endif
319 
328 typedef int (*thrd_start_t)(void *arg);
329 
342 int thrd_create(thrd_t *thr, thrd_start_t func, void *arg);
343 
347 thrd_t thrd_current(void);
348 
351 int thrd_detach(thrd_t thr);
352 
358 int thrd_equal(thrd_t thr0, thrd_t thr1);
359 
363 void thrd_exit(int res);
364 
374 int thrd_join(thrd_t thr, int *res);
375 
386 int thrd_sleep(const struct timespec *time_point, struct timespec *remaining);
387 
392 void thrd_yield(void);
393 
394 /* Thread local storage */
395 #if defined(_TTHREAD_WIN32_)
396 typedef DWORD tss_t;
397 #else
398 typedef pthread_key_t tss_t;
399 #endif
400 
404 typedef void (*tss_dtor_t)(void *val);
405 
416 int tss_create(tss_t *key, tss_dtor_t dtor);
417 
423 void tss_delete(tss_t key);
424 
430 void *tss_get(tss_t key);
431 
439 int tss_set(tss_t key, void *val);
440 
441 
442 #endif /* _TINYTHREAD_H_ */
443 
```

27.14 stdint.h

```
1 // ISO C9x  compliant stdint.h for Microsoft Visual Studio
2 // Based on ISO/IEC 9899:TC2 Committee draft (May 6, 2005) WG14/N1124
3 //
4 //  Copyright (c) 2006-2008 Alexander Chemeris
5 //
6 // Redistribution and use in source and binary forms, with or without
7 // modification, are permitted provided that the following conditions are met:
8 //
9 //   1. Redistributions of source code must retain the above copyright notice,
10 //      this list of conditions and the following disclaimer.
11 //
12 //   2. Redistributions in binary form must reproduce the above copyright
13 //      notice, this list of conditions and the following disclaimer in the
14 //      documentation and/or other materials provided with the distribution.
```

```
15 //
16 //   3. The name of the author may be used to endorse or promote products
17 //      derived from this software without specific prior written permission.
18 //
19 // THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR IMPLIED
20 // WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
21 // MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
22 // EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
23 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
24 // PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
25 // OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
26 // WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
27 // OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
28 // ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29 //
30
31
32 #ifndef _MSC_VER // [
33 #error "Use this header only with Microsoft Visual C++ compilers!"
34 #endif // _MSC_VER ]
35
36 #ifndef _MSC_STDINT_H_ // [
37 #define _MSC_STDINT_H_
38
39 #if _MSC_VER > 1000
40 #pragma once
41 #endif
42
43 #include <limits.h>
44
45 // For Visual Studio 6 in C++ mode and for many Visual Studio versions when
46 // compiling for ARM we should wrap <wchar.h> include with 'extern "C++" {}'
47 // or compiler give many errors like this:
48 //   error C2733: second C linkage of overloaded function 'wmemchr' not allowed
49 #ifdef __cplusplus
50 extern "C" {
51 #endif
52 #  include <wchar.h>
53 #ifdef __cplusplus
54 }
55 #endif
56
57 // Define _W64 macros to mark types changing their size, like intptr_t.
58 #ifndef _W64
59 #  if !defined(__midl) && (defined(_X86_) || defined(_M_IX86)) && _MSC_VER >= 1300
60 #     define _W64 __w64
61 #  else
62 #     define _W64
63 #  endif
64 #endif
65
66
67 // 7.18.1 Integer types
68
69 // 7.18.1.1 Exact-width integer types
70
71 // Visual Studio 6 and Embedded Visual C++ 4 doesn't
72 // realize that, e.g. char has the same size as __int8
73 // so we give up on __intX for them.
74 #if (_MSC_VER < 1300)
75    typedef signed char       int8_t;
76    typedef signed short      int16_t;
77    typedef signed int        int32_t;
78    typedef unsigned char     uint8_t;
79    typedef unsigned short    uint16_t;
80    typedef unsigned int      uint32_t;
81 #else
82    typedef signed __int8     int8_t;
83    typedef signed __int16    int16_t;
84    typedef signed __int32    int32_t;
85    typedef unsigned __int8   uint8_t;
86    typedef unsigned __int16  uint16_t;
87    typedef unsigned __int32  uint32_t;
88 #endif
89 typedef signed __int64       int64_t;
90 typedef unsigned __int64     uint64_t;
91
92
93 // 7.18.1.2 Minimum-width integer types
94 typedef int8_t    int_least8_t;
95 typedef int16_t   int_least16_t;
96 typedef int32_t   int_least32_t;
97 typedef int64_t   int_least64_t;
98 typedef uint8_t   uint_least8_t;
99 typedef uint16_t  uint_least16_t;
100 typedef uint32_t  uint_least32_t;
101 typedef uint64_t  uint_least64_t;
102
```

```
103 // 7.18.1.3 Fastest minimum-width integer types
104 typedef int8_t    int_fast8_t;
105 typedef int16_t   int_fast16_t;
106 typedef int32_t   int_fast32_t;
107 typedef int64_t   int_fast64_t;
108 typedef uint8_t   uint_fast8_t;
109 typedef uint16_t  uint_fast16_t;
110 typedef uint32_t  uint_fast32_t;
111 typedef uint64_t  uint_fast64_t;
112
113 // 7.18.1.4 Integer types capable of holding object pointers
114 #ifdef _WIN64 // [
115    typedef signed __int64    intptr_t;
116    typedef unsigned __int64  uintptr_t;
117 #else // _WIN64 ][
118    typedef _W64 signed int   intptr_t;
119    typedef _W64 unsigned int uintptr_t;
120 #endif // _WIN64 ]
121
122 // 7.18.1.5 Greatest-width integer types
123 typedef int64_t   intmax_t;
124 typedef uint64_t  uintmax_t;
125
126
127 // 7.18.2 Limits of specified-width integer types
128
129 #if !defined(__cplusplus) || defined(__STDC_LIMIT_MACROS) // [   See footnote 220 at page 257 and
       footnote 221 at page 259
130
131 // 7.18.2.1 Limits of exact-width integer types
132 #define INT8_MIN     ((int8_t)_I8_MIN)
133 #define INT8_MAX     _I8_MAX
134 #define INT16_MIN    ((int16_t)_I16_MIN)
135 #define INT16_MAX    _I16_MAX
136 #define INT32_MIN    ((int32_t)_I32_MIN)
137 #define INT32_MAX    _I32_MAX
138 #define INT64_MIN    ((int64_t)_I64_MIN)
139 #define INT64_MAX    _I64_MAX
140 #define UINT8_MAX    _UI8_MAX
141 #define UINT16_MAX   _UI16_MAX
142 #define UINT32_MAX   _UI32_MAX
143 #define UINT64_MAX   _UI64_MAX
144
145 // 7.18.2.2 Limits of minimum-width integer types
146 #define INT_LEAST8_MIN    INT8_MIN
147 #define INT_LEAST8_MAX    INT8_MAX
148 #define INT_LEAST16_MIN   INT16_MIN
149 #define INT_LEAST16_MAX   INT16_MAX
150 #define INT_LEAST32_MIN   INT32_MIN
151 #define INT_LEAST32_MAX   INT32_MAX
152 #define INT_LEAST64_MIN   INT64_MIN
153 #define INT_LEAST64_MAX   INT64_MAX
154 #define UINT_LEAST8_MAX   UINT8_MAX
155 #define UINT_LEAST16_MAX  UINT16_MAX
156 #define UINT_LEAST32_MAX  UINT32_MAX
157 #define UINT_LEAST64_MAX  UINT64_MAX
158
159 // 7.18.2.3 Limits of fastest minimum-width integer types
160 #define INT_FAST8_MIN    INT8_MIN
161 #define INT_FAST8_MAX    INT8_MAX
162 #define INT_FAST16_MIN   INT16_MIN
163 #define INT_FAST16_MAX   INT16_MAX
164 #define INT_FAST32_MIN   INT32_MIN
165 #define INT_FAST32_MAX   INT32_MAX
166 #define INT_FAST64_MIN   INT64_MIN
167 #define INT_FAST64_MAX   INT64_MAX
168 #define UINT_FAST8_MAX   UINT8_MAX
169 #define UINT_FAST16_MAX  UINT16_MAX
170 #define UINT_FAST32_MAX  UINT32_MAX
171 #define UINT_FAST64_MAX  UINT64_MAX
172
173 // 7.18.2.4 Limits of integer types capable of holding object pointers
174 #ifdef _WIN64 // [
175 #  define INTPTR_MIN   INT64_MIN
176 #  define INTPTR_MAX   INT64_MAX
177 #  define UINTPTR_MAX  UINT64_MAX
178 #else // _WIN64 ][
179 #  define INTPTR_MIN   INT32_MIN
180 #  define INTPTR_MAX   INT32_MAX
181 #  define UINTPTR_MAX  UINT32_MAX
182 #endif // _WIN64 ]
183
184 // 7.18.2.5 Limits of greatest-width integer types
185 #define INTMAX_MIN   INT64_MIN
186 #define INTMAX_MAX   INT64_MAX
187 #define UINTMAX_MAX  UINT64_MAX
188
```

```
189 // 7.18.3 Limits of other integer types
190
191 #ifdef _WIN64 // [
192 #  define PTRDIFF_MIN  _I64_MIN
193 #  define PTRDIFF_MAX  _I64_MAX
194 #else  // _WIN64 ][
195 #  define PTRDIFF_MIN  _I32_MIN
196 #  define PTRDIFF_MAX  _I32_MAX
197 #endif  // _WIN64 ]
198
199 #define SIG_ATOMIC_MIN  INT_MIN
200 #define SIG_ATOMIC_MAX  INT_MAX
201
202 #ifndef SIZE_MAX // [
203 #  ifdef _WIN64 // [
204 #     define SIZE_MAX  _UI64_MAX
205 #  else // _WIN64 ][
206 #     define SIZE_MAX  _UI32_MAX
207 #  endif // _WIN64 ]
208 #endif // SIZE_MAX ]
209
210 // WCHAR_MIN and WCHAR_MAX are also defined in <wchar.h>
211 #ifndef WCHAR_MIN // [
212 #  define WCHAR_MIN  0
213 #endif  // WCHAR_MIN ]
214 #ifndef WCHAR_MAX // [
215 #  define WCHAR_MAX  _UI16_MAX
216 #endif  // WCHAR_MAX ]
217
218 #define WINT_MIN  0
219 #define WINT_MAX  _UI16_MAX
220
221 #endif // __STDC_LIMIT_MACROS ]
222
223
224 // 7.18.4 Limits of other integer types
225
226 #if !defined(__cplusplus) || defined(__STDC_CONSTANT_MACROS) // [   See footnote 224 at page 260
227
228 // 7.18.4.1 Macros for minimum-width integer constants
229
230 #define INT8_C(val)  val##i8
231 #define INT16_C(val) val##i16
232 #define INT32_C(val) val##i32
233 #define INT64_C(val) val##i64
234
235 #define UINT8_C(val)  val##ui8
236 #define UINT16_C(val) val##ui16
237 #define UINT32_C(val) val##ui32
238 #define UINT64_C(val) val##ui64
239
240 // 7.18.4.2 Macros for greatest-width integer constants
241 #define INTMAX_C   INT64_C
242 #define UINTMAX_C  UINT64_C
243
244 #endif // __STDC_CONSTANT_MACROS ]
245
246
247 #endif // _MSC_STDINT_H_ ]
```

27.15 lib/glfw/include/GLFW/glfw3.h File Reference

The header of the GLFW 3 API.

```
#include <stddef.h>
#include <stdint.h>
#include <GL/gl.h>
```

Classes

- struct GLFWvidmode

  Video mode type.

- struct GLFWgammaramp

  Gamma ramp.
- struct GLFWimage

  Image data.
- struct GLFWgamepadstate

  Gamepad input state.
- struct GLFWallocator

Macros

- #define **APIENTRY**
- #define **GLFW_APIENTRY_DEFINED**
- #define **GLFWAPI**
- #define GLFW_TRUE 1

  One.
- #define GLFW_FALSE 0

  Zero.
- #define **GLFW_HAT_CENTERED** 0
- #define **GLFW_HAT_UP** 1
- #define **GLFW_HAT_RIGHT** 2
- #define **GLFW_HAT_DOWN** 4
- #define **GLFW_HAT_LEFT** 8
- #define **GLFW_HAT_RIGHT_UP** (GLFW_HAT_RIGHT | GLFW_HAT_UP)
- #define **GLFW_HAT_RIGHT_DOWN** (GLFW_HAT_RIGHT | GLFW_HAT_DOWN)
- #define **GLFW_HAT_LEFT_UP** (GLFW_HAT_LEFT | GLFW_HAT_UP)
- #define **GLFW_HAT_LEFT_DOWN** (GLFW_HAT_LEFT | GLFW_HAT_DOWN)
- #define **GLFW_KEY_UNKNOWN** -1
- #define **GLFW_KEY_SPACE** 32
- #define **GLFW_KEY_APOSTROPHE** 39 /* ' */
- #define **GLFW_KEY_COMMA** 44 /* , */
- #define **GLFW_KEY_MINUS** 45 /* - */
- #define **GLFW_KEY_PERIOD** 46 /* . */
- #define **GLFW_KEY_SLASH** 47 /* / */
- #define **GLFW_KEY_0** 48
- #define **GLFW_KEY_1** 49
- #define **GLFW_KEY_2** 50
- #define **GLFW_KEY_3** 51
- #define **GLFW_KEY_4** 52
- #define **GLFW_KEY_5** 53
- #define **GLFW_KEY_6** 54
- #define **GLFW_KEY_7** 55
- #define **GLFW_KEY_8** 56
- #define **GLFW_KEY_9** 57
- #define **GLFW_KEY_SEMICOLON** 59 /* ; */
- #define **GLFW_KEY_EQUAL** 61 /* = */
- #define **GLFW_KEY_A** 65
- #define **GLFW_KEY_B** 66
- #define **GLFW_KEY_C** 67
- #define **GLFW_KEY_D** 68
- #define **GLFW_KEY_E** 69
- #define **GLFW_KEY_F** 70
- #define **GLFW_KEY_G** 71
- #define **GLFW_KEY_H** 72

- #define **GLFW_KEY_I** 73
- #define **GLFW_KEY_J** 74
- #define **GLFW_KEY_K** 75
- #define **GLFW_KEY_L** 76
- #define **GLFW_KEY_M** 77
- #define **GLFW_KEY_N** 78
- #define **GLFW_KEY_O** 79
- #define **GLFW_KEY_P** 80
- #define **GLFW_KEY_Q** 81
- #define **GLFW_KEY_R** 82
- #define **GLFW_KEY_S** 83
- #define **GLFW_KEY_T** 84
- #define **GLFW_KEY_U** 85
- #define **GLFW_KEY_V** 86
- #define **GLFW_KEY_W** 87
- #define **GLFW_KEY_X** 88
- #define **GLFW_KEY_Y** 89
- #define **GLFW_KEY_Z** 90
- #define **GLFW_KEY_LEFT_BRACKET** 91 /* [*/
- #define **GLFW_KEY_BACKSLASH** 92 /* \ */
- #define **GLFW_KEY_RIGHT_BRACKET** 93 /*] */
- #define **GLFW_KEY_GRAVE_ACCENT** 96 /* ` */
- #define **GLFW_KEY_WORLD_1** 161 /* non-US #1 */
- #define **GLFW_KEY_WORLD_2** 162 /* non-US #2 */
- #define **GLFW_KEY_ESCAPE** 256
- #define **GLFW_KEY_ENTER** 257
- #define **GLFW_KEY_TAB** 258
- #define **GLFW_KEY_BACKSPACE** 259
- #define **GLFW_KEY_INSERT** 260
- #define **GLFW_KEY_DELETE** 261
- #define **GLFW_KEY_RIGHT** 262
- #define **GLFW_KEY_LEFT** 263
- #define **GLFW_KEY_DOWN** 264
- #define **GLFW_KEY_UP** 265
- #define **GLFW_KEY_PAGE_UP** 266
- #define **GLFW_KEY_PAGE_DOWN** 267
- #define **GLFW_KEY_HOME** 268
- #define **GLFW_KEY_END** 269
- #define **GLFW_KEY_CAPS_LOCK** 280
- #define **GLFW_KEY_SCROLL_LOCK** 281
- #define **GLFW_KEY_NUM_LOCK** 282
- #define **GLFW_KEY_PRINT_SCREEN** 283
- #define **GLFW_KEY_PAUSE** 284
- #define **GLFW_KEY_F1** 290
- #define **GLFW_KEY_F2** 291
- #define **GLFW_KEY_F3** 292
- #define **GLFW_KEY_F4** 293
- #define **GLFW_KEY_F5** 294
- #define **GLFW_KEY_F6** 295
- #define **GLFW_KEY_F7** 296
- #define **GLFW_KEY_F8** 297
- #define **GLFW_KEY_F9** 298
- #define **GLFW_KEY_F10** 299
- #define **GLFW_KEY_F11** 300
- #define **GLFW_KEY_F12** 301

- #define **GLFW_KEY_F13** 302
- #define **GLFW_KEY_F14** 303
- #define **GLFW_KEY_F15** 304
- #define **GLFW_KEY_F16** 305
- #define **GLFW_KEY_F17** 306
- #define **GLFW_KEY_F18** 307
- #define **GLFW_KEY_F19** 308
- #define **GLFW_KEY_F20** 309
- #define **GLFW_KEY_F21** 310
- #define **GLFW_KEY_F22** 311
- #define **GLFW_KEY_F23** 312
- #define **GLFW_KEY_F24** 313
- #define **GLFW_KEY_F25** 314
- #define **GLFW_KEY_KP_0** 320
- #define **GLFW_KEY_KP_1** 321
- #define **GLFW_KEY_KP_2** 322
- #define **GLFW_KEY_KP_3** 323
- #define **GLFW_KEY_KP_4** 324
- #define **GLFW_KEY_KP_5** 325
- #define **GLFW_KEY_KP_6** 326
- #define **GLFW_KEY_KP_7** 327
- #define **GLFW_KEY_KP_8** 328
- #define **GLFW_KEY_KP_9** 329
- #define **GLFW_KEY_KP_DECIMAL** 330
- #define **GLFW_KEY_KP_DIVIDE** 331
- #define **GLFW_KEY_KP_MULTIPLY** 332
- #define **GLFW_KEY_KP_SUBTRACT** 333
- #define **GLFW_KEY_KP_ADD** 334
- #define **GLFW_KEY_KP_ENTER** 335
- #define **GLFW_KEY_KP_EQUAL** 336
- #define **GLFW_KEY_LEFT_SHIFT** 340
- #define **GLFW_KEY_LEFT_CONTROL** 341
- #define **GLFW_KEY_LEFT_ALT** 342
- #define **GLFW_KEY_LEFT_SUPER** 343
- #define **GLFW_KEY_RIGHT_SHIFT** 344
- #define **GLFW_KEY_RIGHT_CONTROL** 345
- #define **GLFW_KEY_RIGHT_ALT** 346
- #define **GLFW_KEY_RIGHT_SUPER** 347
- #define **GLFW_KEY_MENU** 348
- #define **GLFW_KEY_LAST** GLFW_KEY_MENU
- #define GLFW_MOD_SHIFT 0x0001

  If this bit is set one or more Shift keys were held down.
- #define GLFW_MOD_CONTROL 0x0002

  If this bit is set one or more Control keys were held down.
- #define GLFW_MOD_ALT 0x0004

  If this bit is set one or more Alt keys were held down.
- #define GLFW_MOD_SUPER 0x0008

  If this bit is set one or more Super keys were held down.
- #define GLFW_MOD_CAPS_LOCK 0x0010

  If this bit is set the Caps Lock key is enabled.
- #define GLFW_MOD_NUM_LOCK 0x0020

  If this bit is set the Num Lock key is enabled.
- #define **GLFW_MOUSE_BUTTON_1** 0
- #define **GLFW_MOUSE_BUTTON_2** 1

- #define **GLFW_MOUSE_BUTTON_3** 2
- #define **GLFW_MOUSE_BUTTON_4** 3
- #define **GLFW_MOUSE_BUTTON_5** 4
- #define **GLFW_MOUSE_BUTTON_6** 5
- #define **GLFW_MOUSE_BUTTON_7** 6
- #define **GLFW_MOUSE_BUTTON_8** 7
- #define **GLFW_MOUSE_BUTTON_LAST** GLFW_MOUSE_BUTTON_8
- #define **GLFW_MOUSE_BUTTON_LEFT** GLFW_MOUSE_BUTTON_1
- #define **GLFW_MOUSE_BUTTON_RIGHT** GLFW_MOUSE_BUTTON_2
- #define **GLFW_MOUSE_BUTTON_MIDDLE** GLFW_MOUSE_BUTTON_3
- #define **GLFW_JOYSTICK_1** 0
- #define **GLFW_JOYSTICK_2** 1
- #define **GLFW_JOYSTICK_3** 2
- #define **GLFW_JOYSTICK_4** 3
- #define **GLFW_JOYSTICK_5** 4
- #define **GLFW_JOYSTICK_6** 5
- #define **GLFW_JOYSTICK_7** 6
- #define **GLFW_JOYSTICK_8** 7
- #define **GLFW_JOYSTICK_9** 8
- #define **GLFW_JOYSTICK_10** 9
- #define **GLFW_JOYSTICK_11** 10
- #define **GLFW_JOYSTICK_12** 11
- #define **GLFW_JOYSTICK_13** 12
- #define **GLFW_JOYSTICK_14** 13
- #define **GLFW_JOYSTICK_15** 14
- #define **GLFW_JOYSTICK_16** 15
- #define **GLFW_JOYSTICK_LAST** GLFW_JOYSTICK_16
- #define **GLFW_GAMEPAD_BUTTON_A** 0
- #define **GLFW_GAMEPAD_BUTTON_B** 1
- #define **GLFW_GAMEPAD_BUTTON_X** 2
- #define **GLFW_GAMEPAD_BUTTON_Y** 3
- #define **GLFW_GAMEPAD_BUTTON_LEFT_BUMPER** 4
- #define **GLFW_GAMEPAD_BUTTON_RIGHT_BUMPER** 5
- #define **GLFW_GAMEPAD_BUTTON_BACK** 6
- #define **GLFW_GAMEPAD_BUTTON_START** 7
- #define **GLFW_GAMEPAD_BUTTON_GUIDE** 8
- #define **GLFW_GAMEPAD_BUTTON_LEFT_THUMB** 9
- #define **GLFW_GAMEPAD_BUTTON_RIGHT_THUMB** 10
- #define **GLFW_GAMEPAD_BUTTON_DPAD_UP** 11
- #define **GLFW_GAMEPAD_BUTTON_DPAD_RIGHT** 12
- #define **GLFW_GAMEPAD_BUTTON_DPAD_DOWN** 13
- #define **GLFW_GAMEPAD_BUTTON_DPAD_LEFT** 14
- #define **GLFW_GAMEPAD_BUTTON_LAST** GLFW_GAMEPAD_BUTTON_DPAD_LEFT
- #define **GLFW_GAMEPAD_BUTTON_CROSS** GLFW_GAMEPAD_BUTTON_A
- #define **GLFW_GAMEPAD_BUTTON_CIRCLE** GLFW_GAMEPAD_BUTTON_B
- #define **GLFW_GAMEPAD_BUTTON_SQUARE** GLFW_GAMEPAD_BUTTON_X
- #define **GLFW_GAMEPAD_BUTTON_TRIANGLE** GLFW_GAMEPAD_BUTTON_Y
- #define **GLFW_GAMEPAD_AXIS_LEFT_X** 0
- #define **GLFW_GAMEPAD_AXIS_LEFT_Y** 1
- #define **GLFW_GAMEPAD_AXIS_RIGHT_X** 2
- #define **GLFW_GAMEPAD_AXIS_RIGHT_Y** 3
- #define **GLFW_GAMEPAD_AXIS_LEFT_TRIGGER** 4
- #define **GLFW_GAMEPAD_AXIS_RIGHT_TRIGGER** 5
- #define **GLFW_GAMEPAD_AXIS_LAST** GLFW_GAMEPAD_AXIS_RIGHT_TRIGGER
- #define GLFW_NO_ERROR 0

No error has occurred.

- #define GLFW_NOT_INITIALIZED 0x00010001

  GLFW has not been initialized.
- #define GLFW_NO_CURRENT_CONTEXT 0x00010002

  No context is current for this thread.
- #define GLFW_INVALID_ENUM 0x00010003

  One of the arguments to the function was an invalid enum value.
- #define GLFW_INVALID_VALUE 0x00010004

  One of the arguments to the function was an invalid value.
- #define GLFW_OUT_OF_MEMORY 0x00010005

  A memory allocation failed.
- #define GLFW_API_UNAVAILABLE 0x00010006

  GLFW could not find support for the requested API on the system.
- #define GLFW_VERSION_UNAVAILABLE 0x00010007

  The requested OpenGL or OpenGL ES version is not available.
- #define GLFW_PLATFORM_ERROR 0x00010008

  A platform-specific error occurred that does not match any of the more specific categories.
- #define GLFW_FORMAT_UNAVAILABLE 0x00010009

  The requested format is not supported or available.
- #define GLFW_NO_WINDOW_CONTEXT 0x0001000A

  The specified window does not have an OpenGL or OpenGL ES context.
- #define GLFW_CURSOR_UNAVAILABLE 0x0001000B

  The specified cursor shape is not available.
- #define GLFW_FEATURE_UNAVAILABLE 0x0001000C

  The requested feature is not provided by the platform.
- #define GLFW_FEATURE_UNIMPLEMENTED 0x0001000D

  The requested feature is not implemented for the platform.
- #define GLFW_PLATFORM_UNAVAILABLE 0x0001000E

  Platform unavailable or no matching platform was found.
- #define GLFW_FOCUSED 0x00020001

  Input focus window hint and attribute.
- #define GLFW_ICONIFIED 0x00020002

  Window iconification window attribute.
- #define GLFW_RESIZABLE 0x00020003

  Window resize-ability window hint and attribute.
- #define GLFW_VISIBLE 0x00020004

  Window visibility window hint and attribute.
- #define GLFW_DECORATED 0x00020005

  Window decoration window hint and attribute.
- #define GLFW_AUTO_ICONIFY 0x00020006

  Window auto-iconification window hint and attribute.
- #define GLFW_FLOATING 0x00020007

  Window decoration window hint and attribute.
- #define GLFW_MAXIMIZED 0x00020008

  Window maximization window hint and attribute.
- #define GLFW_CENTER_CURSOR 0x00020009

  Cursor centering window hint.
- #define GLFW_TRANSPARENT_FRAMEBUFFER 0x0002000A

  Window framebuffer transparency hint and attribute.
- #define GLFW_HOVERED 0x0002000B

  Mouse cursor hover window attribute.

- #define GLFW_FOCUS_ON_SHOW 0x0002000C

  Input focus on calling show window hint and attribute.
- #define GLFW_MOUSE_PASSTHROUGH 0x0002000D

  Mouse input transparency window hint and attribute.
- #define GLFW_RED_BITS 0x00021001

  Framebuffer bit depth hint.
- #define GLFW_GREEN_BITS 0x00021002

  Framebuffer bit depth hint.
- #define GLFW_BLUE_BITS 0x00021003

  Framebuffer bit depth hint.
- #define GLFW_ALPHA_BITS 0x00021004

  Framebuffer bit depth hint.
- #define GLFW_DEPTH_BITS 0x00021005

  Framebuffer bit depth hint.
- #define GLFW_STENCIL_BITS 0x00021006

  Framebuffer bit depth hint.
- #define GLFW_ACCUM_RED_BITS 0x00021007

  Framebuffer bit depth hint.
- #define GLFW_ACCUM_GREEN_BITS 0x00021008

  Framebuffer bit depth hint.
- #define GLFW_ACCUM_BLUE_BITS 0x00021009

  Framebuffer bit depth hint.
- #define GLFW_ACCUM_ALPHA_BITS 0x0002100A

  Framebuffer bit depth hint.
- #define GLFW_AUX_BUFFERS 0x0002100B

  Framebuffer auxiliary buffer hint.
- #define GLFW_STEREO 0x0002100C

  OpenGL stereoscopic rendering hint.
- #define GLFW_SAMPLES 0x0002100D

  Framebuffer MSAA samples hint.
- #define GLFW_SRGB_CAPABLE 0x0002100E

  Framebuffer sRGB hint.
- #define GLFW_REFRESH_RATE 0x0002100F

  Monitor refresh rate hint.
- #define GLFW_DOUBLEBUFFER 0x00021010

  Framebuffer double buffering hint and attribute.
- #define GLFW_CLIENT_API 0x00022001

  Context client API hint and attribute.
- #define GLFW_CONTEXT_VERSION_MAJOR 0x00022002

  Context client API major version hint and attribute.
- #define GLFW_CONTEXT_VERSION_MINOR 0x00022003

  Context client API minor version hint and attribute.
- #define GLFW_CONTEXT_REVISION 0x00022004

  Context client API revision number attribute.
- #define GLFW_CONTEXT_ROBUSTNESS 0x00022005

  Context robustness hint and attribute.
- #define GLFW_OPENGL_FORWARD_COMPAT 0x00022006

  OpenGL forward-compatibility hint and attribute.
- #define GLFW_CONTEXT_DEBUG 0x00022007

  Debug mode context hint and attribute.
- #define GLFW_OPENGL_DEBUG_CONTEXT GLFW_CONTEXT_DEBUG

Legacy name for compatibility.

- #define GLFW_OPENGL_PROFILE 0x00022008

  OpenGL profile hint and attribute.
- #define GLFW_CONTEXT_RELEASE_BEHAVIOR 0x00022009

  Context flush-on-release hint and attribute.
- #define GLFW_CONTEXT_NO_ERROR 0x0002200A

  Context error suppression hint and attribute.
- #define GLFW_CONTEXT_CREATION_API 0x0002200B

  Context creation API hint and attribute.
- #define **GLFW_SCALE_TO_MONITOR** 0x0002200C

  Window content area scaling window window hint.
- #define **GLFW_COCOA_RETINA_FRAMEBUFFER** 0x00023001

  macOS specific window hint.
- #define **GLFW_COCOA_FRAME_NAME** 0x00023002

  macOS specific window hint.
- #define **GLFW_COCOA_GRAPHICS_SWITCHING** 0x00023003

  macOS specific window hint.
- #define **GLFW_X11_CLASS_NAME** 0x00024001

  X11 specific window hint.
- #define **GLFW_X11_INSTANCE_NAME** 0x00024002

  X11 specific window hint.
- #define **GLFW_WIN32_KEYBOARD_MENU** 0x00025001
- #define **GLFW_NO_API** 0
- #define **GLFW_OPENGL_API** 0x00030001
- #define **GLFW_OPENGL_ES_API** 0x00030002
- #define **GLFW_NO_ROBUSTNESS** 0
- #define **GLFW_NO_RESET_NOTIFICATION** 0x00031001
- #define **GLFW_LOSE_CONTEXT_ON_RESET** 0x00031002
- #define **GLFW_OPENGL_ANY_PROFILE** 0
- #define **GLFW_OPENGL_CORE_PROFILE** 0x00032001
- #define **GLFW_OPENGL_COMPAT_PROFILE** 0x00032002
- #define **GLFW_CURSOR** 0x00033001
- #define **GLFW_STICKY_KEYS** 0x00033002
- #define **GLFW_STICKY_MOUSE_BUTTONS** 0x00033003
- #define **GLFW_LOCK_KEY_MODS** 0x00033004
- #define **GLFW_RAW_MOUSE_MOTION** 0x00033005
- #define **GLFW_CURSOR_NORMAL** 0x00034001
- #define **GLFW_CURSOR_HIDDEN** 0x00034002
- #define **GLFW_CURSOR_DISABLED** 0x00034003
- #define **GLFW_ANY_RELEASE_BEHAVIOR** 0
- #define **GLFW_RELEASE_BEHAVIOR_FLUSH** 0x00035001
- #define **GLFW_RELEASE_BEHAVIOR_NONE** 0x00035002
- #define **GLFW_NATIVE_CONTEXT_API** 0x00036001
- #define **GLFW_EGL_CONTEXT_API** 0x00036002
- #define **GLFW_OSMESA_CONTEXT_API** 0x00036003
- #define **GLFW_ANGLE_PLATFORM_TYPE_NONE** 0x00037001
- #define **GLFW_ANGLE_PLATFORM_TYPE_OPENGL** 0x00037002
- #define **GLFW_ANGLE_PLATFORM_TYPE_OPENGLES** 0x00037003
- #define **GLFW_ANGLE_PLATFORM_TYPE_D3D9** 0x00037004
- #define **GLFW_ANGLE_PLATFORM_TYPE_D3D11** 0x00037005
- #define **GLFW_ANGLE_PLATFORM_TYPE_VULKAN** 0x00037007
- #define **GLFW_ANGLE_PLATFORM_TYPE_METAL** 0x00037008
- #define GLFW_ARROW_CURSOR 0x00036001

The regular arrow cursor shape.

- #define GLFW_IBEAM_CURSOR 0x00036002

  The text input I-beam cursor shape.
- #define GLFW_CROSSHAIR_CURSOR 0x00036003

  The crosshair cursor shape.
- #define GLFW_POINTING_HAND_CURSOR 0x00036004

  The pointing hand cursor shape.
- #define GLFW_RESIZE_EW_CURSOR 0x00036005

  The horizontal resize/move arrow shape.
- #define GLFW_RESIZE_NS_CURSOR 0x00036006

  The vertical resize/move arrow shape.
- #define GLFW_RESIZE_NWSE_CURSOR 0x00036007

  The top-left to bottom-right diagonal resize/move arrow shape.
- #define GLFW_RESIZE_NESW_CURSOR 0x00036008

  The top-right to bottom-left diagonal resize/move arrow shape.
- #define GLFW_RESIZE_ALL_CURSOR 0x00036009

  The omni-directional resize/move cursor shape.
- #define GLFW_NOT_ALLOWED_CURSOR 0x0003600A

  The operation-not-allowed shape.
- #define GLFW_HRESIZE_CURSOR GLFW_RESIZE_EW_CURSOR

  Legacy name for compatibility.
- #define GLFW_VRESIZE_CURSOR GLFW_RESIZE_NS_CURSOR

  Legacy name for compatibility.
- #define GLFW_HAND_CURSOR GLFW_POINTING_HAND_CURSOR

  Legacy name for compatibility.
- #define **GLFW_CONNECTED** 0x00040001
- #define **GLFW_DISCONNECTED** 0x00040002
- #define GLFW_JOYSTICK_HAT_BUTTONS 0x00050001

  Joystick hat buttons init hint.
- #define GLFW_ANGLE_PLATFORM_TYPE 0x00050002

  ANGLE rendering backend init hint.
- #define GLFW_PLATFORM 0x00050003

  Platform selection init hint.
- #define GLFW_COCOA_CHDIR_RESOURCES 0x00051001

  macOS specific init hint.
- #define GLFW_COCOA_MENUBAR 0x00051002

  macOS specific init hint.
- #define GLFW_X11_XCB_VULKAN_SURFACE 0x00052001

  X11 specific init hint.
- #define GLFW_ANY_PLATFORM 0x00060000

  Hint value that enables automatic platform selection.
- #define **GLFW_PLATFORM_WIN32** 0x00060001
- #define **GLFW_PLATFORM_COCOA** 0x00060002
- #define **GLFW_PLATFORM_WAYLAND** 0x00060003
- #define **GLFW_PLATFORM_X11** 0x00060004
- #define **GLFW_PLATFORM_NULL** 0x00060005
- #define **GLFW_DONT_CARE** -1
- #define **GLAPIENTRY** APIENTRY

GLFW version macros

- #define GLFW_VERSION_MAJOR 3
  The major version number of the GLFW header.
- #define GLFW_VERSION_MINOR 4
  The minor version number of the GLFW header.
- #define GLFW_VERSION_REVISION 0
  The revision number of the GLFW header.

Key and button actions

- #define GLFW_RELEASE 0
  The key or mouse button was released.
- #define GLFW_PRESS 1
  The key or mouse button was pressed.
- #define GLFW_REPEAT 2
  The key was held down until it repeated.

Typedefs

- typedef void(* GLFWglproc) (void)
  Client API function pointer type.
- typedef void(* GLFWvkproc) (void)
  Vulkan API function pointer type.
- typedef struct GLFWmonitor GLFWmonitor
  Opaque monitor object.
- typedef struct GLFWwindow GLFWwindow
  Opaque window object.
- typedef struct GLFWcursor GLFWcursor
  Opaque cursor object.
- typedef void *(* GLFWallocatefun) (size_t size, void *user)
  The function pointer type for memory allocation callbacks.
- typedef void *(* GLFWreallocatefun) (void *block, size_t size, void *user)
  The function pointer type for memory reallocation callbacks.
- typedef void(* GLFWdeallocatefun) (void *block, void *user)
  The function pointer type for memory deallocation callbacks.
- typedef void(* GLFWerrorfun) (int error_code, const char *description)
  The function pointer type for error callbacks.
- typedef void(* GLFWwindowposfun) (GLFWwindow *window, int xpos, int ypos)
  The function pointer type for window position callbacks.
- typedef void(* GLFWwindowsizefun) (GLFWwindow *window, int width, int height)
  The function pointer type for window size callbacks.
- typedef void(* GLFWwindowclosefun) (GLFWwindow *window)
  The function pointer type for window close callbacks.
- typedef void(* GLFWwindowrefreshfun) (GLFWwindow *window)
  The function pointer type for window content refresh callbacks.
- typedef void(* GLFWwindowfocusfun) (GLFWwindow *window, int focused)
  The function pointer type for window focus callbacks.
- typedef void(* GLFWwindowiconifyfun) (GLFWwindow *window, int iconified)
  The function pointer type for window iconify callbacks.
- typedef void(* GLFWwindowmaximizefun) (GLFWwindow *window, int maximized)
  The function pointer type for window maximize callbacks.
- typedef void(* GLFWframebuffersizefun) (GLFWwindow *window, int width, int height)

The function pointer type for framebuffer size callbacks.

- typedef void(* GLFWwindowcontentscalefun) (GLFWwindow *window, float xscale, float yscale)

  The function pointer type for window content scale callbacks.
- typedef void(* GLFWmousebuttonfun) (GLFWwindow *window, int button, int action, int mods)

  The function pointer type for mouse button callbacks.
- typedef void(* GLFWcursorposfun) (GLFWwindow *window, double xpos, double ypos)

  The function pointer type for cursor position callbacks.
- typedef void(* GLFWcursorenterfun) (GLFWwindow *window, int entered)

  The function pointer type for cursor enter/leave callbacks.
- typedef void(* GLFWscrollfun) (GLFWwindow *window, double xoffset, double yoffset)

  The function pointer type for scroll callbacks.
- typedef void(* GLFWkeyfun) (GLFWwindow *window, int key, int scancode, int action, int mods)

  The function pointer type for keyboard key callbacks.
- typedef void(* GLFWcharfun) (GLFWwindow *window, unsigned int codepoint)

  The function pointer type for Unicode character callbacks.
- typedef void(* GLFWcharmodsfun) (GLFWwindow *window, unsigned int codepoint, int mods)

  The function pointer type for Unicode character with modifiers callbacks.
- typedef void(* GLFWdropfun) (GLFWwindow *window, int path_count, const char *paths[])

  The function pointer type for path drop callbacks.
- typedef void(* GLFWmonitorfun) (GLFWmonitor *monitor, int event)

  The function pointer type for monitor configuration callbacks.
- typedef void(* GLFWjoystickfun) (int jid, int event)

  The function pointer type for joystick configuration callbacks.
- typedef struct GLFWvidmode GLFWvidmode

  Video mode type.
- typedef struct GLFWgammaramp GLFWgammaramp

  Gamma ramp.
- typedef struct GLFWimage GLFWimage

  Image data.
- typedef struct GLFWgamepadstate GLFWgamepadstate

  Gamepad input state.
- typedef struct GLFWallocator GLFWallocator

Functions

- GLFWAPI int glfwInit (void)

  Initializes the GLFW library.
- GLFWAPI void glfwTerminate (void)

  Terminates the GLFW library.
- GLFWAPI void glfwInitHint (int hint, int value)

  Sets the specified init hint to the desired value.
- GLFWAPI void glfwInitAllocator (const GLFWallocator *allocator)

  Sets the init allocator to the desired value.
- GLFWAPI void glfwGetVersion (int *major, int *minor, int *rev)

  Retrieves the version of the GLFW library.
- GLFWAPI const char * glfwGetVersionString (void)

  Returns a string describing the compile-time configuration.
- GLFWAPI int glfwGetError (const char **description)

  Returns and clears the last error for the calling thread.
- GLFWAPI GLFWerrorfun glfwSetErrorCallback (GLFWerrorfun callback)

Sets the error callback.

- GLFWAPI int glfwGetPlatform (void)

  Returns the currently selected platform.
- GLFWAPI int glfwPlatformSupported (int platform)

  Returns whether the library includes support for the specified platform.
- GLFWAPI GLFWmonitor ** glfwGetMonitors (int *count)

  Returns the currently connected monitors.
- GLFWAPI GLFWmonitor * glfwGetPrimaryMonitor (void)

  Returns the primary monitor.
- GLFWAPI void glfwGetMonitorPos (GLFWmonitor *monitor, int *xpos, int *ypos)

  Returns the position of the monitor's viewport on the virtual screen.
- GLFWAPI void glfwGetMonitorWorkarea (GLFWmonitor *monitor, int *xpos, int *ypos, int *width, int *height)

  Retrieves the work area of the monitor.
- GLFWAPI void glfwGetMonitorPhysicalSize (GLFWmonitor *monitor, int *widthMM, int *heightMM)

  Returns the physical size of the monitor.
- GLFWAPI void glfwGetMonitorContentScale (GLFWmonitor *monitor, float *xscale, float *yscale)

  Retrieves the content scale for the specified monitor.
- GLFWAPI const char * glfwGetMonitorName (GLFWmonitor *monitor)

  Returns the name of the specified monitor.
- GLFWAPI void glfwSetMonitorUserPointer (GLFWmonitor *monitor, void *pointer)

  Sets the user pointer of the specified monitor.
- GLFWAPI void * glfwGetMonitorUserPointer (GLFWmonitor *monitor)

  Returns the user pointer of the specified monitor.
- GLFWAPI GLFWmonitorfun glfwSetMonitorCallback (GLFWmonitorfun callback)

  Sets the monitor configuration callback.
- GLFWAPI const GLFWvidmode * glfwGetVideoModes (GLFWmonitor *monitor, int *count)

  Returns the available video modes for the specified monitor.
- GLFWAPI const GLFWvidmode * glfwGetVideoMode (GLFWmonitor *monitor)

  Returns the current mode of the specified monitor.
- GLFWAPI void glfwSetGamma (GLFWmonitor *monitor, float gamma)

  Generates a gamma ramp and sets it for the specified monitor.
- GLFWAPI const GLFWgammaramp * glfwGetGammaRamp (GLFWmonitor *monitor)

  Returns the current gamma ramp for the specified monitor.
- GLFWAPI void glfwSetGammaRamp (GLFWmonitor *monitor, const GLFWgammaramp *ramp)

  Sets the current gamma ramp for the specified monitor.
- GLFWAPI void glfwDefaultWindowHints (void)

  Resets all window hints to their default values.
- GLFWAPI void glfwWindowHint (int hint, int value)

  Sets the specified window hint to the desired value.
- GLFWAPI void glfwWindowHintString (int hint, const char *value)

  Sets the specified window hint to the desired value.
- GLFWAPI GLFWwindow * glfwCreateWindow (int width, int height, const char *title, GLFWmonitor *monitor, GLFWwindow *share)

  Creates a window and its associated context.
- GLFWAPI void glfwDestroyWindow (GLFWwindow *window)

  Destroys the specified window and its context.
- GLFWAPI int glfwWindowShouldClose (GLFWwindow *window)

  Checks the close flag of the specified window.
- GLFWAPI void glfwSetWindowShouldClose (GLFWwindow *window, int value)

  Sets the close flag of the specified window.
- GLFWAPI void glfwSetWindowTitle (GLFWwindow *window, const char *title)

Sets the title of the specified window.

- GLFWAPI void glfwSetWindowIcon (GLFWwindow *window, int count, const GLFWimage *images)

  Sets the icon for the specified window.
- GLFWAPI void glfwGetWindowPos (GLFWwindow *window, int *xpos, int *ypos)

  Retrieves the position of the content area of the specified window.
- GLFWAPI void glfwSetWindowPos (GLFWwindow *window, int xpos, int ypos)

  Sets the position of the content area of the specified window.
- GLFWAPI void glfwGetWindowSize (GLFWwindow *window, int *width, int *height)

  Retrieves the size of the content area of the specified window.
- GLFWAPI void glfwSetWindowSizeLimits (GLFWwindow *window, int minwidth, int minheight, int maxwidth, int maxheight)

  Sets the size limits of the specified window.
- GLFWAPI void glfwSetWindowAspectRatio (GLFWwindow *window, int numer, int denom)

  Sets the aspect ratio of the specified window.
- GLFWAPI void glfwSetWindowSize (GLFWwindow *window, int width, int height)

  Sets the size of the content area of the specified window.
- GLFWAPI void glfwGetFramebufferSize (GLFWwindow *window, int *width, int *height)

  Retrieves the size of the framebuffer of the specified window.
- GLFWAPI void glfwGetWindowFrameSize (GLFWwindow *window, int *left, int *top, int *right, int *bottom)

  Retrieves the size of the frame of the window.
- GLFWAPI void glfwGetWindowContentScale (GLFWwindow *window, float *xscale, float *yscale)

  Retrieves the content scale for the specified window.
- GLFWAPI float glfwGetWindowOpacity (GLFWwindow *window)

  Returns the opacity of the whole window.
- GLFWAPI void glfwSetWindowOpacity (GLFWwindow *window, float opacity)

  Sets the opacity of the whole window.
- GLFWAPI void glfwIconifyWindow (GLFWwindow *window)

  Iconifies the specified window.
- GLFWAPI void glfwRestoreWindow (GLFWwindow *window)

  Restores the specified window.
- GLFWAPI void glfwMaximizeWindow (GLFWwindow *window)

  Maximizes the specified window.
- GLFWAPI void glfwShowWindow (GLFWwindow *window)

  Makes the specified window visible.
- GLFWAPI void glfwHideWindow (GLFWwindow *window)

  Hides the specified window.
- GLFWAPI void glfwFocusWindow (GLFWwindow *window)

  Brings the specified window to front and sets input focus.
- GLFWAPI void glfwRequestWindowAttention (GLFWwindow *window)

  Requests user attention to the specified window.
- GLFWAPI GLFWmonitor * glfwGetWindowMonitor (GLFWwindow *window)

  Returns the monitor that the window uses for full screen mode.
- GLFWAPI void glfwSetWindowMonitor (GLFWwindow *window, GLFWmonitor *monitor, int xpos, int ypos, int width, int height, int refreshRate)

  Sets the mode, monitor, video mode and placement of a window.
- GLFWAPI int glfwGetWindowAttrib (GLFWwindow *window, int attrib)

  Returns an attribute of the specified window.
- GLFWAPI void glfwSetWindowAttrib (GLFWwindow *window, int attrib, int value)

  Sets an attribute of the specified window.
- GLFWAPI void glfwSetWindowUserPointer (GLFWwindow *window, void *pointer)

  Sets the user pointer of the specified window.

- GLFWAPI void * glfwGetWindowUserPointer (GLFWwindow *window)

  Returns the user pointer of the specified window.
- GLFWAPI GLFWwindowposfun glfwSetWindowPosCallback (GLFWwindow *window, GLFWwindowposfun callback)

  Sets the position callback for the specified window.
- GLFWAPI GLFWwindowsizefun glfwSetWindowSizeCallback (GLFWwindow *window, GLFWwindowsizefun callback)

  Sets the size callback for the specified window.
- GLFWAPI GLFWwindowclosefun glfwSetWindowCloseCallback (GLFWwindow *window, GLFWwindowclosefun callback)

  Sets the close callback for the specified window.
- GLFWAPI GLFWwindowrefreshfun glfwSetWindowRefreshCallback (GLFWwindow *window, GLFWwindowrefreshfun callback)

  Sets the refresh callback for the specified window.
- GLFWAPI GLFWwindowfocusfun glfwSetWindowFocusCallback (GLFWwindow *window, GLFWwindowfocusfun callback)

  Sets the focus callback for the specified window.
- GLFWAPI GLFWwindowiconifyfun glfwSetWindowIconifyCallback (GLFWwindow *window, GLFWwindowiconifyfun callback)

  Sets the iconify callback for the specified window.
- GLFWAPI GLFWwindowmaximizefun glfwSetWindowMaximizeCallback (GLFWwindow *window, GLFWwindowmaximizefun callback)

  Sets the maximize callback for the specified window.
- GLFWAPI GLFWframebuffersizefun glfwSetFramebufferSizeCallback (GLFWwindow *window, GLFWframebuffersizefun callback)

  Sets the framebuffer resize callback for the specified window.
- GLFWAPI GLFWwindowcontentscalefun glfwSetWindowContentScaleCallback (GLFWwindow *window, GLFWwindowcontentscalefun callback)

  Sets the window content scale callback for the specified window.
- GLFWAPI void glfwPollEvents (void)

  Processes all pending events.
- GLFWAPI void glfwWaitEvents (void)

  Waits until events are queued and processes them.
- GLFWAPI void glfwWaitEventsTimeout (double timeout)

  Waits with timeout until events are queued and processes them.
- GLFWAPI void glfwPostEmptyEvent (void)

  Posts an empty event to the event queue.
- GLFWAPI int glfwGetInputMode (GLFWwindow *window, int mode)

  Returns the value of an input option for the specified window.
- GLFWAPI void glfwSetInputMode (GLFWwindow *window, int mode, int value)

  Sets an input option for the specified window.
- GLFWAPI int glfwRawMouseMotionSupported (void)

  Returns whether raw mouse motion is supported.
- GLFWAPI const char * glfwGetKeyName (int key, int scancode)

  Returns the layout-specific name of the specified printable key.
- GLFWAPI int glfwGetKeyScancode (int key)

  Returns the platform-specific scancode of the specified key.
- GLFWAPI int glfwGetKey (GLFWwindow *window, int key)

  Returns the last reported state of a keyboard key for the specified window.
- GLFWAPI int glfwGetMouseButton (GLFWwindow *window, int button)

  Returns the last reported state of a mouse button for the specified window.
- GLFWAPI void glfwGetCursorPos (GLFWwindow *window, double *xpos, double *ypos)

Retrieves the position of the cursor relative to the content area of the window.

- GLFWAPI void glfwSetCursorPos (GLFWwindow *window, double xpos, double ypos)

  Sets the position of the cursor, relative to the content area of the window.
- GLFWAPI GLFWcursor * glfwCreateCursor (const GLFWimage *image, int xhot, int yhot)

  Creates a custom cursor.
- GLFWAPI GLFWcursor * glfwCreateStandardCursor (int shape)

  Creates a cursor with a standard shape.
- GLFWAPI void glfwDestroyCursor (GLFWcursor *cursor)

  Destroys a cursor.
- GLFWAPI void glfwSetCursor (GLFWwindow *window, GLFWcursor *cursor)

  Sets the cursor for the window.
- GLFWAPI GLFWkeyfun glfwSetKeyCallback (GLFWwindow *window, GLFWkeyfun callback)

  Sets the key callback.
- GLFWAPI GLFWcharfun glfwSetCharCallback (GLFWwindow *window, GLFWcharfun callback)

  Sets the Unicode character callback.
- GLFWAPI GLFWcharmodsfun glfwSetCharModsCallback (GLFWwindow *window, GLFWcharmodsfun callback)

  Sets the Unicode character with modifiers callback.
- GLFWAPI GLFWmousebuttonfun glfwSetMouseButtonCallback (GLFWwindow *window, GLFWmousebuttonfun callback)

  Sets the mouse button callback.
- GLFWAPI GLFWcursorposfun glfwSetCursorPosCallback (GLFWwindow *window, GLFWcursorposfun callback)

  Sets the cursor position callback.
- GLFWAPI GLFWcursorenterfun glfwSetCursorEnterCallback (GLFWwindow *window, GLFWcursorenterfun callback)

  Sets the cursor enter/leave callback.
- GLFWAPI GLFWscrollfun glfwSetScrollCallback (GLFWwindow *window, GLFWscrollfun callback)

  Sets the scroll callback.
- GLFWAPI GLFWdropfun glfwSetDropCallback (GLFWwindow *window, GLFWdropfun callback)

  Sets the path drop callback.
- GLFWAPI int glfwJoystickPresent (int jid)

  Returns whether the specified joystick is present.
- GLFWAPI const float * glfwGetJoystickAxes (int jid, int *count)

  Returns the values of all axes of the specified joystick.
- GLFWAPI const unsigned char * glfwGetJoystickButtons (int jid, int *count)

  Returns the state of all buttons of the specified joystick.
- GLFWAPI const unsigned char * glfwGetJoystickHats (int jid, int *count)

  Returns the state of all hats of the specified joystick.
- GLFWAPI const char * glfwGetJoystickName (int jid)

  Returns the name of the specified joystick.
- GLFWAPI const char * glfwGetJoystickGUID (int jid)

  Returns the SDL compatible GUID of the specified joystick.
- GLFWAPI void glfwSetJoystickUserPointer (int jid, void *pointer)

  Sets the user pointer of the specified joystick.
- GLFWAPI void * glfwGetJoystickUserPointer (int jid)

  Returns the user pointer of the specified joystick.
- GLFWAPI int glfwJoystickIsGamepad (int jid)

  Returns whether the specified joystick has a gamepad mapping.
- GLFWAPI GLFWjoystickfun glfwSetJoystickCallback (GLFWjoystickfun callback)

  Sets the joystick configuration callback.

- GLFWAPI int glfwUpdateGamepadMappings (const char *string)

  Adds the specified SDL_GameControllerDB gamepad mappings.
- GLFWAPI const char * glfwGetGamepadName (int jid)

  Returns the human-readable gamepad name for the specified joystick.
- GLFWAPI int glfwGetGamepadState (int jid, GLFWgamepadstate *state)

  Retrieves the state of the specified joystick remapped as a gamepad.
- GLFWAPI void glfwSetClipboardString (GLFWwindow *window, const char *string)

  Sets the clipboard to the specified string.
- GLFWAPI const char * glfwGetClipboardString (GLFWwindow *window)

  Returns the contents of the clipboard as a string.
- GLFWAPI double glfwGetTime (void)

  Returns the GLFW time.
- GLFWAPI void glfwSetTime (double time)

  Sets the GLFW time.
- GLFWAPI uint64_t glfwGetTimerValue (void)

  Returns the current value of the raw timer.
- GLFWAPI uint64_t glfwGetTimerFrequency (void)

  Returns the frequency, in Hz, of the raw timer.
- GLFWAPI void glfwMakeContextCurrent (GLFWwindow *window)

  Makes the context of the specified window current for the calling thread.
- GLFWAPI GLFWwindow * glfwGetCurrentContext (void)

  Returns the window whose context is current on the calling thread.
- GLFWAPI void glfwSwapBuffers (GLFWwindow *window)

  Swaps the front and back buffers of the specified window.
- GLFWAPI void glfwSwapInterval (int interval)

  Sets the swap interval for the current context.
- GLFWAPI int glfwExtensionSupported (const char *extension)

  Returns whether the specified extension is available.
- GLFWAPI GLFWglproc glfwGetProcAddress (const char *procname)

  Returns the address of the specified function for the current context.
- GLFWAPI int glfwVulkanSupported (void)

  Returns whether the Vulkan loader and an ICD have been found.
- GLFWAPI const char ** glfwGetRequiredInstanceExtensions (uint32_t *count)

  Returns the Vulkan instance extensions required by GLFW.

27.15.1 Detailed Description

The header of the GLFW 3 API.

This is the header file of the GLFW 3 API. It defines all its types and declares all its functions.

For more information about how to use this file, see Including the GLFW header file.

27.16 glfw3.h

Go to the documentation of this file.

```
1 /*************************************************************************
2  * GLFW 3.4 - www.glfw.org
3  * A library for OpenGL, window and input
4  *------------------------------------------------------------------------
5  * Copyright (c) 2002-2006 Marcus Geelnard
6  * Copyright (c) 2006-2019 Camilla Löwy <elmindreda@glfw.org>
7  *
8  * This software is provided 'as-is', without any express or implied
9  * warranty. In no event will the authors be held liable for any damages
10  * arising from the use of this software.
11  *
12  * Permission is granted to anyone to use this software for any purpose,
13  * including commercial applications, and to alter it and redistribute it
14  * freely, subject to the following restrictions:
15  *
16  * 1. The origin of this software must not be misrepresented; you must not
17  *    claim that you wrote the original software. If you use this software
18  *    in a product, an acknowledgment in the product documentation would
19  *    be appreciated but is not required.
20  *
21  * 2. Altered source versions must be plainly marked as such, and must not
22  *    be misrepresented as being the original software.
23  *
24  * 3. This notice may not be removed or altered from any source
25  *    distribution.
26  *
27  *************************************************************************/
28
29 #ifndef _glfw3_h_
30 #define _glfw3_h_
31
32 #ifdef __cplusplus
33 extern "C" {
34 #endif
35
36
37 /*************************************************************************
38  * Doxygen documentation
39  *************************************************************************/
40
89 /*************************************************************************
90  * Compiler- and platform-specific preprocessor work
91  *************************************************************************/
92
93 /* If we are we on Windows, we want a single define for it.
94  */
95 #if !defined(_WIN32) && (defined(__WIN32__) || defined(WIN32) || defined(__MINGW32__))
96  #define _WIN32
97 #endif /* _WIN32 */
98
99 /* Include because most Windows GLU headers need wchar_t and
100  * the macOS OpenGL header blocks the definition of ptrdiff_t by glext.h.
101  * Include it unconditionally to avoid surprising side-effects.
102  */
103 #include <stddef.h>
104
105 /* Include because it is needed by Vulkan and related functions.
106  * Include it unconditionally to avoid surprising side-effects.
107  */
108 #include <stdint.h>
109
110 #if defined(GLFW_INCLUDE_VULKAN)
111   #include <vulkan/vulkan.h>
112 #endif /* Vulkan header */
113
114 /* The Vulkan header may have indirectly included windows.h (because of
115  * VK_USE_PLATFORM_WIN32_KHR) so we offer our replacement symbols after it.
116  */
117
118 /* It is customary to use APIENTRY for OpenGL function pointer declarations on
119  * all platforms.  Additionally, the Windows OpenGL header needs APIENTRY.
120  */
121 #if !defined(APIENTRY)
122  #if defined(_WIN32)
123   #define APIENTRY __stdcall
124  #else
125   #define APIENTRY
126  #endif
127  #define GLFW_APIENTRY_DEFINED
128 #endif /* APIENTRY */
129
130 /* Some Windows OpenGL headers need this.
```

```
131  */
132 #if !defined(WINGDIAPI) && defined(_WIN32)
133  #define WINGDIAPI __declspec(dllimport)
134  #define GLFW_WINGDIAPI_DEFINED
135 #endif /* WINGDIAPI */
136
137 /* Some Windows GLU headers need this.
138  */
139 #if !defined(CALLBACK) && defined(_WIN32)
140  #define CALLBACK __stdcall
141  #define GLFW_CALLBACK_DEFINED
142 #endif /* CALLBACK */
143
144 /* Include the chosen OpenGL or OpenGL ES headers.
145  */
146 #if defined(GLFW_INCLUDE_ES1)
147
148  #include <GLES/gl.h>
149  #if defined(GLFW_INCLUDE_GLEXT)
150   #include <GLES/glext.h>
151  #endif
152
153 #elif defined(GLFW_INCLUDE_ES2)
154
155  #include <GLES2/gl2.h>
156  #if defined(GLFW_INCLUDE_GLEXT)
157   #include <GLES2/gl2ext.h>
158  #endif
159
160 #elif defined(GLFW_INCLUDE_ES3)
161
162  #include <GLES3/gl3.h>
163  #if defined(GLFW_INCLUDE_GLEXT)
164   #include <GLES2/gl2ext.h>
165  #endif
166
167 #elif defined(GLFW_INCLUDE_ES31)
168
169  #include <GLES3/gl31.h>
170  #if defined(GLFW_INCLUDE_GLEXT)
171   #include <GLES2/gl2ext.h>
172  #endif
173
174 #elif defined(GLFW_INCLUDE_ES32)
175
176  #include <GLES3/gl32.h>
177  #if defined(GLFW_INCLUDE_GLEXT)
178   #include <GLES2/gl2ext.h>
179  #endif
180
181 #elif defined(GLFW_INCLUDE_GLCOREARB)
182
183  #if defined(__APPLE__)
184
185   #include <OpenGL/gl3.h>
186   #if defined(GLFW_INCLUDE_GLEXT)
187    #include <OpenGL/gl3ext.h>
188   #endif /*GLFW_INCLUDE_GLEXT*/
189
190  #else /*__APPLE__*/
191
192   #include <GL/glcorearb.h>
193   #if defined(GLFW_INCLUDE_GLEXT)
194    #include <GL/glext.h>
195   #endif
196
197  #endif /*__APPLE__*/
198
199 #elif defined(GLFW_INCLUDE_GLU)
200
201  #if defined(__APPLE__)
202
203   #if defined(GLFW_INCLUDE_GLU)
204    #include <OpenGL/glu.h>
205   #endif
206
207  #else /*__APPLE__*/
208
209   #if defined(GLFW_INCLUDE_GLU)
210    #include <GL/glu.h>
211   #endif
212
213  #endif /*__APPLE__*/
214
215 #elif !defined(GLFW_INCLUDE_NONE) && \
216       !defined(__gl_h_) && \
217       !defined(__gles1_gl_h_) && \
```

```
218       !defined(__gles2_gl2_h_) && \
219       !defined(__gles2_gl3_h_) && \
220       !defined(__gles2_gl31_h_) && \
221       !defined(__gles2_gl32_h_) && \
222       !defined(__gl_glcorearb_h_) && \
223       !defined(__gl2_h_) /*legacy*/ && \
224       !defined(__gl3_h_) /*legacy*/ && \
225       !defined(__gl31_h_) /*legacy*/ && \
226       !defined(__gl32_h_) /*legacy*/ && \
227       !defined(__glcorearb_h_) /*legacy*/ && \
228       !defined(__GL_H__) /*non-standard*/ && \
229       !defined(__gltypes_h_) /*non-standard*/ && \
230       !defined(__glee_h_) /*non-standard*/
231
232  #if defined(__APPLE__)
233
234   #if !defined(GLFW_INCLUDE_GLEXT)
235    #define GL_GLEXT_LEGACY
236   #endif
237   #include <OpenGL/gl.h>
238
239  #else /*__APPLE__*/
240
241   #include <GL/gl.h>
242   #if defined(GLFW_INCLUDE_GLEXT)
243    #include <GL/glext.h>
244   #endif
245
246  #endif /*__APPLE__*/
247
248 #endif /* OpenGL and OpenGL ES headers */
249
250 #if defined(GLFW_DLL) && defined(_GLFW_BUILD_DLL)
251  /* GLFW_DLL must be defined by applications that are linking against the DLL
252   * version of the GLFW library.  _GLFW_BUILD_DLL is defined by the GLFW
253   * configuration header when compiling the DLL version of the library.
254   */
255  #error "You must not have both GLFW_DLL and _GLFW_BUILD_DLL defined"
256 #endif
257
258 /* GLFWAPI is used to declare public API functions for export
259  * from the DLL / shared library / dynamic library.
260  */
261 #if defined(_WIN32) && defined(_GLFW_BUILD_DLL)
262  /* We are building GLFW as a Win32 DLL */
263  #define GLFWAPI __declspec(dllexport)
264 #elif defined(_WIN32) && defined(GLFW_DLL)
265  /* We are calling GLFW as a Win32 DLL */
266  #define GLFWAPI __declspec(dllimport)
267 #elif defined(__GNUC__) && defined(_GLFW_BUILD_DLL)
268  /* We are building GLFW as a shared / dynamic library */
269  #define GLFWAPI __attribute__((visibility("default")))
270 #else
271  /* We are building or calling GLFW as a static library */
272  #define GLFWAPI
273 #endif
274
275
276 /*************************************************************************
277  * GLFW API tokens
278  *************************************************************************/
279
288 #define GLFW_VERSION_MAJOR          3
295 #define GLFW_VERSION_MINOR          4
302 #define GLFW_VERSION_REVISION       0
313 #define GLFW_TRUE                   1
322 #define GLFW_FALSE                  0
323
332 #define GLFW_RELEASE                0
339 #define GLFW_PRESS                  1
346 #define GLFW_REPEAT                 2
356 #define GLFW_HAT_CENTERED           0
357 #define GLFW_HAT_UP                 1
358 #define GLFW_HAT_RIGHT              2
359 #define GLFW_HAT_DOWN               4
360 #define GLFW_HAT_LEFT               8
361 #define GLFW_HAT_RIGHT_UP           (GLFW_HAT_RIGHT | GLFW_HAT_UP)
362 #define GLFW_HAT_RIGHT_DOWN         (GLFW_HAT_RIGHT | GLFW_HAT_DOWN)
363 #define GLFW_HAT_LEFT_UP            (GLFW_HAT_LEFT  | GLFW_HAT_UP)
364 #define GLFW_HAT_LEFT_DOWN          (GLFW_HAT_LEFT  | GLFW_HAT_DOWN)
391 /* The unknown key */
392 #define GLFW_KEY_UNKNOWN            -1
393
394 /* Printable keys */
395 #define GLFW_KEY_SPACE              32
396 #define GLFW_KEY_APOSTROPHE         39  /* ' */
397 #define GLFW_KEY_COMMA              44  /* , */
```

```
398 #define GLFW_KEY_MINUS              45  /* - */
399 #define GLFW_KEY_PERIOD             46  /* . */
400 #define GLFW_KEY_SLASH              47  /* / */
401 #define GLFW_KEY_0                  48
402 #define GLFW_KEY_1                  49
403 #define GLFW_KEY_2                  50
404 #define GLFW_KEY_3                  51
405 #define GLFW_KEY_4                  52
406 #define GLFW_KEY_5                  53
407 #define GLFW_KEY_6                  54
408 #define GLFW_KEY_7                  55
409 #define GLFW_KEY_8                  56
410 #define GLFW_KEY_9                  57
411 #define GLFW_KEY_SEMICOLON          59  /* ; */
412 #define GLFW_KEY_EQUAL              61  /* = */
413 #define GLFW_KEY_A                  65
414 #define GLFW_KEY_B                  66
415 #define GLFW_KEY_C                  67
416 #define GLFW_KEY_D                  68
417 #define GLFW_KEY_E                  69
418 #define GLFW_KEY_F                  70
419 #define GLFW_KEY_G                  71
420 #define GLFW_KEY_H                  72
421 #define GLFW_KEY_I                  73
422 #define GLFW_KEY_J                  74
423 #define GLFW_KEY_K                  75
424 #define GLFW_KEY_L                  76
425 #define GLFW_KEY_M                  77
426 #define GLFW_KEY_N                  78
427 #define GLFW_KEY_O                  79
428 #define GLFW_KEY_P                  80
429 #define GLFW_KEY_Q                  81
430 #define GLFW_KEY_R                  82
431 #define GLFW_KEY_S                  83
432 #define GLFW_KEY_T                  84
433 #define GLFW_KEY_U                  85
434 #define GLFW_KEY_V                  86
435 #define GLFW_KEY_W                  87
436 #define GLFW_KEY_X                  88
437 #define GLFW_KEY_Y                  89
438 #define GLFW_KEY_Z                  90
439 #define GLFW_KEY_LEFT_BRACKET       91  /* [ */
440 #define GLFW_KEY_BACKSLASH          92  /* \ */
441 #define GLFW_KEY_RIGHT_BRACKET      93  /* ] */
442 #define GLFW_KEY_GRAVE_ACCENT       96  /* ` */
443 #define GLFW_KEY_WORLD_1            161 /* non-US #1 */
444 #define GLFW_KEY_WORLD_2            162 /* non-US #2 */
445
446 /* Function keys */
447 #define GLFW_KEY_ESCAPE             256
448 #define GLFW_KEY_ENTER              257
449 #define GLFW_KEY_TAB                258
450 #define GLFW_KEY_BACKSPACE          259
451 #define GLFW_KEY_INSERT             260
452 #define GLFW_KEY_DELETE             261
453 #define GLFW_KEY_RIGHT              262
454 #define GLFW_KEY_LEFT               263
455 #define GLFW_KEY_DOWN               264
456 #define GLFW_KEY_UP                 265
457 #define GLFW_KEY_PAGE_UP            266
458 #define GLFW_KEY_PAGE_DOWN          267
459 #define GLFW_KEY_HOME               268
460 #define GLFW_KEY_END                269
461 #define GLFW_KEY_CAPS_LOCK          280
462 #define GLFW_KEY_SCROLL_LOCK        281
463 #define GLFW_KEY_NUM_LOCK           282
464 #define GLFW_KEY_PRINT_SCREEN       283
465 #define GLFW_KEY_PAUSE              284
466 #define GLFW_KEY_F1                 290
467 #define GLFW_KEY_F2                 291
468 #define GLFW_KEY_F3                 292
469 #define GLFW_KEY_F4                 293
470 #define GLFW_KEY_F5                 294
471 #define GLFW_KEY_F6                 295
472 #define GLFW_KEY_F7                 296
473 #define GLFW_KEY_F8                 297
474 #define GLFW_KEY_F9                 298
475 #define GLFW_KEY_F10                299
476 #define GLFW_KEY_F11                300
477 #define GLFW_KEY_F12                301
478 #define GLFW_KEY_F13                302
479 #define GLFW_KEY_F14                303
480 #define GLFW_KEY_F15                304
481 #define GLFW_KEY_F16                305
482 #define GLFW_KEY_F17                306
483 #define GLFW_KEY_F18                307
484 #define GLFW_KEY_F19                308
```

```
485 #define GLFW_KEY_F20               309
486 #define GLFW_KEY_F21               310
487 #define GLFW_KEY_F22               311
488 #define GLFW_KEY_F23               312
489 #define GLFW_KEY_F24               313
490 #define GLFW_KEY_F25               314
491 #define GLFW_KEY_KP_0              320
492 #define GLFW_KEY_KP_1              321
493 #define GLFW_KEY_KP_2              322
494 #define GLFW_KEY_KP_3              323
495 #define GLFW_KEY_KP_4              324
496 #define GLFW_KEY_KP_5              325
497 #define GLFW_KEY_KP_6              326
498 #define GLFW_KEY_KP_7              327
499 #define GLFW_KEY_KP_8              328
500 #define GLFW_KEY_KP_9              329
501 #define GLFW_KEY_KP_DECIMAL        330
502 #define GLFW_KEY_KP_DIVIDE         331
503 #define GLFW_KEY_KP_MULTIPLY       332
504 #define GLFW_KEY_KP_SUBTRACT       333
505 #define GLFW_KEY_KP_ADD            334
506 #define GLFW_KEY_KP_ENTER          335
507 #define GLFW_KEY_KP_EQUAL          336
508 #define GLFW_KEY_LEFT_SHIFT        340
509 #define GLFW_KEY_LEFT_CONTROL      341
510 #define GLFW_KEY_LEFT_ALT          342
511 #define GLFW_KEY_LEFT_SUPER        343
512 #define GLFW_KEY_RIGHT_SHIFT       344
513 #define GLFW_KEY_RIGHT_CONTROL     345
514 #define GLFW_KEY_RIGHT_ALT         346
515 #define GLFW_KEY_RIGHT_SUPER       347
516 #define GLFW_KEY_MENU              348
517
518 #define GLFW_KEY_LAST              GLFW_KEY_MENU
519
534 #define GLFW_MOD_SHIFT           0x0001
539 #define GLFW_MOD_CONTROL         0x0002
544 #define GLFW_MOD_ALT             0x0004
549 #define GLFW_MOD_SUPER           0x0008
555 #define GLFW_MOD_CAPS_LOCK       0x0010
561 #define GLFW_MOD_NUM_LOCK        0x0020
562
572 #define GLFW_MOUSE_BUTTON_1         0
573 #define GLFW_MOUSE_BUTTON_2         1
574 #define GLFW_MOUSE_BUTTON_3         2
575 #define GLFW_MOUSE_BUTTON_4         3
576 #define GLFW_MOUSE_BUTTON_5         4
577 #define GLFW_MOUSE_BUTTON_6         5
578 #define GLFW_MOUSE_BUTTON_7         6
579 #define GLFW_MOUSE_BUTTON_8         7
580 #define GLFW_MOUSE_BUTTON_LAST      GLFW_MOUSE_BUTTON_8
581 #define GLFW_MOUSE_BUTTON_LEFT      GLFW_MOUSE_BUTTON_1
582 #define GLFW_MOUSE_BUTTON_RIGHT     GLFW_MOUSE_BUTTON_2
583 #define GLFW_MOUSE_BUTTON_MIDDLE    GLFW_MOUSE_BUTTON_3
593 #define GLFW_JOYSTICK_1             0
594 #define GLFW_JOYSTICK_2             1
595 #define GLFW_JOYSTICK_3             2
596 #define GLFW_JOYSTICK_4             3
597 #define GLFW_JOYSTICK_5             4
598 #define GLFW_JOYSTICK_6             5
599 #define GLFW_JOYSTICK_7             6
600 #define GLFW_JOYSTICK_8             7
601 #define GLFW_JOYSTICK_9             8
602 #define GLFW_JOYSTICK_10            9
603 #define GLFW_JOYSTICK_11            10
604 #define GLFW_JOYSTICK_12            11
605 #define GLFW_JOYSTICK_13            12
606 #define GLFW_JOYSTICK_14            13
607 #define GLFW_JOYSTICK_15            14
608 #define GLFW_JOYSTICK_16            15
609 #define GLFW_JOYSTICK_LAST          GLFW_JOYSTICK_16
619 #define GLFW_GAMEPAD_BUTTON_A               0
620 #define GLFW_GAMEPAD_BUTTON_B               1
621 #define GLFW_GAMEPAD_BUTTON_X               2
622 #define GLFW_GAMEPAD_BUTTON_Y               3
623 #define GLFW_GAMEPAD_BUTTON_LEFT_BUMPER     4
624 #define GLFW_GAMEPAD_BUTTON_RIGHT_BUMPER    5
625 #define GLFW_GAMEPAD_BUTTON_BACK            6
626 #define GLFW_GAMEPAD_BUTTON_START           7
627 #define GLFW_GAMEPAD_BUTTON_GUIDE           8
628 #define GLFW_GAMEPAD_BUTTON_LEFT_THUMB      9
629 #define GLFW_GAMEPAD_BUTTON_RIGHT_THUMB     10
630 #define GLFW_GAMEPAD_BUTTON_DPAD_UP         11
631 #define GLFW_GAMEPAD_BUTTON_DPAD_RIGHT      12
632 #define GLFW_GAMEPAD_BUTTON_DPAD_DOWN       13
633 #define GLFW_GAMEPAD_BUTTON_DPAD_LEFT       14
634 #define GLFW_GAMEPAD_BUTTON_LAST            GLFW_GAMEPAD_BUTTON_DPAD_LEFT
```

```
635
636 #define GLFW_GAMEPAD_BUTTON_CROSS       GLFW_GAMEPAD_BUTTON_A
637 #define GLFW_GAMEPAD_BUTTON_CIRCLE      GLFW_GAMEPAD_BUTTON_B
638 #define GLFW_GAMEPAD_BUTTON_SQUARE      GLFW_GAMEPAD_BUTTON_X
639 #define GLFW_GAMEPAD_BUTTON_TRIANGLE    GLFW_GAMEPAD_BUTTON_Y
649 #define GLFW_GAMEPAD_AXIS_LEFT_X        0
650 #define GLFW_GAMEPAD_AXIS_LEFT_Y        1
651 #define GLFW_GAMEPAD_AXIS_RIGHT_X       2
652 #define GLFW_GAMEPAD_AXIS_RIGHT_Y       3
653 #define GLFW_GAMEPAD_AXIS_LEFT_TRIGGER  4
654 #define GLFW_GAMEPAD_AXIS_RIGHT_TRIGGER 5
655 #define GLFW_GAMEPAD_AXIS_LAST          GLFW_GAMEPAD_AXIS_RIGHT_TRIGGER
671 #define GLFW_NO_ERROR               0
680 #define GLFW_NOT_INITIALIZED        0x00010001
690 #define GLFW_NO_CURRENT_CONTEXT     0x00010002
698 #define GLFW_INVALID_ENUM           0x00010003
709 #define GLFW_INVALID_VALUE          0x00010004
717 #define GLFW_OUT_OF_MEMORY          0x00010005
733 #define GLFW_API_UNAVAILABLE        0x00010006
750 #define GLFW_VERSION_UNAVAILABLE    0x00010007
761 #define GLFW_PLATFORM_ERROR         0x00010008
780 #define GLFW_FORMAT_UNAVAILABLE     0x00010009
788 #define GLFW_NO_WINDOW_CONTEXT      0x0001000A
799 #define GLFW_CURSOR_UNAVAILABLE     0x0001000B
813 #define GLFW_FEATURE_UNAVAILABLE    0x0001000C
826 #define GLFW_FEATURE_UNIMPLEMENTED  0x0001000D
848 #define GLFW_PLATFORM_UNAVAILABLE   0x0001000E
858 #define GLFW_FOCUSED                0x00020001
863 #define GLFW_ICONIFIED              0x00020002
869 #define GLFW_RESIZABLE              0x00020003
875 #define GLFW_VISIBLE                0x00020004
881 #define GLFW_DECORATED              0x00020005
887 #define GLFW_AUTO_ICONIFY           0x00020006
893 #define GLFW_FLOATING               0x00020007
899 #define GLFW_MAXIMIZED              0x00020008
904 #define GLFW_CENTER_CURSOR          0x00020009
911 #define GLFW_TRANSPARENT_FRAMEBUFFER 0x0002000A
916 #define GLFW_HOVERED                0x0002000B
922 #define GLFW_FOCUS_ON_SHOW          0x0002000C
923
929 #define GLFW_MOUSE_PASSTHROUGH      0x0002000D
930
935 #define GLFW_RED_BITS               0x00021001
940 #define GLFW_GREEN_BITS             0x00021002
945 #define GLFW_BLUE_BITS              0x00021003
950 #define GLFW_ALPHA_BITS             0x00021004
955 #define GLFW_DEPTH_BITS             0x00021005
960 #define GLFW_STENCIL_BITS           0x00021006
965 #define GLFW_ACCUM_RED_BITS         0x00021007
970 #define GLFW_ACCUM_GREEN_BITS       0x00021008
975 #define GLFW_ACCUM_BLUE_BITS        0x00021009
980 #define GLFW_ACCUM_ALPHA_BITS       0x0002100A
985 #define GLFW_AUX_BUFFERS            0x0002100B
990 #define GLFW_STEREO                 0x0002100C
995 #define GLFW_SAMPLES                0x0002100D
1000 #define GLFW_SRGB_CAPABLE           0x0002100E
1005 #define GLFW_REFRESH_RATE           0x0002100F
1011 #define GLFW_DOUBLEBUFFER           0x00021010
1012
1018 #define GLFW_CLIENT_API             0x00022001
1024 #define GLFW_CONTEXT_VERSION_MAJOR  0x00022002
1030 #define GLFW_CONTEXT_VERSION_MINOR  0x00022003
1036 #define GLFW_CONTEXT_REVISION       0x00022004
1042 #define GLFW_CONTEXT_ROBUSTNESS     0x00022005
1048 #define GLFW_OPENGL_FORWARD_COMPAT  0x00022006
1054 #define GLFW_CONTEXT_DEBUG          0x00022007
1059 #define GLFW_OPENGL_DEBUG_CONTEXT   GLFW_CONTEXT_DEBUG
1065 #define GLFW_OPENGL_PROFILE         0x00022008
1071 #define GLFW_CONTEXT_RELEASE_BEHAVIOR 0x00022009
1077 #define GLFW_CONTEXT_NO_ERROR       0x0002200A
1083 #define GLFW_CONTEXT_CREATION_API   0x0002200B
1087 #define GLFW_SCALE_TO_MONITOR       0x0002200C
1091 #define GLFW_COCOA_RETINA_FRAMEBUFFER 0x00023001
1095 #define GLFW_COCOA_FRAME_NAME         0x00023002
1099 #define GLFW_COCOA_GRAPHICS_SWITCHING 0x00023003
1103 #define GLFW_X11_CLASS_NAME         0x00024001
1107 #define GLFW_X11_INSTANCE_NAME      0x00024002
1108 #define GLFW_WIN32_KEYBOARD_MENU    0x00025001
1111 #define GLFW_NO_API                         0
1112 #define GLFW_OPENGL_API             0x00030001
1113 #define GLFW_OPENGL_ES_API          0x00030002
1114
1115 #define GLFW_NO_ROBUSTNESS                  0
1116 #define GLFW_NO_RESET_NOTIFICATION  0x00031001
1117 #define GLFW_LOSE_CONTEXT_ON_RESET  0x00031002
1118
1119 #define GLFW_OPENGL_ANY_PROFILE             0
```

```
1120 #define GLFW_OPENGL_CORE_PROFILE    0x00032001
1121 #define GLFW_OPENGL_COMPAT_PROFILE  0x00032002
1122
1123 #define GLFW_CURSOR                 0x00033001
1124 #define GLFW_STICKY_KEYS            0x00033002
1125 #define GLFW_STICKY_MOUSE_BUTTONS   0x00033003
1126 #define GLFW_LOCK_KEY_MODS          0x00033004
1127 #define GLFW_RAW_MOUSE_MOTION       0x00033005
1128
1129 #define GLFW_CURSOR_NORMAL          0x00034001
1130 #define GLFW_CURSOR_HIDDEN          0x00034002
1131 #define GLFW_CURSOR_DISABLED        0x00034003
1132
1133 #define GLFW_ANY_RELEASE_BEHAVIOR           0
1134 #define GLFW_RELEASE_BEHAVIOR_FLUSH 0x00035001
1135 #define GLFW_RELEASE_BEHAVIOR_NONE  0x00035002
1136
1137 #define GLFW_NATIVE_CONTEXT_API     0x00036001
1138 #define GLFW_EGL_CONTEXT_API        0x00036002
1139 #define GLFW_OSMESA_CONTEXT_API     0x00036003
1140
1141 #define GLFW_ANGLE_PLATFORM_TYPE_NONE    0x00037001
1142 #define GLFW_ANGLE_PLATFORM_TYPE_OPENGL  0x00037002
1143 #define GLFW_ANGLE_PLATFORM_TYPE_OPENGLES 0x00037003
1144 #define GLFW_ANGLE_PLATFORM_TYPE_D3D9    0x00037004
1145 #define GLFW_ANGLE_PLATFORM_TYPE_D3D11   0x00037005
1146 #define GLFW_ANGLE_PLATFORM_TYPE_VULKAN  0x00037007
1147 #define GLFW_ANGLE_PLATFORM_TYPE_METAL   0x00037008
1148
1162 #define GLFW_ARROW_CURSOR           0x00036001
1167 #define GLFW_IBEAM_CURSOR           0x00036002
1172 #define GLFW_CROSSHAIR_CURSOR       0x00036003
1177 #define GLFW_POINTING_HAND_CURSOR   0x00036004
1183 #define GLFW_RESIZE_EW_CURSOR       0x00036005
1189 #define GLFW_RESIZE_NS_CURSOR       0x00036006
1204 #define GLFW_RESIZE_NWSE_CURSOR     0x00036007
1219 #define GLFW_RESIZE_NESW_CURSOR     0x00036008
1225 #define GLFW_RESIZE_ALL_CURSOR      0x00036009
1237 #define GLFW_NOT_ALLOWED_CURSOR     0x0003600A
1242 #define GLFW_HRESIZE_CURSOR         GLFW_RESIZE_EW_CURSOR
1247 #define GLFW_VRESIZE_CURSOR         GLFW_RESIZE_NS_CURSOR
1252 #define GLFW_HAND_CURSOR            GLFW_POINTING_HAND_CURSOR
1255 #define GLFW_CONNECTED              0x00040001
1256 #define GLFW_DISCONNECTED           0x00040002
1257
1264 #define GLFW_JOYSTICK_HAT_BUTTONS   0x00050001
1269 #define GLFW_ANGLE_PLATFORM_TYPE    0x00050002
1274 #define GLFW_PLATFORM               0x00050003
1279 #define GLFW_COCOA_CHDIR_RESOURCES  0x00051001
1284 #define GLFW_COCOA_MENUBAR          0x00051002
1289 #define GLFW_X11_XCB_VULKAN_SURFACE 0x00052001
1298 #define GLFW_ANY_PLATFORM           0x00060000
1299 #define GLFW_PLATFORM_WIN32         0x00060001
1300 #define GLFW_PLATFORM_COCOA         0x00060002
1301 #define GLFW_PLATFORM_WAYLAND       0x00060003
1302 #define GLFW_PLATFORM_X11           0x00060004
1303 #define GLFW_PLATFORM_NULL          0x00060005
1306 #define GLFW_DONT_CARE              -1
1307
1308
1309 /*************************************************************************
1310  * GLFW API types
1311  *************************************************************************/
1312
1325 typedef void (*GLFWglproc)(void);
1326
1339 typedef void (*GLFWvkproc)(void);
1340
1351 typedef struct GLFWmonitor GLFWmonitor;
1352
1363 typedef struct GLFWwindow GLFWwindow;
1364
1375 typedef struct GLFWcursor GLFWcursor;
1376
1418 typedef void* (* GLFWallocatefun)(size_t size, void* user);
1419
1463 typedef void* (* GLFWreallocatefun)(void* block, size_t size, void* user);
1464
1500 typedef void (* GLFWdeallocatefun)(void* block, void* user);
1501
1524 typedef void (* GLFWerrorfun)(int error_code, const char* description);
1525
1547 typedef void (* GLFWwindowposfun)(GLFWwindow* window, int xpos, int ypos);
1548
1569 typedef void (* GLFWwindowsizefun)(GLFWwindow* window, int width, int height);
1570
1589 typedef void (* GLFWwindowclosefun)(GLFWwindow* window);
```

```
1590
1609 typedef void (* GLFWwindowrefreshfun)(GLFWwindow* window);
1610
1630 typedef void (* GLFWwindowfocusfun)(GLFWwindow* window, int focused);
1631
1651 typedef void (* GLFWwindowiconifyfun)(GLFWwindow* window, int iconified);
1652
1672 typedef void (* GLFWwindowmaximizefun)(GLFWwindow* window, int maximized);
1673
1693 typedef void (* GLFWframebuffersizefun)(GLFWwindow* window, int width, int height);
1694
1714 typedef void (* GLFWwindowcontentscalefun)(GLFWwindow* window, float xscale, float yscale);
1715
1740 typedef void (* GLFWmousebuttonfun)(GLFWwindow* window, int button, int action, int mods);
1741
1763 typedef void (* GLFWcursorposfun)(GLFWwindow* window, double xpos, double ypos);
1764
1784 typedef void (* GLFWcursorenterfun)(GLFWwindow* window, int entered);
1785
1805 typedef void (* GLFWscrollfun)(GLFWwindow* window, double xoffset, double yoffset);
1806
1831 typedef void (* GLFWkeyfun)(GLFWwindow* window, int key, int scancode, int action, int mods);
1832
1852 typedef void (* GLFWcharfun)(GLFWwindow* window, unsigned int codepoint);
1853
1879 typedef void (* GLFWcharmodsfun)(GLFWwindow* window, unsigned int codepoint, int mods);
1880
1903 typedef void (* GLFWdropfun)(GLFWwindow* window, int path_count, const char* paths[]);
1904
1924 typedef void (* GLFWmonitorfun)(GLFWmonitor* monitor, int event);
1925
1945 typedef void (* GLFWjoystickfun)(int jid, int event);
1946
1960 typedef struct GLFWvidmode
1961 {
1964     int width;
1967     int height;
1970     int redBits;
1973     int greenBits;
1976     int blueBits;
1979     int refreshRate;
1980 } GLFWvidmode;
1981
1994 typedef struct GLFWgammaramp
1995 {
1998     unsigned short* red;
2001     unsigned short* green;
2004     unsigned short* blue;
2007     unsigned int size;
2008 } GLFWgammaramp;
2009
2023 typedef struct GLFWimage
2024 {
2027     int width;
2030     int height;
2033     unsigned char* pixels;
2034 } GLFWimage;
2035
2047 typedef struct GLFWgamepadstate
2048 {
2052     unsigned char buttons[15];
2056     float axes[6];
2057 } GLFWgamepadstate;
2058
2068 typedef struct GLFWallocator
2069 {
2070     GLFWallocatefun allocate;
2071     GLFWreallocatefun reallocate;
2072     GLFWdeallocatefun deallocate;
2073     void* user;
2074 } GLFWallocator;
2075
2076
2077 /*************************************************************************
2078  * GLFW API functions
2079  *************************************************************************/
2080
2132 GLFWAPI int glfwInit(void);
2133
2166 GLFWAPI void glfwTerminate(void);
2167
2198 GLFWAPI void glfwInitHint(int hint, int value);
2199
2225 GLFWAPI void glfwInitAllocator(const GLFWallocator* allocator);
2226
2227 #if defined(VK_VERSION_1_0)
2228
```

```
2271 GLFWAPI void glfwInitVulkanLoader(PFN_vkGetInstanceProcAddr loader);
2272
2273 #endif /*VK_VERSION_1_0*/
2274
2300 GLFWAPI void glfwGetVersion(int* major, int* minor, int* rev);
2301
2334 GLFWAPI const char* glfwGetVersionString(void);
2335
2365 GLFWAPI int glfwGetError(const char** description);
2366
2411 GLFWAPI GLFWerrorfun glfwSetErrorCallback(GLFWerrorfun callback);
2412
2432 GLFWAPI int glfwGetPlatform(void);
2433
2456 GLFWAPI int glfwPlatformSupported(int platform);
2457
2485 GLFWAPI GLFWmonitor** glfwGetMonitors(int* count);
2486
2509 GLFWAPI GLFWmonitor* glfwGetPrimaryMonitor(void);
2510
2534 GLFWAPI void glfwGetMonitorPos(GLFWmonitor* monitor, int* xpos, int* ypos);
2535
2565 GLFWAPI void glfwGetMonitorWorkarea(GLFWmonitor* monitor, int* xpos, int* ypos, int* width, int*
       height);
2566
2599 GLFWAPI void glfwGetMonitorPhysicalSize(GLFWmonitor* monitor, int* widthMM, int* heightMM);
2600
2631 GLFWAPI void glfwGetMonitorContentScale(GLFWmonitor* monitor, float* xscale, float* yscale);
2632
2657 GLFWAPI const char* glfwGetMonitorName(GLFWmonitor* monitor);
2658
2683 GLFWAPI void glfwSetMonitorUserPointer(GLFWmonitor* monitor, void* pointer);
2684
2707 GLFWAPI void* glfwGetMonitorUserPointer(GLFWmonitor* monitor);
2708
2737 GLFWAPI GLFWmonitorfun glfwSetMonitorCallback(GLFWmonitorfun callback);
2738
2771 GLFWAPI const GLFWvidmode* glfwGetVideoModes(GLFWmonitor* monitor, int* count);
2772
2799 GLFWAPI const GLFWvidmode* glfwGetVideoMode(GLFWmonitor* monitor);
2800
2832 GLFWAPI void glfwSetGamma(GLFWmonitor* monitor, float gamma);
2833
2862 GLFWAPI const GLFWgammaramp* glfwGetGammaRamp(GLFWmonitor* monitor);
2863
2903 GLFWAPI void glfwSetGammaRamp(GLFWmonitor* monitor, const GLFWgammaramp* ramp);
2904
2922 GLFWAPI void glfwDefaultWindowHints(void);
2923
2957 GLFWAPI void glfwWindowHint(int hint, int value);
2958
2995 GLFWAPI void glfwWindowHintString(int hint, const char* value);
2996
3141 GLFWAPI GLFWwindow* glfwCreateWindow(int width, int height, const char* title, GLFWmonitor* monitor,
       GLFWwindow* share);
3142
3170 GLFWAPI void glfwDestroyWindow(GLFWwindow* window);
3171
3190 GLFWAPI int glfwWindowShouldClose(GLFWwindow* window);
3191
3212 GLFWAPI void glfwSetWindowShouldClose(GLFWwindow* window, int value);
3213
3237 GLFWAPI void glfwSetWindowTitle(GLFWwindow* window, const char* title);
3238
3285 GLFWAPI void glfwSetWindowIcon(GLFWwindow* window, int count, const GLFWimage* images);
3286
3317 GLFWAPI void glfwGetWindowPos(GLFWwindow* window, int* xpos, int* ypos);
3318
3352 GLFWAPI void glfwSetWindowPos(GLFWwindow* window, int xpos, int ypos);
3353
3382 GLFWAPI void glfwGetWindowSize(GLFWwindow* window, int* width, int* height);
3383
3425 GLFWAPI void glfwSetWindowSizeLimits(GLFWwindow* window, int minwidth, int minheight, int maxwidth, int
       maxheight);
3426
3468 GLFWAPI void glfwSetWindowAspectRatio(GLFWwindow* window, int numer, int denom);
3469
3509 GLFWAPI void glfwSetWindowSize(GLFWwindow* window, int width, int height);
3510
3538 GLFWAPI void glfwGetFramebufferSize(GLFWwindow* window, int* width, int* height);
3539
3575 GLFWAPI void glfwGetWindowFrameSize(GLFWwindow* window, int* left, int* top, int* right, int* bottom);
3576
3608 GLFWAPI void glfwGetWindowContentScale(GLFWwindow* window, float* xscale, float* yscale);
3609
3635 GLFWAPI float glfwGetWindowOpacity(GLFWwindow* window);
3636
```

```
3667 GLFWAPI void glfwSetWindowOpacity(GLFWwindow* window, float opacity);
3668
3698 GLFWAPI void glfwIconifyWindow(GLFWwindow* window);
3699
3725 GLFWAPI void glfwRestoreWindow(GLFWwindow* window);
3726
3750 GLFWAPI void glfwMaximizeWindow(GLFWwindow* window);
3751
3782 GLFWAPI void glfwShowWindow(GLFWwindow* window);
3783
3804 GLFWAPI void glfwHideWindow(GLFWwindow* window);
3805
3843 GLFWAPI void glfwFocusWindow(GLFWwindow* window);
3844
3870 GLFWAPI void glfwRequestWindowAttention(GLFWwindow* window);
3871
3892 GLFWAPI GLFWmonitor* glfwGetWindowMonitor(GLFWwindow* window);
3893
3951 GLFWAPI void glfwSetWindowMonitor(GLFWwindow* window, GLFWmonitor* monitor, int xpos, int ypos, int
       width, int height, int refreshRate);
3952
3985 GLFWAPI int glfwGetWindowAttrib(GLFWwindow* window, int attrib);
3986
4023 GLFWAPI void glfwSetWindowAttrib(GLFWwindow* window, int attrib, int value);
4024
4046 GLFWAPI void glfwSetWindowUserPointer(GLFWwindow* window, void* pointer);
4047
4067 GLFWAPI void* glfwGetWindowUserPointer(GLFWwindow* window);
4068
4102 GLFWAPI GLFWwindowposfun glfwSetWindowPosCallback(GLFWwindow* window, GLFWwindowposfun callback);
4103
4134 GLFWAPI GLFWwindowsizefun glfwSetWindowSizeCallback(GLFWwindow* window, GLFWwindowsizefun callback);
4135
4174 GLFWAPI GLFWwindowclosefun glfwSetWindowCloseCallback(GLFWwindow* window, GLFWwindowclosefun callback);
4175
4210 GLFWAPI GLFWwindowrefreshfun glfwSetWindowRefreshCallback(GLFWwindow* window, GLFWwindowrefreshfun
       callback);
4211
4245 GLFWAPI GLFWwindowfocusfun glfwSetWindowFocusCallback(GLFWwindow* window, GLFWwindowfocusfun callback);
4246
4275 GLFWAPI GLFWwindowiconifyfun glfwSetWindowIconifyCallback(GLFWwindow* window, GLFWwindowiconifyfun
       callback);
4276
4305 GLFWAPI GLFWwindowmaximizefun glfwSetWindowMaximizeCallback(GLFWwindow* window, GLFWwindowmaximizefun
       callback);
4306
4335 GLFWAPI GLFWframebuffersizefun glfwSetFramebufferSizeCallback(GLFWwindow* window,
       GLFWframebuffersizefun callback);
4336
4366 GLFWAPI GLFWwindowcontentscalefun glfwSetWindowContentScaleCallback(GLFWwindow* window,
       GLFWwindowcontentscalefun callback);
4367
4404 GLFWAPI void glfwPollEvents(void);
4405
4449 GLFWAPI void glfwWaitEvents(void);
4450
4498 GLFWAPI void glfwWaitEventsTimeout(double timeout);
4499
4518 GLFWAPI void glfwPostEmptyEvent(void);
4519
4543 GLFWAPI int glfwGetInputMode(GLFWwindow* window, int mode);
4544
4606 GLFWAPI void glfwSetInputMode(GLFWwindow* window, int mode, int value);
4607
4635 GLFWAPI int glfwRawMouseMotionSupported(void);
4636
4703 GLFWAPI const char* glfwGetKeyName(int key, int scancode);
4704
4727 GLFWAPI int glfwGetKeyScancode(int key);
4728
4767 GLFWAPI int glfwGetKey(GLFWwindow* window, int key);
4768
4796 GLFWAPI int glfwGetMouseButton(GLFWwindow* window, int button);
4797
4834 GLFWAPI void glfwGetCursorPos(GLFWwindow* window, double* xpos, double* ypos);
4835
4874 GLFWAPI void glfwSetCursorPos(GLFWwindow* window, double xpos, double ypos);
4875
4912 GLFWAPI GLFWcursor* glfwCreateCursor(const GLFWimage* image, int xhot, int yhot);
4913
4960 GLFWAPI GLFWcursor* glfwCreateStandardCursor(int shape);
4961
4987 GLFWAPI void glfwDestroyCursor(GLFWcursor* cursor);
4988
5014 GLFWAPI void glfwSetCursor(GLFWwindow* window, GLFWcursor* cursor);
5015
5064 GLFWAPI GLFWkeyfun glfwSetKeyCallback(GLFWwindow* window, GLFWkeyfun callback);
```

```
5065
5107 GLFWAPI GLFWcharfun glfwSetCharCallback(GLFWwindow* window, GLFWcharfun callback);
5108
5149 GLFWAPI GLFWcharmodsfun glfwSetCharModsCallback(GLFWwindow* window, GLFWcharmodsfun callback);
5150
5186 GLFWAPI GLFWmousebuttonfun glfwSetMouseButtonCallback(GLFWwindow* window, GLFWmousebuttonfun callback);
5187
5218 GLFWAPI GLFWcursorposfun glfwSetCursorPosCallback(GLFWwindow* window, GLFWcursorposfun callback);
5219
5249 GLFWAPI GLFWcursorenterfun glfwSetCursorEnterCallback(GLFWwindow* window, GLFWcursorenterfun callback);
5250
5283 GLFWAPI GLFWscrollfun glfwSetScrollCallback(GLFWwindow* window, GLFWscrollfun callback);
5284
5320 GLFWAPI GLFWdropfun glfwSetDropCallback(GLFWwindow* window, GLFWdropfun callback);
5321
5344 GLFWAPI int glfwJoystickPresent(int jid);
5345
5377 GLFWAPI const float* glfwGetJoystickAxes(int jid, int* count);
5378
5418 GLFWAPI const unsigned char* glfwGetJoystickButtons(int jid, int* count);
5419
5475 GLFWAPI const unsigned char* glfwGetJoystickHats(int jid, int* count);
5476
5506 GLFWAPI const char* glfwGetJoystickName(int jid);
5507
5547 GLFWAPI const char* glfwGetJoystickGUID(int jid);
5548
5573 GLFWAPI void glfwSetJoystickUserPointer(int jid, void* pointer);
5574
5597 GLFWAPI void* glfwGetJoystickUserPointer(int jid);
5598
5625 GLFWAPI int glfwJoystickIsGamepad(int jid);
5626
5661 GLFWAPI GLFWjoystickfun glfwSetJoystickCallback(GLFWjoystickfun callback);
5662
5695 GLFWAPI int glfwUpdateGamepadMappings(const char* string);
5696
5727 GLFWAPI const char* glfwGetGamepadName(int jid);
5728
5765 GLFWAPI int glfwGetGamepadState(int jid, GLFWgamepadstate* state);
5766
5790 GLFWAPI void glfwSetClipboardString(GLFWwindow* window, const char* string);
5791
5820 GLFWAPI const char* glfwGetClipboardString(GLFWwindow* window);
5821
5850 GLFWAPI double glfwGetTime(void);
5851
5880 GLFWAPI void glfwSetTime(double time);
5881
5902 GLFWAPI uint64_t glfwGetTimerValue(void);
5903
5922 GLFWAPI uint64_t glfwGetTimerFrequency(void);
5923
5960 GLFWAPI void glfwMakeContextCurrent(GLFWwindow* window);
5961
5981 GLFWAPI GLFWwindow* glfwGetCurrentContext(void);
5982
6015 GLFWAPI void glfwSwapBuffers(GLFWwindow* window);
6016
6061 GLFWAPI void glfwSwapInterval(int interval);
6062
6099 GLFWAPI int glfwExtensionSupported(const char* extension);
6100
6141 GLFWAPI GLFWglproc glfwGetProcAddress(const char* procname);
6142
6167 GLFWAPI int glfwVulkanSupported(void);
6168
6211 GLFWAPI const char** glfwGetRequiredInstanceExtensions(uint32_t* count);
6212
6213 #if defined(VK_VERSION_1_0)
6214
6254 GLFWAPI GLFWvkproc glfwGetInstanceProcAddress(VkInstance instance, const char* procname);
6255
6291 GLFWAPI int glfwGetPhysicalDevicePresentationSupport(VkInstance instance, VkPhysicalDevice device,
       uint32_t queuefamily);
6292
6361 GLFWAPI VkResult glfwCreateWindowSurface(VkInstance instance, GLFWwindow* window, const
       VkAllocationCallbacks* allocator, VkSurfaceKHR* surface);
6362
6363 #endif /*VK_VERSION_1_0*/
6364
6365
6366 /*************************************************************************
6367  * Global definition cleanup
6368  ************************************************************************/
6369
6370 /* ------------------- BEGIN SYSTEM/COMPILER SPECIFIC -------------------- */
```

```
6371
6372 #ifdef GLFW_WINGDIAPI_DEFINED
6373  #undef WINGDIAPI
6374  #undef GLFW_WINGDIAPI_DEFINED
6375 #endif
6376
6377 #ifdef GLFW_CALLBACK_DEFINED
6378  #undef CALLBACK
6379  #undef GLFW_CALLBACK_DEFINED
6380 #endif
6381
6382 /* Some OpenGL related headers need GLAPIENTRY, but it is unconditionally
6383  * defined by some gl.h variants (OpenBSD) so define it after if needed.
6384  */
6385 #ifndef GLAPIENTRY
6386  #define GLAPIENTRY APIENTRY
6387 #endif
6388
6389 /* ------------------- END SYSTEM/COMPILER SPECIFIC -------------------- */
6390
6391
6392 #ifdef __cplusplus
6393 }
6394 #endif
6395
6396 #endif /* _glfw3_h_ */
6397
```

27.17 lib/glfw/include/GLFW/glfw3native.h File Reference

The header of the native access functions.

27.17.1 Detailed Description

The header of the native access functions.

This is the header file of the native access functions. See Native access for more information.

27.18 glfw3native.h

Go to the documentation of this file.

```
1 /*************************************************************************
2  * GLFW 3.4 - www.glfw.org
3  * A library for OpenGL, window and input
4  *------------------------------------------------------------------------
5  * Copyright (c) 2002-2006 Marcus Geelnard
6  * Copyright (c) 2006-2018 Camilla Löwy <elmindreda@glfw.org>
7  *
8  * This software is provided 'as-is', without any express or implied
9  * warranty. In no event will the authors be held liable for any damages
10  * arising from the use of this software.
11  *
12  * Permission is granted to anyone to use this software for any purpose,
13  * including commercial applications, and to alter it and redistribute it
14  * freely, subject to the following restrictions:
15  *
16  * 1. The origin of this software must not be misrepresented; you must not
17  *    claim that you wrote the original software. If you use this software
18  *    in a product, an acknowledgment in the product documentation would
19  *    be appreciated but is not required.
20  *
21  * 2. Altered source versions must be plainly marked as such, and must not
22  *    be misrepresented as being the original software.
23  *
24  * 3. This notice may not be removed or altered from any source
25  *    distribution.
26  *
27  *************************************************************************/
28
```

```
29 #ifndef _glfw3_native_h_
30 #define _glfw3_native_h_
31
32 #ifdef __cplusplus
33 extern "C" {
34 #endif
35
36
37 /*************************************************************************
38  * Doxygen documentation
39  *************************************************************************/
40
41 /*************************************************************************
42  * System headers and types
43  *************************************************************************/
44
45 #if defined(GLFW_EXPOSE_NATIVE_WIN32) || defined(GLFW_EXPOSE_NATIVE_WGL)
46  /* This is a workaround for the fact that glfw3.h needs to export APIENTRY (for
47   * example to allow applications to correctly declare a GL_KHR_debug callback)
48   * but windows.h assumes no one will define APIENTRY before it does
49   */
50  #if defined(GLFW_APIENTRY_DEFINED)
51   #undef APIENTRY
52   #undef GLFW_APIENTRY_DEFINED
53  #endif
54  #include <windows.h>
55 #elif defined(GLFW_EXPOSE_NATIVE_COCOA) || defined(GLFW_EXPOSE_NATIVE_NSGL)
56  #if defined(__OBJC__)
57   #import <Cocoa/Cocoa.h>
58  #else
59   #include <ApplicationServices/ApplicationServices.h>
60   typedef void* id;
61  #endif
62 #elif defined(GLFW_EXPOSE_NATIVE_X11) || defined(GLFW_EXPOSE_NATIVE_GLX)
63  #include <X11/Xlib.h>
64  #include <X11/extensions/Xrandr.h>
65 #elif defined(GLFW_EXPOSE_NATIVE_WAYLAND)
66  #include <wayland-client.h>
67 #endif
68
69 #if defined(GLFW_EXPOSE_NATIVE_WGL)
70  /* WGL is declared by windows.h */
71 #endif
72 #if defined(GLFW_EXPOSE_NATIVE_NSGL)
73  /* NSGL is declared by Cocoa.h */
74 #endif
75 #if defined(GLFW_EXPOSE_NATIVE_GLX)
76  /* This is a workaround for the fact that glfw3.h defines GLAPIENTRY because by
77   * default it also acts as an OpenGL header
78   * However, glx.h will include gl.h, which will define it unconditionally
79   */
80  #undef GLAPIENTRY
81  #include <GL/glx.h>
82 #endif
83 #if defined(GLFW_EXPOSE_NATIVE_EGL)
84  #include <EGL/egl.h>
85 #endif
86 #if defined(GLFW_EXPOSE_NATIVE_OSMESA)
87  /* This is a workaround for the fact that glfw3.h defines GLAPIENTRY because by
88   * default it also acts as an OpenGL header
89   * However, osmesa.h will include gl.h, which will define it unconditionally
90   */
91  #undef GLAPIENTRY
92  #include <GL/osmesa.h>
93 #endif
94
95
96 /*************************************************************************
97  * Functions
98  *************************************************************************/
99
100 #if defined(GLFW_EXPOSE_NATIVE_WIN32)
116 GLFWAPI const char* glfwGetWin32Adapter(GLFWmonitor* monitor);
117
133 GLFWAPI const char* glfwGetWin32Monitor(GLFWmonitor* monitor);
134
157 GLFWAPI HWND glfwGetWin32Window(GLFWwindow* window);
158 #endif
159
160 #if defined(GLFW_EXPOSE_NATIVE_WGL)
184 GLFWAPI HGLRC glfwGetWGLContext(GLFWwindow* window);
185 #endif
186
187 #if defined(GLFW_EXPOSE_NATIVE_COCOA)
202 GLFWAPI CGDirectDisplayID glfwGetCocoaMonitor(GLFWmonitor* monitor);
203
218 GLFWAPI id glfwGetCocoaWindow(GLFWwindow* window);
```

```
258 #endif
259
260 #if defined(GLFW_EXPOSE_NATIVE_NSGL)
276 GLFWAPI id glfwGetNSGLContext(GLFWwindow* window);
277 #endif
278
279 #if defined(GLFW_EXPOSE_NATIVE_X11)
294 GLFWAPI Display* glfwGetX11Display(void);
295
310 GLFWAPI RRCrtc glfwGetX11Adapter(GLFWmonitor* monitor);
311
326 GLFWAPI RROutput glfwGetX11Monitor(GLFWmonitor* monitor);
327
342 GLFWAPI Window glfwGetX11Window(GLFWwindow* window);
343
364 GLFWAPI void glfwSetX11SelectionString(const char* string);
365
392 GLFWAPI const char* glfwGetX11SelectionString(void);
393 #endif
394
395 #if defined(GLFW_EXPOSE_NATIVE_GLX)
411 GLFWAPI GLXContext glfwGetGLXContext(GLFWwindow* window);
412
428 GLFWAPI GLXWindow glfwGetGLXWindow(GLFWwindow* window);
429 #endif
430
431 #if defined(GLFW_EXPOSE_NATIVE_WAYLAND)
446 GLFWAPI struct wl_display* glfwGetWaylandDisplay(void);
447
462 GLFWAPI struct wl_output* glfwGetWaylandMonitor(GLFWmonitor* monitor);
463
478 GLFWAPI struct wl_surface* glfwGetWaylandWindow(GLFWwindow* window);
479 #endif
480
481 #if defined(GLFW_EXPOSE_NATIVE_EGL)
496 GLFWAPI EGLDisplay glfwGetEGLDisplay(void);
497
513 GLFWAPI EGLContext glfwGetEGLContext(GLFWwindow* window);
514
530 GLFWAPI EGLSurface glfwGetEGLSurface(GLFWwindow* window);
531 #endif
532
533 #if defined(GLFW_EXPOSE_NATIVE_OSMESA)
556 GLFWAPI int glfwGetOSMesaColorBuffer(GLFWwindow* window, int* width, int* height, int* format, void**
       buffer);
557
580 GLFWAPI int glfwGetOSMesaDepthBuffer(GLFWwindow* window, int* width, int* height, int* bytesPerValue,
       void** buffer);
581
597 GLFWAPI OSMesaContext glfwGetOSMesaContext(GLFWwindow* window);
598 #endif
599
600 #ifdef __cplusplus
601 }
602 #endif
603
604 #endif /* _glfw3_native_h_ */
605
```

27.19 cocoa_joystick.h

```
1 //========================================================================
2 // GLFW 3.4 Cocoa - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2006-2017 Camilla Löwy <elmindreda@glfw.org>
5 //
6 // This software is provided 'as-is', without any express or implied
7 // warranty. In no event will the authors be held liable for any damages
8 // arising from the use of this software.
9 //
10 // Permission is granted to anyone to use this software for any purpose,
11 // including commercial applications, and to alter it and redistribute it
12 // freely, subject to the following restrictions:
13 //
14 // 1. The origin of this software must not be misrepresented; you must not
15 //    claim that you wrote the original software. If you use this software
16 //    in a product, an acknowledgment in the product documentation would
17 //    be appreciated but is not required.
18 //
19 // 2. Altered source versions must be plainly marked as such, and must not
20 //    be misrepresented as being the original software.
21 //
22 // 3. This notice may not be removed or altered from any source
```

```
23 //    distribution.
24 //
25 //========================================================================
26
27 #include <IOKit/IOKitLib.h>
28 #include <IOKit/IOCFPlugIn.h>
29 #include <IOKit/hid/IOHIDKeys.h>
30
31 #define GLFW_COCOA_JOYSTICK_STATE         _GLFWjoystickNS ns;
32 #define GLFW_COCOA_LIBRARY_JOYSTICK_STATE
33
34 #define GLFW_BUILD_COCOA_MAPPINGS
35
36 // Cocoa-specific per-joystick data
37 //
38 typedef struct _GLFWjoystickNS
39 {
40     IOHIDDeviceRef      device;
41     CFMutableArrayRef   axes;
42     CFMutableArrayRef   buttons;
43     CFMutableArrayRef   hats;
44 } _GLFWjoystickNS;
45
46 GLFWbool _glfwInitJoysticksCocoa(void);
47 void _glfwTerminateJoysticksCocoa(void);
48 int _glfwPollJoystickCocoa(_GLFWjoystick* js, int mode);
49 const char* _glfwGetMappingNameCocoa(void);
50 void _glfwUpdateGamepadGUIDCocoa(char* guid);
51
```

27.20 cocoa_platform.h

```
1 //========================================================================
2 // GLFW 3.4 macOS - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2009-2019 Camilla Löwy <elmindreda@glfw.org>
5 //
6 // This software is provided 'as-is', without any express or implied
7 // warranty. In no event will the authors be held liable for any damages
8 // arising from the use of this software.
9 //
10 // Permission is granted to anyone to use this software for any purpose,
11 // including commercial applications, and to alter it and redistribute it
12 // freely, subject to the following restrictions:
13 //
14 // 1. The origin of this software must not be misrepresented; you must not
15 //    claim that you wrote the original software. If you use this software
16 //    in a product, an acknowledgment in the product documentation would
17 //    be appreciated but is not required.
18 //
19 // 2. Altered source versions must be plainly marked as such, and must not
20 //    be misrepresented as being the original software.
21 //
22 // 3. This notice may not be removed or altered from any source
23 //    distribution.
24 //
25 //========================================================================
26
27 #include <stdint.h>
28
29 #include <Carbon/Carbon.h>
30 #include <IOKit/hid/IOHIDLib.h>
31
32 // NOTE: All of NSGL was deprecated in the 10.14 SDK
33 //       This disables the pointless warnings for every symbol we use
34 #ifndef GL_SILENCE_DEPRECATION
35 #define GL_SILENCE_DEPRECATION
36 #endif
37
38 #if defined(__OBJC__)
39 #import <Cocoa/Cocoa.h>
40 #else
41 typedef void* id;
42 #endif
43
44 // NOTE: Many Cocoa enum values have been renamed and we need to build across
45 //       SDK versions where one is unavailable or deprecated.
46 //       We use the newer names in code and replace them with the older names if
47 //       the base SDK does not provide the newer names.
48
49 #if MAC_OS_X_VERSION_MAX_ALLOWED < 101400
50  #define NSOpenGLContextParameterSwapInterval NSOpenGLCPSwapInterval
51  #define NSOpenGLContextParameterSurfaceOpacity NSOpenGLCPSurfaceOpacity
```

```
52 #endif
53
54 #if MAC_OS_X_VERSION_MAX_ALLOWED < 101200
55  #define NSBitmapFormatAlphaNonpremultiplied NSAlphaNonpremultipliedBitmapFormat
56  #define NSEventMaskAny NSAnyEventMask
57  #define NSEventMaskKeyUp NSKeyUpMask
58  #define NSEventModifierFlagCapsLock NSAlphaShiftKeyMask
59  #define NSEventModifierFlagCommand NSCommandKeyMask
60  #define NSEventModifierFlagControl NSControlKeyMask
61  #define NSEventModifierFlagDeviceIndependentFlagsMask NSDeviceIndependentModifierFlagsMask
62  #define NSEventModifierFlagOption NSAlternateKeyMask
63  #define NSEventModifierFlagShift NSShiftKeyMask
64  #define NSEventTypeApplicationDefined NSApplicationDefined
65  #define NSWindowStyleMaskBorderless NSBorderlessWindowMask
66  #define NSWindowStyleMaskClosable NSClosableWindowMask
67  #define NSWindowStyleMaskMiniaturizable NSMiniaturizableWindowMask
68  #define NSWindowStyleMaskResizable NSResizableWindowMask
69  #define NSWindowStyleMaskTitled NSTitledWindowMask
70 #endif
71
72 // NOTE: Many Cocoa dynamically linked constants have been renamed and we need
73 //       to build across SDK versions where one is unavailable or deprecated.
74 //       We use the newer names in code and replace them with the older names if
75 //       the deployment target is older than the newer names.
76
77 #if MAC_OS_X_VERSION_MIN_REQUIRED < 101300
78  #define NSPasteboardTypeURL NSURLPboardType
79 #endif
80
81 typedef VkFlags VkMacOSSurfaceCreateFlagsMVK;
82 typedef VkFlags VkMetalSurfaceCreateFlagsEXT;
83
84 typedef struct VkMacOSSurfaceCreateInfoMVK
85 {
86     VkStructureType                 sType;
87     const void*                     pNext;
88     VkMacOSSurfaceCreateFlagsMVK    flags;
89     const void*                     pView;
90 } VkMacOSSurfaceCreateInfoMVK;
91
92 typedef struct VkMetalSurfaceCreateInfoEXT
93 {
94     VkStructureType                 sType;
95     const void*                     pNext;
96     VkMetalSurfaceCreateFlagsEXT    flags;
97     const void*                     pLayer;
98 } VkMetalSurfaceCreateInfoEXT;
99
100 typedef VkResult (APIENTRY *PFN_vkCreateMacOSSurfaceMVK)(VkInstance,const
       VkMacOSSurfaceCreateInfoMVK*,const VkAllocationCallbacks*,VkSurfaceKHR*);
101 typedef VkResult (APIENTRY *PFN_vkCreateMetalSurfaceEXT)(VkInstance,const
       VkMetalSurfaceCreateInfoEXT*,const VkAllocationCallbacks*,VkSurfaceKHR*);
102
103 #define GLFW_COCOA_WINDOW_STATE         _GLFWwindowNS  ns;
104 #define GLFW_COCOA_LIBRARY_WINDOW_STATE _GLFWlibraryNS ns;
105 #define GLFW_COCOA_MONITOR_STATE        _GLFWmonitorNS ns;
106 #define GLFW_COCOA_CURSOR_STATE         _GLFWcursorNS  ns;
107
108 #define GLFW_NSGL_CONTEXT_STATE         _GLFWcontextNSGL nsgl;
109 #define GLFW_NSGL_LIBRARY_CONTEXT_STATE _GLFWlibraryNSGL nsgl;
110
111 // HIToolbox.framework pointer typedefs
112 #define kTISPropertyUnicodeKeyLayoutData _glfw.ns.tis.kPropertyUnicodeKeyLayoutData
113 typedef TISInputSourceRef (*PFN_TISCopyCurrentKeyboardLayoutInputSource)(void);
114 #define TISCopyCurrentKeyboardLayoutInputSource _glfw.ns.tis.CopyCurrentKeyboardLayoutInputSource
115 typedef void* (*PFN_TISGetInputSourceProperty)(TISInputSourceRef,CFStringRef);
116 #define TISGetInputSourceProperty _glfw.ns.tis.GetInputSourceProperty
117 typedef UInt8 (*PFN_LMGetKbdType)(void);
118 #define LMGetKbdType _glfw.ns.tis.GetKbdType
119
120
121 // NSGL-specific per-context data
122 //
123 typedef struct _GLFWcontextNSGL
124 {
125     id              pixelFormat;
126     id              object;
127 } _GLFWcontextNSGL;
128
129 // NSGL-specific global data
130 //
131 typedef struct _GLFWlibraryNSGL
132 {
133     // dlopen handle for OpenGL.framework (for glfwGetProcAddress)
134     CFBundleRef     framework;
135 } _GLFWlibraryNSGL;
136
```

```
137 // Cocoa-specific per-window data
138 //
139 typedef struct _GLFWwindowNS
140 {
141     id              object;
142     id              delegate;
143     id              view;
144     id              layer;
145
146     GLFWbool        maximized;
147     GLFWbool        occluded;
148     GLFWbool        retina;
149
150     // Cached window properties to filter out duplicate events
151     int             width, height;
152     int             fbWidth, fbHeight;
153     float           xscale, yscale;
154
155     // The total sum of the distances the cursor has been warped
156     // since the last cursor motion event was processed
157     // This is kept to counteract Cocoa doing the same internally
158     double          cursorWarpDeltaX, cursorWarpDeltaY;
159 } _GLFWwindowNS;
160
161 // Cocoa-specific global data
162 //
163 typedef struct _GLFWlibraryNS
164 {
165     CGEventSourceRef    eventSource;
166     id                  delegate;
167     GLFWbool            cursorHidden;
168     TISInputSourceRef   inputSource;
169     IOHIDManagerRef     hidManager;
170     id                  unicodeData;
171     id                  helper;
172     id                  keyUpMonitor;
173     id                  nibObjects;
174
175     char                keynames[GLFW_KEY_LAST + 1][17];
176     short int           keycodes[256];
177     short int           scancodes[GLFW_KEY_LAST + 1];
178     char*               clipboardString;
179     CGPoint             cascadePoint;
180     // Where to place the cursor when re-enabled
181     double              restoreCursorPosX, restoreCursorPosY;
182     // The window whose disabled cursor mode is active
183     _GLFWwindow*        disabledCursorWindow;
184
185     struct {
186         CFBundleRef     bundle;
187         PFN_TISCopyCurrentKeyboardLayoutInputSource CopyCurrentKeyboardLayoutInputSource;
188         PFN_TISGetInputSourceProperty GetInputSourceProperty;
189         PFN_LMGetKbdType GetKbdType;
190         CFStringRef     kPropertyUnicodeKeyLayoutData;
191     } tis;
192 } _GLFWlibraryNS;
193
194 // Cocoa-specific per-monitor data
195 //
196 typedef struct _GLFWmonitorNS
197 {
198     CGDirectDisplayID   displayID;
199     CGDisplayModeRef    previousMode;
200     uint32_t            unitNumber;
201     id                  screen;
202     double              fallbackRefreshRate;
203 } _GLFWmonitorNS;
204
205 // Cocoa-specific per-cursor data
206 //
207 typedef struct _GLFWcursorNS
208 {
209     id              object;
210 } _GLFWcursorNS;
211
212
213 GLFWbool _glfwConnectCocoa(int platformID, _GLFWplatform* platform);
214 int _glfwInitCocoa(void);
215 void _glfwTerminateCocoa(void);
216
217 int _glfwCreateWindowCocoa(_GLFWwindow* window, const _GLFWwndconfig* wndconfig, const _GLFWctxconfig*
    ctxconfig, const _GLFWfbconfig* fbconfig);
218 void _glfwDestroyWindowCocoa(_GLFWwindow* window);
219 void _glfwSetWindowTitleCocoa(_GLFWwindow* window, const char* title);
220 void _glfwSetWindowIconCocoa(_GLFWwindow* window, int count, const GLFWimage* images);
221 void _glfwGetWindowPosCocoa(_GLFWwindow* window, int* xpos, int* ypos);
222 void _glfwSetWindowPosCocoa(_GLFWwindow* window, int xpos, int ypos);
```

```
223 void _glfwGetWindowSizeCocoa(_GLFWwindow* window, int* width, int* height);
224 void _glfwSetWindowSizeCocoa(_GLFWwindow* window, int width, int height);
225 void _glfwSetWindowSizeLimitsCocoa(_GLFWwindow* window, int minwidth, int minheight, int maxwidth, int
       maxheight);
226 void _glfwSetWindowAspectRatioCocoa(_GLFWwindow* window, int numer, int denom);
227 void _glfwGetFramebufferSizeCocoa(_GLFWwindow* window, int* width, int* height);
228 void _glfwGetWindowFrameSizeCocoa(_GLFWwindow* window, int* left, int* top, int* right, int* bottom);
229 void _glfwGetWindowContentScaleCocoa(_GLFWwindow* window, float* xscale, float* yscale);
230 void _glfwIconifyWindowCocoa(_GLFWwindow* window);
231 void _glfwRestoreWindowCocoa(_GLFWwindow* window);
232 void _glfwMaximizeWindowCocoa(_GLFWwindow* window);
233 void _glfwShowWindowCocoa(_GLFWwindow* window);
234 void _glfwHideWindowCocoa(_GLFWwindow* window);
235 void _glfwRequestWindowAttentionCocoa(_GLFWwindow* window);
236 void _glfwFocusWindowCocoa(_GLFWwindow* window);
237 void _glfwSetWindowMonitorCocoa(_GLFWwindow* window, _GLFWmonitor* monitor, int xpos, int ypos, int
       width, int height, int refreshRate);
238 int _glfwWindowFocusedCocoa(_GLFWwindow* window);
239 int _glfwWindowIconifiedCocoa(_GLFWwindow* window);
240 int _glfwWindowVisibleCocoa(_GLFWwindow* window);
241 int _glfwWindowMaximizedCocoa(_GLFWwindow* window);
242 int _glfwWindowHoveredCocoa(_GLFWwindow* window);
243 int _glfwFramebufferTransparentCocoa(_GLFWwindow* window);
244 void _glfwSetWindowResizableCocoa(_GLFWwindow* window, GLFWbool enabled);
245 void _glfwSetWindowDecoratedCocoa(_GLFWwindow* window, GLFWbool enabled);
246 void _glfwSetWindowFloatingCocoa(_GLFWwindow* window, GLFWbool enabled);
247 float _glfwGetWindowOpacityCocoa(_GLFWwindow* window);
248 void _glfwSetWindowOpacityCocoa(_GLFWwindow* window, float opacity);
249 void _glfwSetWindowMousePassthroughCocoa(_GLFWwindow* window, GLFWbool enabled);
250
251 void _glfwSetRawMouseMotionCocoa(_GLFWwindow *window, GLFWbool enabled);
252 GLFWbool _glfwRawMouseMotionSupportedCocoa(void);
253
254 void _glfwPollEventsCocoa(void);
255 void _glfwWaitEventsCocoa(void);
256 void _glfwWaitEventsTimeoutCocoa(double timeout);
257 void _glfwPostEmptyEventCocoa(void);
258
259 void _glfwGetCursorPosCocoa(_GLFWwindow* window, double* xpos, double* ypos);
260 void _glfwSetCursorPosCocoa(_GLFWwindow* window, double xpos, double ypos);
261 void _glfwSetCursorModeCocoa(_GLFWwindow* window, int mode);
262 const char* _glfwGetScancodeNameCocoa(int scancode);
263 int _glfwGetKeyScancodeCocoa(int key);
264 int _glfwCreateCursorCocoa(_GLFWcursor* cursor, const GLFWimage* image, int xhot, int yhot);
265 int _glfwCreateStandardCursorCocoa(_GLFWcursor* cursor, int shape);
266 void _glfwDestroyCursorCocoa(_GLFWcursor* cursor);
267 void _glfwSetCursorCocoa(_GLFWwindow* window, _GLFWcursor* cursor);
268 void _glfwSetClipboardStringCocoa(const char* string);
269 const char* _glfwGetClipboardStringCocoa(void);
270
271 EGLenum _glfwGetEGLPlatformCocoa(EGLint** attribs);
272 EGLNativeDisplayType _glfwGetEGLNativeDisplayCocoa(void);
273 EGLNativeWindowType _glfwGetEGLNativeWindowCocoa(_GLFWwindow* window);
274
275 void _glfwGetRequiredInstanceExtensionsCocoa(char** extensions);
276 int _glfwGetPhysicalDevicePresentationSupportCocoa(VkInstance instance, VkPhysicalDevice device,
       uint32_t queuefamily);
277 VkResult _glfwCreateWindowSurfaceCocoa(VkInstance instance, _GLFWwindow* window, const
       VkAllocationCallbacks* allocator, VkSurfaceKHR* surface);
278
279 void _glfwFreeMonitorCocoa(_GLFWmonitor* monitor);
280 void _glfwGetMonitorPosCocoa(_GLFWmonitor* monitor, int* xpos, int* ypos);
281 void _glfwGetMonitorContentScaleCocoa(_GLFWmonitor* monitor, float* xscale, float* yscale);
282 void _glfwGetMonitorWorkareaCocoa(_GLFWmonitor* monitor, int* xpos, int* ypos, int* width, int* height);
283 GLFWvidmode* _glfwGetVideoModesCocoa(_GLFWmonitor* monitor, int* count);
284 void _glfwGetVideoModeCocoa(_GLFWmonitor* monitor, GLFWvidmode* mode);
285 GLFWbool _glfwGetGammaRampCocoa(_GLFWmonitor* monitor, GLFWgammaramp* ramp);
286 void _glfwSetGammaRampCocoa(_GLFWmonitor* monitor, const GLFWgammaramp* ramp);
287
288 void _glfwPollMonitorsCocoa(void);
289 void _glfwSetVideoModeCocoa(_GLFWmonitor* monitor, const GLFWvidmode* desired);
290 void _glfwRestoreVideoModeCocoa(_GLFWmonitor* monitor);
291
292 float _glfwTransformYCocoa(float y);
293
294 void* _glfwLoadLocalVulkanLoaderCocoa(void);
295
296 GLFWbool _glfwInitNSGL(void);
297 void _glfwTerminateNSGL(void);
298 GLFWbool _glfwCreateContextNSGL(_GLFWwindow* window,
299                                 const _GLFWctxconfig* ctxconfig,
300                                 const _GLFWfbconfig* fbconfig);
301 void _glfwDestroyContextNSGL(_GLFWwindow* window);
302
```

27.21 cocoa_time.h

```
1 //========================================================================
2 // GLFW 3.4 macOS - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2009-2021 Camilla Löwy <elmindreda@glfw.org>
5 //
6 // This software is provided 'as-is', without any express or implied
7 // warranty. In no event will the authors be held liable for any damages
8 // arising from the use of this software.
9 //
10 // Permission is granted to anyone to use this software for any purpose,
11 // including commercial applications, and to alter it and redistribute it
12 // freely, subject to the following restrictions:
13 //
14 // 1. The origin of this software must not be misrepresented; you must not
15 //    claim that you wrote the original software. If you use this software
16 //    in a product, an acknowledgment in the product documentation would
17 //    be appreciated but is not required.
18 //
19 // 2. Altered source versions must be plainly marked as such, and must not
20 //    be misrepresented as being the original software.
21 //
22 // 3. This notice may not be removed or altered from any source
23 //    distribution.
24 //
25 //========================================================================
26
27 #define GLFW_COCOA_LIBRARY_TIMER_STATE  _GLFWtimerNS   ns;
28
29 // Cocoa-specific global timer data
30 //
31 typedef struct _GLFWtimerNS
32 {
33     uint64_t        frequency;
34 } _GLFWtimerNS;
35
```

27.22 internal.h

```
1 //========================================================================
2 // GLFW 3.4 - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2002-2006 Marcus Geelnard
5 // Copyright (c) 2006-2019 Camilla Löwy <elmindreda@glfw.org>
6 //
7 // This software is provided 'as-is', without any express or implied
8 // warranty. In no event will the authors be held liable for any damages
9 // arising from the use of this software.
10 //
11 // Permission is granted to anyone to use this software for any purpose,
12 // including commercial applications, and to alter it and redistribute it
13 // freely, subject to the following restrictions:
14 //
15 // 1. The origin of this software must not be misrepresented; you must not
16 //    claim that you wrote the original software. If you use this software
17 //    in a product, an acknowledgment in the product documentation would
18 //    be appreciated but is not required.
19 //
20 // 2. Altered source versions must be plainly marked as such, and must not
21 //    be misrepresented as being the original software.
22 //
23 // 3. This notice may not be removed or altered from any source
24 //    distribution.
25 //
26 //========================================================================
27
28 #pragma once
29
30 #if defined(_GLFW_USE_CONFIG_H)
31  #include "glfw_config.h"
32 #endif
33
34 #if defined(GLFW_INCLUDE_GLCOREARB) || \
35     defined(GLFW_INCLUDE_ES1)       || \
36     defined(GLFW_INCLUDE_ES2)       || \
37     defined(GLFW_INCLUDE_ES3)       || \
38     defined(GLFW_INCLUDE_ES31)      || \
39     defined(GLFW_INCLUDE_ES32)      || \
40     defined(GLFW_INCLUDE_NONE)      || \
41     defined(GLFW_INCLUDE_GLEXT)     || \
42     defined(GLFW_INCLUDE_GLU)       || \
43     defined(GLFW_INCLUDE_VULKAN)    || \
```

```
44     defined(GLFW_DLL)
45  #error "You must not define any header option macros when compiling GLFW"
46 #endif
47
48 #define GLFW_INCLUDE_NONE
49 #include "../include/GLFW/glfw3.h"
50
51 #define _GLFW_INSERT_FIRST      0
52 #define _GLFW_INSERT_LAST       1
53
54 #define _GLFW_POLL_PRESENCE     0
55 #define _GLFW_POLL_AXES         1
56 #define _GLFW_POLL_BUTTONS      2
57 #define _GLFW_POLL_ALL          (_GLFW_POLL_AXES | _GLFW_POLL_BUTTONS)
58
59 #define _GLFW_MESSAGE_SIZE      1024
60
61 typedef int GLFWbool;
62 typedef void (*GLFWproc)(void);
63
64 typedef struct _GLFWerror       _GLFWerror;
65 typedef struct _GLFWinitconfig  _GLFWinitconfig;
66 typedef struct _GLFWwndconfig   _GLFWwndconfig;
67 typedef struct _GLFWctxconfig   _GLFWctxconfig;
68 typedef struct _GLFWfbconfig    _GLFWfbconfig;
69 typedef struct _GLFWcontext     _GLFWcontext;
70 typedef struct _GLFWwindow      _GLFWwindow;
71 typedef struct _GLFWplatform    _GLFWplatform;
72 typedef struct _GLFWlibrary     _GLFWlibrary;
73 typedef struct _GLFWmonitor     _GLFWmonitor;
74 typedef struct _GLFWcursor      _GLFWcursor;
75 typedef struct _GLFWmapelement  _GLFWmapelement;
76 typedef struct _GLFWmapping     _GLFWmapping;
77 typedef struct _GLFWjoystick    _GLFWjoystick;
78 typedef struct _GLFWtls         _GLFWtls;
79 typedef struct _GLFWmutex       _GLFWmutex;
80
81 #define GL_VERSION 0x1f02
82 #define GL_NONE 0
83 #define GL_COLOR_BUFFER_BIT 0x00004000
84 #define GL_UNSIGNED_BYTE 0x1401
85 #define GL_EXTENSIONS 0x1f03
86 #define GL_NUM_EXTENSIONS 0x821d
87 #define GL_CONTEXT_FLAGS 0x821e
88 #define GL_CONTEXT_FLAG_FORWARD_COMPATIBLE_BIT 0x00000001
89 #define GL_CONTEXT_FLAG_DEBUG_BIT 0x00000002
90 #define GL_CONTEXT_PROFILE_MASK 0x9126
91 #define GL_CONTEXT_COMPATIBILITY_PROFILE_BIT 0x00000002
92 #define GL_CONTEXT_CORE_PROFILE_BIT 0x00000001
93 #define GL_RESET_NOTIFICATION_STRATEGY_ARB 0x8256
94 #define GL_LOSE_CONTEXT_ON_RESET_ARB 0x8252
95 #define GL_NO_RESET_NOTIFICATION_ARB 0x8261
96 #define GL_CONTEXT_RELEASE_BEHAVIOR 0x82fb
97 #define GL_CONTEXT_RELEASE_BEHAVIOR_FLUSH 0x82fc
98 #define GL_CONTEXT_FLAG_NO_ERROR_BIT_KHR 0x00000008
99
100 typedef int GLint;
101 typedef unsigned int GLuint;
102 typedef unsigned int GLenum;
103 typedef unsigned int GLbitfield;
104 typedef unsigned char GLubyte;
105
106 typedef void (APIENTRY * PFNGLCLEARPROC)(GLbitfield);
107 typedef const GLubyte* (APIENTRY * PFNGLGETSTRINGPROC)(GLenum);
108 typedef void (APIENTRY * PFNGLGETINTEGERVPROC)(GLenum,GLint*);
109 typedef const GLubyte* (APIENTRY * PFNGLGETSTRINGIPROC)(GLenum,GLuint);
110
111 #if defined(_GLFW_WIN32)
112  #define EGLAPIENTRY __stdcall
113 #else
114  #define EGLAPIENTRY
115 #endif
116
117 #define EGL_SUCCESS 0x3000
118 #define EGL_NOT_INITIALIZED 0x3001
119 #define EGL_BAD_ACCESS 0x3002
120 #define EGL_BAD_ALLOC 0x3003
121 #define EGL_BAD_ATTRIBUTE 0x3004
122 #define EGL_BAD_CONFIG 0x3005
123 #define EGL_BAD_CONTEXT 0x3006
124 #define EGL_BAD_CURRENT_SURFACE 0x3007
125 #define EGL_BAD_DISPLAY 0x3008
126 #define EGL_BAD_MATCH 0x3009
127 #define EGL_BAD_NATIVE_PIXMAP 0x300a
128 #define EGL_BAD_NATIVE_WINDOW 0x300b
129 #define EGL_BAD_PARAMETER 0x300c
130 #define EGL_BAD_SURFACE 0x300d
```

```
131 #define EGL_CONTEXT_LOST 0x300e
132 #define EGL_COLOR_BUFFER_TYPE 0x303f
133 #define EGL_RGB_BUFFER 0x308e
134 #define EGL_SURFACE_TYPE 0x3033
135 #define EGL_WINDOW_BIT 0x0004
136 #define EGL_RENDERABLE_TYPE 0x3040
137 #define EGL_OPENGL_ES_BIT 0x0001
138 #define EGL_OPENGL_ES2_BIT 0x0004
139 #define EGL_OPENGL_BIT 0x0008
140 #define EGL_ALPHA_SIZE 0x3021
141 #define EGL_BLUE_SIZE 0x3022
142 #define EGL_GREEN_SIZE 0x3023
143 #define EGL_RED_SIZE 0x3024
144 #define EGL_DEPTH_SIZE 0x3025
145 #define EGL_STENCIL_SIZE 0x3026
146 #define EGL_SAMPLES 0x3031
147 #define EGL_OPENGL_ES_API 0x30a0
148 #define EGL_OPENGL_API 0x30a2
149 #define EGL_NONE 0x3038
150 #define EGL_RENDER_BUFFER 0x3086
151 #define EGL_SINGLE_BUFFER 0x3085
152 #define EGL_EXTENSIONS 0x3055
153 #define EGL_CONTEXT_CLIENT_VERSION 0x3098
154 #define EGL_NATIVE_VISUAL_ID 0x302e
155 #define EGL_NO_SURFACE ((EGLSurface) 0)
156 #define EGL_NO_DISPLAY ((EGLDisplay) 0)
157 #define EGL_NO_CONTEXT ((EGLContext) 0)
158 #define EGL_DEFAULT_DISPLAY ((EGLNativeDisplayType) 0)
159
160 #define EGL_CONTEXT_OPENGL_FORWARD_COMPATIBLE_BIT_KHR 0x00000002
161 #define EGL_CONTEXT_OPENGL_CORE_PROFILE_BIT_KHR 0x00000001
162 #define EGL_CONTEXT_OPENGL_COMPATIBILITY_PROFILE_BIT_KHR 0x00000002
163 #define EGL_CONTEXT_OPENGL_DEBUG_BIT_KHR 0x00000001
164 #define EGL_CONTEXT_OPENGL_RESET_NOTIFICATION_STRATEGY_KHR 0x31bd
165 #define EGL_NO_RESET_NOTIFICATION_KHR 0x31be
166 #define EGL_LOSE_CONTEXT_ON_RESET_KHR 0x31bf
167 #define EGL_CONTEXT_OPENGL_ROBUST_ACCESS_BIT_KHR 0x00000004
168 #define EGL_CONTEXT_MAJOR_VERSION_KHR 0x3098
169 #define EGL_CONTEXT_MINOR_VERSION_KHR 0x30fb
170 #define EGL_CONTEXT_OPENGL_PROFILE_MASK_KHR 0x30fd
171 #define EGL_CONTEXT_FLAGS_KHR 0x30fc
172 #define EGL_CONTEXT_OPENGL_NO_ERROR_KHR 0x31b3
173 #define EGL_GL_COLORSPACE_KHR 0x309d
174 #define EGL_GL_COLORSPACE_SRGB_KHR 0x3089
175 #define EGL_CONTEXT_RELEASE_BEHAVIOR_KHR 0x2097
176 #define EGL_CONTEXT_RELEASE_BEHAVIOR_NONE_KHR 0
177 #define EGL_CONTEXT_RELEASE_BEHAVIOR_FLUSH_KHR 0x2098
178 #define EGL_PLATFORM_X11_EXT 0x31d5
179 #define EGL_PLATFORM_WAYLAND_EXT 0x31d8
180 #define EGL_PRESENT_OPAQUE_EXT 0x31df
181 #define EGL_PLATFORM_ANGLE_ANGLE 0x3202
182 #define EGL_PLATFORM_ANGLE_TYPE_ANGLE 0x3203
183 #define EGL_PLATFORM_ANGLE_TYPE_OPENGL_ANGLE 0x320d
184 #define EGL_PLATFORM_ANGLE_TYPE_OPENGLES_ANGLE 0x320e
185 #define EGL_PLATFORM_ANGLE_TYPE_D3D9_ANGLE 0x3207
186 #define EGL_PLATFORM_ANGLE_TYPE_D3D11_ANGLE 0x3208
187 #define EGL_PLATFORM_ANGLE_TYPE_VULKAN_ANGLE 0x3450
188 #define EGL_PLATFORM_ANGLE_TYPE_METAL_ANGLE 0x3489
189 #define EGL_PLATFORM_ANGLE_NATIVE_PLATFORM_TYPE_ANGLE 0x348f
190
191 typedef int EGLint;
192 typedef unsigned int EGLBoolean;
193 typedef unsigned int EGLenum;
194 typedef void* EGLConfig;
195 typedef void* EGLContext;
196 typedef void* EGLDisplay;
197 typedef void* EGLSurface;
198
199 typedef void* EGLNativeDisplayType;
200 typedef void* EGLNativeWindowType;
201
202 // EGL function pointer typedefs
203 typedef EGLBoolean (EGLAPIENTRY * PFN_eglGetConfigAttrib)(EGLDisplay,EGLConfig,EGLint,EGLint*);
204 typedef EGLBoolean (EGLAPIENTRY * PFN_eglGetConfigs)(EGLDisplay,EGLConfig*,EGLint,EGLint*);
205 typedef EGLDisplay (EGLAPIENTRY * PFN_eglGetDisplay)(EGLNativeDisplayType);
206 typedef EGLint (EGLAPIENTRY * PFN_eglGetError)(void);
207 typedef EGLBoolean (EGLAPIENTRY * PFN_eglInitialize)(EGLDisplay,EGLint*,EGLint*);
208 typedef EGLBoolean (EGLAPIENTRY * PFN_eglTerminate)(EGLDisplay);
209 typedef EGLBoolean (EGLAPIENTRY * PFN_eglBindAPI)(EGLenum);
210 typedef EGLContext (EGLAPIENTRY * PFN_eglCreateContext)(EGLDisplay,EGLConfig,EGLContext,const EGLint*);
211 typedef EGLBoolean (EGLAPIENTRY * PFN_eglDestroySurface)(EGLDisplay,EGLSurface);
212 typedef EGLBoolean (EGLAPIENTRY * PFN_eglDestroyContext)(EGLDisplay,EGLContext);
213 typedef EGLSurface (EGLAPIENTRY *
        PFN_eglCreateWindowSurface)(EGLDisplay,EGLConfig,EGLNativeWindowType,const EGLint*);
214 typedef EGLBoolean (EGLAPIENTRY * PFN_eglMakeCurrent)(EGLDisplay,EGLSurface,EGLSurface,EGLContext);
215 typedef EGLBoolean (EGLAPIENTRY * PFN_eglSwapBuffers)(EGLDisplay,EGLSurface);
216 typedef EGLBoolean (EGLAPIENTRY * PFN_eglSwapInterval)(EGLDisplay,EGLint);
```

```
217 typedef const char* (EGLAPIENTRY * PFN_eglQueryString)(EGLDisplay,EGLint);
218 typedef GLFWglproc (EGLAPIENTRY * PFN_eglGetProcAddress)(const char*);
219 #define eglGetConfigAttrib _glfw.egl.GetConfigAttrib
220 #define eglGetConfigs _glfw.egl.GetConfigs
221 #define eglGetDisplay _glfw.egl.GetDisplay
222 #define eglGetError _glfw.egl.GetError
223 #define eglInitialize _glfw.egl.Initialize
224 #define eglTerminate _glfw.egl.Terminate
225 #define eglBindAPI _glfw.egl.BindAPI
226 #define eglCreateContext _glfw.egl.CreateContext
227 #define eglDestroySurface _glfw.egl.DestroySurface
228 #define eglDestroyContext _glfw.egl.DestroyContext
229 #define eglCreateWindowSurface _glfw.egl.CreateWindowSurface
230 #define eglMakeCurrent _glfw.egl.MakeCurrent
231 #define eglSwapBuffers _glfw.egl.SwapBuffers
232 #define eglSwapInterval _glfw.egl.SwapInterval
233 #define eglQueryString _glfw.egl.QueryString
234 #define eglGetProcAddress _glfw.egl.GetProcAddress
235
236 typedef EGLDisplay (EGLAPIENTRY * PFNEGLGETPLATFORMDISPLAYEXTPROC)(EGLenum,void*,const EGLint*);
237 typedef EGLSurface (EGLAPIENTRY *
       PFNEGLCREATEPLATFORMWINDOWSURFACEEXTPROC)(EGLDisplay,EGLConfig,void*,const EGLint*);
238 #define eglGetPlatformDisplayEXT _glfw.egl.GetPlatformDisplayEXT
239 #define eglCreatePlatformWindowSurfaceEXT _glfw.egl.CreatePlatformWindowSurfaceEXT
240
241 #define OSMESA_RGBA 0x1908
242 #define OSMESA_FORMAT 0x22
243 #define OSMESA_DEPTH_BITS 0x30
244 #define OSMESA_STENCIL_BITS 0x31
245 #define OSMESA_ACCUM_BITS 0x32
246 #define OSMESA_PROFILE 0x33
247 #define OSMESA_CORE_PROFILE 0x34
248 #define OSMESA_COMPAT_PROFILE 0x35
249 #define OSMESA_CONTEXT_MAJOR_VERSION 0x36
250 #define OSMESA_CONTEXT_MINOR_VERSION 0x37
251
252 typedef void* OSMesaContext;
253 typedef void (*OSMESAproc)(void);
254
255 typedef OSMesaContext (GLAPIENTRY * PFN_OSMesaCreateContextExt)(GLenum,GLint,GLint,GLint,OSMesaContext);
256 typedef OSMesaContext (GLAPIENTRY * PFN_OSMesaCreateContextAttribs)(const int*,OSMesaContext);
257 typedef void (GLAPIENTRY * PFN_OSMesaDestroyContext)(OSMesaContext);
258 typedef int (GLAPIENTRY * PFN_OSMesaMakeCurrent)(OSMesaContext,void*,int,int,int);
259 typedef int (GLAPIENTRY * PFN_OSMesaGetColorBuffer)(OSMesaContext,int*,int*,int*,void**);
260 typedef int (GLAPIENTRY * PFN_OSMesaGetDepthBuffer)(OSMesaContext,int*,int*,int*,void**);
261 typedef GLFWglproc (GLAPIENTRY * PFN_OSMesaGetProcAddress)(const char*);
262 #define OSMesaCreateContextExt _glfw.osmesa.CreateContextExt
263 #define OSMesaCreateContextAttribs _glfw.osmesa.CreateContextAttribs
264 #define OSMesaDestroyContext _glfw.osmesa.DestroyContext
265 #define OSMesaMakeCurrent _glfw.osmesa.MakeCurrent
266 #define OSMesaGetColorBuffer _glfw.osmesa.GetColorBuffer
267 #define OSMesaGetDepthBuffer _glfw.osmesa.GetDepthBuffer
268 #define OSMesaGetProcAddress _glfw.osmesa.GetProcAddress
269
270 #define VK_NULL_HANDLE 0
271
272 typedef void* VkInstance;
273 typedef void* VkPhysicalDevice;
274 typedef uint64_t VkSurfaceKHR;
275 typedef uint32_t VkFlags;
276 typedef uint32_t VkBool32;
277
278 typedef enum VkStructureType
279 {
280     VK_STRUCTURE_TYPE_XLIB_SURFACE_CREATE_INFO_KHR = 1000004000,
281     VK_STRUCTURE_TYPE_XCB_SURFACE_CREATE_INFO_KHR = 1000005000,
282     VK_STRUCTURE_TYPE_WAYLAND_SURFACE_CREATE_INFO_KHR = 1000006000,
283     VK_STRUCTURE_TYPE_WIN32_SURFACE_CREATE_INFO_KHR = 1000009000,
284     VK_STRUCTURE_TYPE_MACOS_SURFACE_CREATE_INFO_MVK = 1000123000,
285     VK_STRUCTURE_TYPE_METAL_SURFACE_CREATE_INFO_EXT = 1000217000,
286     VK_STRUCTURE_TYPE_MAX_ENUM = 0x7FFFFFFF
287 } VkStructureType;
288
289 typedef enum VkResult
290 {
291     VK_SUCCESS = 0,
292     VK_NOT_READY = 1,
293     VK_TIMEOUT = 2,
294     VK_EVENT_SET = 3,
295     VK_EVENT_RESET = 4,
296     VK_INCOMPLETE = 5,
297     VK_ERROR_OUT_OF_HOST_MEMORY = -1,
298     VK_ERROR_OUT_OF_DEVICE_MEMORY = -2,
299     VK_ERROR_INITIALIZATION_FAILED = -3,
300     VK_ERROR_DEVICE_LOST = -4,
301     VK_ERROR_MEMORY_MAP_FAILED = -5,
302     VK_ERROR_LAYER_NOT_PRESENT = -6,
```

```
303     VK_ERROR_EXTENSION_NOT_PRESENT = -7,
304     VK_ERROR_FEATURE_NOT_PRESENT = -8,
305     VK_ERROR_INCOMPATIBLE_DRIVER = -9,
306     VK_ERROR_TOO_MANY_OBJECTS = -10,
307     VK_ERROR_FORMAT_NOT_SUPPORTED = -11,
308     VK_ERROR_SURFACE_LOST_KHR = -1000000000,
309     VK_SUBOPTIMAL_KHR = 1000001003,
310     VK_ERROR_OUT_OF_DATE_KHR = -1000001004,
311     VK_ERROR_INCOMPATIBLE_DISPLAY_KHR = -1000003001,
312     VK_ERROR_NATIVE_WINDOW_IN_USE_KHR = -1000000001,
313     VK_ERROR_VALIDATION_FAILED_EXT = -1000011001,
314     VK_RESULT_MAX_ENUM = 0x7FFFFFFF
315 } VkResult;
316
317 typedef struct VkAllocationCallbacks VkAllocationCallbacks;
318
319 typedef struct VkExtensionProperties
320 {
321     char            extensionName[256];
322     uint32_t        specVersion;
323 } VkExtensionProperties;
324
325 typedef void (APIENTRY * PFN_vkVoidFunction)(void);
326
327 typedef PFN_vkVoidFunction (APIENTRY * PFN_vkGetInstanceProcAddr)(VkInstance,const char*);
328 typedef VkResult (APIENTRY * PFN_vkEnumerateInstanceExtensionProperties)(const
       char*,uint32_t*,VkExtensionProperties*);
329 #define vkGetInstanceProcAddr _glfw.vk.GetInstanceProcAddr
330
331 #include "platform.h"
332
333 // Constructs a version number string from the public header macros
334 #define _GLFW_CONCAT_VERSION(m, n, r) #m "." #n "." #r
335 #define _GLFW_MAKE_VERSION(m, n, r) _GLFW_CONCAT_VERSION(m, n, r)
336 #define _GLFW_VERSION_NUMBER _GLFW_MAKE_VERSION(GLFW_VERSION_MAJOR, \
337                                                 GLFW_VERSION_MINOR, \
338                                                 GLFW_VERSION_REVISION)
339
340 // Checks for whether the library has been initialized
341 #define _GLFW_REQUIRE_INIT()                            \
342     if (!_glfw.initialized)                             \
343     {                                                   \
344         _glfwInputError(GLFW_NOT_INITIALIZED, NULL); \
345         return;                                         \
346     }
347 #define _GLFW_REQUIRE_INIT_OR_RETURN(x)                 \
348     if (!_glfw.initialized)                             \
349     {                                                   \
350         _glfwInputError(GLFW_NOT_INITIALIZED, NULL); \
351         return x;                                       \
352     }
353
354 // Swaps the provided pointers
355 #define _GLFW_SWAP(type, x, y) \
356     {                          \
357         type t;                \
358         t = x;                 \
359         x = y;                 \
360         y = t;                 \
361     }
362
363 // Per-thread error structure
364 //
365 struct _GLFWerror
366 {
367     _GLFWerror*     next;
368     int             code;
369     char            description[_GLFW_MESSAGE_SIZE];
370 };
371
372 // Initialization configuration
373 //
374 // Parameters relating to the initialization of the library
375 //
376 struct _GLFWinitconfig
377 {
378     GLFWbool      hatButtons;
379     int           angleType;
380     int           platformID;
381     PFN_vkGetInstanceProcAddr vulkanLoader;
382     struct {
383         GLFWbool  menubar;
384         GLFWbool  chdir;
385     } ns;
386     struct {
387         GLFWbool  xcbVulkanSurface;
388     } x11;
```

```
389 };
390
391 // Window configuration
392 //
393 // Parameters relating to the creation of the window but not directly related
394 // to the framebuffer.  This is used to pass window creation parameters from
395 // shared code to the platform API.
396 //
397 struct _GLFWwndconfig
398 {
399     int           width;
400     int           height;
401     const char*   title;
402     GLFWbool      resizable;
403     GLFWbool      visible;
404     GLFWbool      decorated;
405     GLFWbool      focused;
406     GLFWbool      autoIconify;
407     GLFWbool      floating;
408     GLFWbool      maximized;
409     GLFWbool      centerCursor;
410     GLFWbool      focusOnShow;
411     GLFWbool      mousePassthrough;
412     GLFWbool      scaleToMonitor;
413     struct {
414         GLFWbool  retina;
415         char      frameName[256];
416     } ns;
417     struct {
418         char      className[256];
419         char      instanceName[256];
420     } x11;
421     struct {
422         GLFWbool  keymenu;
423     } win32;
424 };
425
426 // Context configuration
427 //
428 // Parameters relating to the creation of the context but not directly related
429 // to the framebuffer.  This is used to pass context creation parameters from
430 // shared code to the platform API.
431 //
432 struct _GLFWctxconfig
433 {
434     int           client;
435     int           source;
436     int           major;
437     int           minor;
438     GLFWbool      forward;
439     GLFWbool      debug;
440     GLFWbool      noerror;
441     int           profile;
442     int           robustness;
443     int           release;
444     _GLFWwindow*  share;
445     struct {
446         GLFWbool  offline;
447     } nsgl;
448 };
449
450 // Framebuffer configuration
451 //
452 // This describes buffers and their sizes.  It also contains
453 // a platform-specific ID used to map back to the backend API object.
454 //
455 // It is used to pass framebuffer parameters from shared code to the platform
456 // API and also to enumerate and select available framebuffer configs.
457 //
458 struct _GLFWfbconfig
459 {
460     int         redBits;
461     int         greenBits;
462     int         blueBits;
463     int         alphaBits;
464     int         depthBits;
465     int         stencilBits;
466     int         accumRedBits;
467     int         accumGreenBits;
468     int         accumBlueBits;
469     int         accumAlphaBits;
470     int         auxBuffers;
471     GLFWbool    stereo;
472     int         samples;
473     GLFWbool    sRGB;
474     GLFWbool    doublebuffer;
475     GLFWbool    transparent;
```

```
476     uintptr_t   handle;
477 };
478
479 // Context structure
480 //
481 struct _GLFWcontext
482 {
483     int                 client;
484     int                 source;
485     int                 major, minor, revision;
486     GLFWbool            forward, debug, noerror;
487     int                 profile;
488     int                 robustness;
489     int                 release;
490
491     PFNGLGETSTRINGIPROC  GetStringi;
492     PFNGLGETINTEGERVPROC GetIntegerv;
493     PFNGLGETSTRINGPROC   GetString;
494
495     void (*makeCurrent)(_GLFWwindow*);
496     void (*swapBuffers)(_GLFWwindow*);
497     void (*swapInterval)(int);
498     int (*extensionSupported)(const char*);
499     GLFWglproc (*getProcAddress)(const char*);
500     void (*destroy)(_GLFWwindow*);
501
502     struct {
503         EGLConfig       config;
504         EGLContext      handle;
505         EGLSurface      surface;
506         void*           client;
507     } egl;
508
509     struct {
510         OSMesaContext   handle;
511         int             width;
512         int             height;
513         void*           buffer;
514     } osmesa;
515
516     // This is defined in platform.h
517     GLFW_PLATFORM_CONTEXT_STATE
518 };
519
520 // Window and context structure
521 //
522 struct _GLFWwindow
523 {
524     struct _GLFWwindow* next;
525
526     // Window settings and state
527     GLFWbool            resizable;
528     GLFWbool            decorated;
529     GLFWbool            autoIconify;
530     GLFWbool            floating;
531     GLFWbool            focusOnShow;
532     GLFWbool            mousePassthrough;
533     GLFWbool            shouldClose;
534     void*               userPointer;
535     GLFWbool            doublebuffer;
536     GLFWvidmode         videoMode;
537     _GLFWmonitor*       monitor;
538     _GLFWcursor*        cursor;
539
540     int                 minwidth, minheight;
541     int                 maxwidth, maxheight;
542     int                 numer, denom;
543
544     GLFWbool            stickyKeys;
545     GLFWbool            stickyMouseButtons;
546     GLFWbool            lockKeyMods;
547     int                 cursorMode;
548     char                mouseButtons[GLFW_MOUSE_BUTTON_LAST + 1];
549     char                keys[GLFW_KEY_LAST + 1];
550     // Virtual cursor position when cursor is disabled
551     double              virtualCursorPosX, virtualCursorPosY;
552     GLFWbool            rawMouseMotion;
553
554     _GLFWcontext        context;
555
556     struct {
557         GLFWwindowposfun          pos;
558         GLFWwindowsizefun         size;
559         GLFWwindowclosefun        close;
560         GLFWwindowrefreshfun      refresh;
561         GLFWwindowfocusfun        focus;
562         GLFWwindowiconifyfun      iconify;
```

```
563         GLFWwindowmaximizefun     maximize;
564         GLFWframebuffersizefun    fbsize;
565         GLFWwindowcontentscalefun scale;
566         GLFWmousebuttonfun        mouseButton;
567         GLFWcursorposfun          cursorPos;
568         GLFWcursorenterfun        cursorEnter;
569         GLFWscrollfun             scroll;
570         GLFWkeyfun                key;
571         GLFWcharfun               character;
572         GLFWcharmodsfun           charmods;
573         GLFWdropfun               drop;
574     } callbacks;
575
576     // This is defined in platform.h
577     GLFW_PLATFORM_WINDOW_STATE
578 };
579
580 // Monitor structure
581 //
582 struct _GLFWmonitor
583 {
584     char            name[128];
585     void*           userPointer;
586
587     // Physical dimensions in millimeters.
588     int             widthMM, heightMM;
589
590     // The window whose video mode is current on this monitor
591     _GLFWwindow*    window;
592
593     GLFWvidmode*    modes;
594     int             modeCount;
595     GLFWvidmode     currentMode;
596
597     GLFWgammaramp   originalRamp;
598     GLFWgammaramp   currentRamp;
599
600     // This is defined in platform.h
601     GLFW_PLATFORM_MONITOR_STATE
602 };
603
604 // Cursor structure
605 //
606 struct _GLFWcursor
607 {
608     _GLFWcursor*    next;
609     // This is defined in platform.h
610     GLFW_PLATFORM_CURSOR_STATE
611 };
612
613 // Gamepad mapping element structure
614 //
615 struct _GLFWmapelement
616 {
617     uint8_t         type;
618     uint8_t         index;
619     int8_t          axisScale;
620     int8_t          axisOffset;
621 };
622
623 // Gamepad mapping structure
624 //
625 struct _GLFWmapping
626 {
627     char            name[128];
628     char            guid[33];
629     _GLFWmapelement buttons[15];
630     _GLFWmapelement axes[6];
631 };
632
633 // Joystick structure
634 //
635 struct _GLFWjoystick
636 {
637     GLFWbool        present;
638     float*          axes;
639     int             axisCount;
640     unsigned char*  buttons;
641     int             buttonCount;
642     unsigned char*  hats;
643     int             hatCount;
644     char            name[128];
645     void*           userPointer;
646     char            guid[33];
647     _GLFWmapping*   mapping;
648
649     // This is defined in platform.h
```

```
650     GLFW_PLATFORM_JOYSTICK_STATE
651 };
652
653 // Thread local storage structure
654 //
655 struct _GLFWtls
656 {
657     // This is defined in platform.h
658     GLFW_PLATFORM_TLS_STATE
659 };
660
661 // Mutex structure
662 //
663 struct _GLFWmutex
664 {
665     // This is defined in platform.h
666     GLFW_PLATFORM_MUTEX_STATE
667 };
668
669 // Platform API structure
670 //
671 struct _GLFWplatform
672 {
673     int platformID;
674     // init
675     GLFWbool (*init)(void);
676     void (*terminate)(void);
677     // input
678     void (*getCursorPos)(_GLFWwindow*,double*,double*);
679     void (*setCursorPos)(_GLFWwindow*,double,double);
680     void (*setCursorMode)(_GLFWwindow*,int);
681     void (*setRawMouseMotion)(_GLFWwindow*,GLFWbool);
682     GLFWbool (*rawMouseMotionSupported)(void);
683     int (*createCursor)(_GLFWcursor*,const GLFWimage*,int,int);
684     int (*createStandardCursor)(_GLFWcursor*,int);
685     void (*destroyCursor)(_GLFWcursor*);
686     void (*setCursor)(_GLFWwindow*,_GLFWcursor*);
687     const char* (*getScancodeName)(int);
688     int (*getKeyScancode)(int);
689     void (*setClipboardString)(const char*);
690     const char* (*getClipboardString)(void);
691     GLFWbool (*initJoysticks)(void);
692     void (*terminateJoysticks)(void);
693     int (*pollJoystick)(_GLFWjoystick*,int);
694     const char* (*getMappingName)(void);
695     void (*updateGamepadGUID)(char*);
696     // monitor
697     void (*freeMonitor)(_GLFWmonitor*);
698     void (*getMonitorPos)(_GLFWmonitor*,int*,int*);
699     void (*getMonitorContentScale)(_GLFWmonitor*,float*,float*);
700     void (*getMonitorWorkarea)(_GLFWmonitor*,int*,int*,int*,int*);
701     GLFWvidmode* (*getVideoModes)(_GLFWmonitor*,int*);
702     void (*getVideoMode)(_GLFWmonitor*,GLFWvidmode*);
703     GLFWbool (*getGammaRamp)(_GLFWmonitor*,GLFWgammaramp*);
704     void (*setGammaRamp)(_GLFWmonitor*,const GLFWgammaramp*);
705     // window
706     int (*createWindow)(_GLFWwindow*,const _GLFWwndconfig*,const _GLFWctxconfig*,const _GLFWfbconfig*);
707     void (*destroyWindow)(_GLFWwindow*);
708     void (*setWindowTitle)(_GLFWwindow*,const char*);
709     void (*setWindowIcon)(_GLFWwindow*,int,const GLFWimage*);
710     void (*getWindowPos)(_GLFWwindow*,int*,int*);
711     void (*setWindowPos)(_GLFWwindow*,int,int);
712     void (*getWindowSize)(_GLFWwindow*,int*,int*);
713     void (*setWindowSize)(_GLFWwindow*,int,int);
714     void (*setWindowSizeLimits)(_GLFWwindow*,int,int,int,int);
715     void (*setWindowAspectRatio)(_GLFWwindow*,int,int);
716     void (*getFramebufferSize)(_GLFWwindow*,int*,int*);
717     void (*getWindowFrameSize)(_GLFWwindow*,int*,int*,int*,int*);
718     void (*getWindowContentScale)(_GLFWwindow*,float*,float*);
719     void (*iconifyWindow)(_GLFWwindow*);
720     void (*restoreWindow)(_GLFWwindow*);
721     void (*maximizeWindow)(_GLFWwindow*);
722     void (*showWindow)(_GLFWwindow*);
723     void (*hideWindow)(_GLFWwindow*);
724     void (*requestWindowAttention)(_GLFWwindow*);
725     void (*focusWindow)(_GLFWwindow*);
726     void (*setWindowMonitor)(_GLFWwindow*,_GLFWmonitor*,int,int,int,int,int);
727     int (*windowFocused)(_GLFWwindow*);
728     int (*windowIconified)(_GLFWwindow*);
729     int (*windowVisible)(_GLFWwindow*);
730     int (*windowMaximized)(_GLFWwindow*);
731     int (*windowHovered)(_GLFWwindow*);
732     int (*framebufferTransparent)(_GLFWwindow*);
733     float (*getWindowOpacity)(_GLFWwindow*);
734     void (*setWindowResizable)(_GLFWwindow*,GLFWbool);
735     void (*setWindowDecorated)(_GLFWwindow*,GLFWbool);
736     void (*setWindowFloating)(_GLFWwindow*,GLFWbool);
```

```
737     void (*setWindowOpacity)(_GLFWwindow*,float);
738     void (*setWindowMousePassthrough)(_GLFWwindow*,GLFWbool);
739     void (*pollEvents)(void);
740     void (*waitEvents)(void);
741     void (*waitEventsTimeout)(double);
742     void (*postEmptyEvent)(void);
743     // EGL
744     EGLenum (*getEGLPlatform)(EGLint**);
745     EGLNativeDisplayType (*getEGLNativeDisplay)(void);
746     EGLNativeWindowType (*getEGLNativeWindow)(_GLFWwindow*);
747     // vulkan
748     void (*getRequiredInstanceExtensions)(char**);
749     int (*getPhysicalDevicePresentationSupport)(VkInstance,VkPhysicalDevice,uint32_t);
750     VkResult (*createWindowSurface)(VkInstance,_GLFWwindow*,const VkAllocationCallbacks*,VkSurfaceKHR*);
751 };
752
753 // Library global data
754 //
755 struct _GLFWlibrary
756 {
757     GLFWbool            initialized;
758     GLFWallocator       allocator;
759
760     _GLFWplatform       platform;
761
762     struct {
763         _GLFWinitconfig init;
764         _GLFWfbconfig   framebuffer;
765         _GLFWwndconfig  window;
766         _GLFWctxconfig  context;
767         int             refreshRate;
768     } hints;
769
770     _GLFWerror*         errorListHead;
771     _GLFWcursor*        cursorListHead;
772     _GLFWwindow*        windowListHead;
773
774     _GLFWmonitor**      monitors;
775     int                 monitorCount;
776
777     GLFWbool            joysticksInitialized;
778     _GLFWjoystick       joysticks[GLFW_JOYSTICK_LAST + 1];
779     _GLFWmapping*       mappings;
780     int                 mappingCount;
781
782     _GLFWtls            errorSlot;
783     _GLFWtls            contextSlot;
784     _GLFWmutex          errorLock;
785
786     struct {
787         uint64_t        offset;
788         // This is defined in platform.h
789         GLFW_PLATFORM_LIBRARY_TIMER_STATE
790     } timer;
791
792     struct {
793         EGLenum         platform;
794         EGLDisplay      display;
795         EGLint          major, minor;
796         GLFWbool        prefix;
797
798         GLFWbool        KHR_create_context;
799         GLFWbool        KHR_create_context_no_error;
800         GLFWbool        KHR_gl_colorspace;
801         GLFWbool        KHR_get_all_proc_addresses;
802         GLFWbool        KHR_context_flush_control;
803         GLFWbool        EXT_client_extensions;
804         GLFWbool        EXT_platform_base;
805         GLFWbool        EXT_platform_x11;
806         GLFWbool        EXT_platform_wayland;
807         GLFWbool        EXT_present_opaque;
808         GLFWbool        ANGLE_platform_angle;
809         GLFWbool        ANGLE_platform_angle_opengl;
810         GLFWbool        ANGLE_platform_angle_d3d;
811         GLFWbool        ANGLE_platform_angle_vulkan;
812         GLFWbool        ANGLE_platform_angle_metal;
813
814         void*           handle;
815
816         PFN_eglGetConfigAttrib      GetConfigAttrib;
817         PFN_eglGetConfigs           GetConfigs;
818         PFN_eglGetDisplay           GetDisplay;
819         PFN_eglGetError             GetError;
820         PFN_eglInitialize           Initialize;
821         PFN_eglTerminate            Terminate;
822         PFN_eglBindAPI              BindAPI;
823         PFN_eglCreateContext        CreateContext;
```

```
824         PFN_eglDestroySurface       DestroySurface;
825         PFN_eglDestroyContext       DestroyContext;
826         PFN_eglCreateWindowSurface  CreateWindowSurface;
827         PFN_eglMakeCurrent          MakeCurrent;
828         PFN_eglSwapBuffers          SwapBuffers;
829         PFN_eglSwapInterval         SwapInterval;
830         PFN_eglQueryString          QueryString;
831         PFN_eglGetProcAddress       GetProcAddress;
832
833         PFNEGLGETPLATFORMDISPLAYEXTPROC GetPlatformDisplayEXT;
834         PFNEGLCREATEPLATFORMWINDOWSURFACEEXTPROC CreatePlatformWindowSurfaceEXT;
835     } egl;
836
837     struct {
838         void*           handle;
839
840         PFN_OSMesaCreateContextExt      CreateContextExt;
841         PFN_OSMesaCreateContextAttribs  CreateContextAttribs;
842         PFN_OSMesaDestroyContext        DestroyContext;
843         PFN_OSMesaMakeCurrent           MakeCurrent;
844         PFN_OSMesaGetColorBuffer        GetColorBuffer;
845         PFN_OSMesaGetDepthBuffer        GetDepthBuffer;
846         PFN_OSMesaGetProcAddress        GetProcAddress;
847
848     } osmesa;
849
850     struct {
851         GLFWbool        available;
852         void*           handle;
853         char*           extensions[2];
854         PFN_vkGetInstanceProcAddr GetInstanceProcAddr;
855         GLFWbool        KHR_surface;
856         GLFWbool        KHR_win32_surface;
857         GLFWbool        MVK_macos_surface;
858         GLFWbool        EXT_metal_surface;
859         GLFWbool        KHR_xlib_surface;
860         GLFWbool        KHR_xcb_surface;
861         GLFWbool        KHR_wayland_surface;
862     } vk;
863
864     struct {
865         GLFWmonitorfun  monitor;
866         GLFWjoystickfun joystick;
867     } callbacks;
868
869     // These are defined in platform.h
870     GLFW_PLATFORM_LIBRARY_WINDOW_STATE
871     GLFW_PLATFORM_LIBRARY_CONTEXT_STATE
872     GLFW_PLATFORM_LIBRARY_JOYSTICK_STATE
873 };
874
875 // Global state shared between compilation units of GLFW
876 //
877 extern _GLFWlibrary _glfw;
878
879
883
884 void _glfwPlatformInitTimer(void);
885 uint64_t _glfwPlatformGetTimerValue(void);
886 uint64_t _glfwPlatformGetTimerFrequency(void);
887
888 GLFWbool _glfwPlatformCreateTls(_GLFWtls* tls);
889 void _glfwPlatformDestroyTls(_GLFWtls* tls);
890 void* _glfwPlatformGetTls(_GLFWtls* tls);
891 void _glfwPlatformSetTls(_GLFWtls* tls, void* value);
892
893 GLFWbool _glfwPlatformCreateMutex(_GLFWmutex* mutex);
894 void _glfwPlatformDestroyMutex(_GLFWmutex* mutex);
895 void _glfwPlatformLockMutex(_GLFWmutex* mutex);
896 void _glfwPlatformUnlockMutex(_GLFWmutex* mutex);
897
898 void* _glfwPlatformLoadModule(const char* path);
899 void _glfwPlatformFreeModule(void* module);
900 GLFWproc _glfwPlatformGetModuleSymbol(void* module, const char* name);
901
902
906
907 void _glfwInputWindowFocus(_GLFWwindow* window, GLFWbool focused);
908 void _glfwInputWindowPos(_GLFWwindow* window, int xpos, int ypos);
909 void _glfwInputWindowSize(_GLFWwindow* window, int width, int height);
910 void _glfwInputFramebufferSize(_GLFWwindow* window, int width, int height);
911 void _glfwInputWindowContentScale(_GLFWwindow* window,
912                                   float xscale, float yscale);
913 void _glfwInputWindowIconify(_GLFWwindow* window, GLFWbool iconified);
914 void _glfwInputWindowMaximize(_GLFWwindow* window, GLFWbool maximized);
915 void _glfwInputWindowDamage(_GLFWwindow* window);
916 void _glfwInputWindowCloseRequest(_GLFWwindow* window);
```

```
917 void _glfwInputWindowMonitor(_GLFWwindow* window, _GLFWmonitor* monitor);
918
919 void _glfwInputKey(_GLFWwindow* window,
920                    int key, int scancode, int action, int mods);
921 void _glfwInputChar(_GLFWwindow* window,
922                     uint32_t codepoint, int mods, GLFWbool plain);
923 void _glfwInputScroll(_GLFWwindow* window, double xoffset, double yoffset);
924 void _glfwInputMouseClick(_GLFWwindow* window, int button, int action, int mods);
925 void _glfwInputCursorPos(_GLFWwindow* window, double xpos, double ypos);
926 void _glfwInputCursorEnter(_GLFWwindow* window, GLFWbool entered);
927 void _glfwInputDrop(_GLFWwindow* window, int count, const char** names);
928 void _glfwInputJoystick(_GLFWjoystick* js, int event);
929 void _glfwInputJoystickAxis(_GLFWjoystick* js, int axis, float value);
930 void _glfwInputJoystickButton(_GLFWjoystick* js, int button, char value);
931 void _glfwInputJoystickHat(_GLFWjoystick* js, int hat, char value);
932
933 void _glfwInputMonitor(_GLFWmonitor* monitor, int action, int placement);
934 void _glfwInputMonitorWindow(_GLFWmonitor* monitor, _GLFWwindow* window);
935
936 #if defined(__GNUC__)
937 void _glfwInputError(int code, const char* format, ...)
938     __attribute__((format(printf, 2, 3)));
939 #else
940 void _glfwInputError(int code, const char* format, ...);
941 #endif
942
943
947
948 GLFWbool _glfwSelectPlatform(int platformID, _GLFWplatform* platform);
949
950 GLFWbool _glfwStringInExtensionString(const char* string, const char* extensions);
951 const _GLFWfbconfig* _glfwChooseFBConfig(const _GLFWfbconfig* desired,
952                                          const _GLFWfbconfig* alternatives,
953                                          unsigned int count);
954 GLFWbool _glfwRefreshContextAttribs(_GLFWwindow* window,
955                                     const _GLFWctxconfig* ctxconfig);
956 GLFWbool _glfwIsValidContextConfig(const _GLFWctxconfig* ctxconfig);
957
958 const GLFWvidmode* _glfwChooseVideoMode(_GLFWmonitor* monitor,
959                                         const GLFWvidmode* desired);
960 int _glfwCompareVideoModes(const GLFWvidmode* first, const GLFWvidmode* second);
961 _GLFWmonitor* _glfwAllocMonitor(const char* name, int widthMM, int heightMM);
962 void _glfwFreeMonitor(_GLFWmonitor* monitor);
963 void _glfwAllocGammaArrays(GLFWgammaramp* ramp, unsigned int size);
964 void _glfwFreeGammaArrays(GLFWgammaramp* ramp);
965 void _glfwSplitBPP(int bpp, int* red, int* green, int* blue);
966
967 void _glfwInitGamepadMappings(void);
968 _GLFWjoystick* _glfwAllocJoystick(const char* name,
969                                   const char* guid,
970                                   int axisCount,
971                                   int buttonCount,
972                                   int hatCount);
973 void _glfwFreeJoystick(_GLFWjoystick* js);
974 void _glfwCenterCursorInContentArea(_GLFWwindow* window);
975
976 GLFWbool _glfwInitEGL(void);
977 void _glfwTerminateEGL(void);
978 GLFWbool _glfwCreateContextEGL(_GLFWwindow* window,
979                                const _GLFWctxconfig* ctxconfig,
980                                const _GLFWfbconfig* fbconfig);
981 #if defined(_GLFW_X11)
982 GLFWbool _glfwChooseVisualEGL(const _GLFWwndconfig* wndconfig,
983                               const _GLFWctxconfig* ctxconfig,
984                               const _GLFWfbconfig* fbconfig,
985                               Visual** visual, int* depth);
986 #endif /*_GLFW_X11*/
987
988 GLFWbool _glfwInitOSMesa(void);
989 void _glfwTerminateOSMesa(void);
990 GLFWbool _glfwCreateContextOSMesa(_GLFWwindow* window,
991                                   const _GLFWctxconfig* ctxconfig,
992                                   const _GLFWfbconfig* fbconfig);
993
994 GLFWbool _glfwInitVulkan(int mode);
995 void _glfwTerminateVulkan(void);
996 const char* _glfwGetVulkanResultString(VkResult result);
997
998 size_t _glfwEncodeUTF8(char* s, uint32_t codepoint);
999
1000 char* _glfw_strdup(const char* source);
1001 int _glfw_min(int a, int b);
1002 int _glfw_max(int a, int b);
1003 float _glfw_fminf(float a, float b);
1004 float _glfw_fmaxf(float a, float b);
1005
1006 void* _glfw_calloc(size_t count, size_t size);
```

```
1007 void* _glfw_realloc(void* pointer, size_t size);
1008 void _glfw_free(void* pointer);
1009
```

27.23 linux_joystick.h

```
1 //========================================================================
2 // GLFW 3.4 Linux - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2014 Jonas Ådahl <jadahl@gmail.com>
5 //
6 // This software is provided 'as-is', without any express or implied
7 // warranty. In no event will the authors be held liable for any damages
8 // arising from the use of this software.
9 //
10 // Permission is granted to anyone to use this software for any purpose,
11 // including commercial applications, and to alter it and redistribute it
12 // freely, subject to the following restrictions:
13 //
14 // 1. The origin of this software must not be misrepresented; you must not
15 //    claim that you wrote the original software. If you use this software
16 //    in a product, an acknowledgment in the product documentation would
17 //    be appreciated but is not required.
18 //
19 // 2. Altered source versions must be plainly marked as such, and must not
20 //    be misrepresented as being the original software.
21 //
22 // 3. This notice may not be removed or altered from any source
23 //    distribution.
24 //
25 //========================================================================
26
27 #include <linux/input.h>
28 #include <linux/limits.h>
29 #include <regex.h>
30
31 #define GLFW_LINUX_JOYSTICK_STATE         _GLFWjoystickLinux linjs;
32 #define GLFW_LINUX_LIBRARY_JOYSTICK_STATE _GLFWlibraryLinux  linjs;
33
34 #define GLFW_BUILD_LINUX_MAPPINGS
35
36 // Linux-specific joystick data
37 //
38 typedef struct _GLFWjoystickLinux
39 {
40     int                     fd;
41     char                    path[PATH_MAX];
42     int                     keyMap[KEY_CNT - BTN_MISC];
43     int                     absMap[ABS_CNT];
44     struct input_absinfo    absInfo[ABS_CNT];
45     int                     hats[4][2];
46 } _GLFWjoystickLinux;
47
48 // Linux-specific joystick API data
49 //
50 typedef struct _GLFWlibraryLinux
51 {
52     int                     inotify;
53     int                     watch;
54     regex_t                 regex;
55     GLFWbool                dropped;
56 } _GLFWlibraryLinux;
57
58 void _glfwDetectJoystickConnectionLinux(void);
59
60 GLFWbool _glfwInitJoysticksLinux(void);
61 void _glfwTerminateJoysticksLinux(void);
62 int _glfwPollJoystickLinux(_GLFWjoystick* js, int mode);
63 const char* _glfwGetMappingNameLinux(void);
64 void _glfwUpdateGamepadGUIDLinux(char* guid);
65
```

27.24 mappings.h

```
1 //========================================================================
2 // GLFW 3.4 - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2006-2018 Camilla Löwy <elmindreda@glfw.org>
5 //
```

```
6 // This software is provided 'as-is', without any express or implied
7 // warranty. In no event will the authors be held liable for any damages
8 // arising from the use of this software.
9 //
10 // Permission is granted to anyone to use this software for any purpose,
11 // including commercial applications, and to alter it and redistribute it
12 // freely, subject to the following restrictions:
13 //
14 // 1. The origin of this software must not be misrepresented; you must not
15 //    claim that you wrote the original software. If you use this software
16 //    in a product, an acknowledgment in the product documentation would
17 //    be appreciated but is not required.
18 //
19 // 2. Altered source versions must be plainly marked as such, and must not
20 //    be misrepresented as being the original software.
21 //
22 // 3. This notice may not be removed or altered from any source
23 //    distribution.
24 //
25 //========================================================================
26 // As mappings.h.in, this file is used by CMake to produce the mappings.h
27 // header file.  If you are adding a GLFW specific gamepad mapping, this is
28 // where to put it.
29 //========================================================================
30 // As mappings.h, this provides all pre-defined gamepad mappings, including
31 // all available in SDL_GameControllerDB.  Do not edit this file.  Any gamepad
32 // mappings not specific to GLFW should be submitted to SDL_GameControllerDB.
33 // This file can be re-generated from mappings.h.in and the upstream
34 // gamecontrollerdb.txt with the 'update_mappings' CMake target.
35 //========================================================================
36
37 // All gamepad mappings not labeled GLFW are copied from the
38 // SDL_GameControllerDB project under the following license:
39 //
40 // Simple DirectMedia Layer
41 // Copyright (C) 1997-2013 Sam Lantinga <slouken@libsdl.org>
42 //
43 // This software is provided 'as-is', without any express or implied warranty.
44 // In no event will the authors be held liable for any damages arising from the
45 // use of this software.
46 //
47 // Permission is granted to anyone to use this software for any purpose,
48 // including commercial applications, and to alter it and redistribute it
49 // freely, subject to the following restrictions:
50 //
51 // 1. The origin of this software must not be misrepresented; you must not
52 //    claim that you wrote the original software. If you use this software
53 //    in a product, an acknowledgment in the product documentation would
54 //    be appreciated but is not required.
55 //
56 // 2. Altered source versions must be plainly marked as such, and must not be
57 //    misrepresented as being the original software.
58 //
59 // 3. This notice may not be removed or altered from any source distribution.
60
61 const char* _glfwDefaultMappings[] =
62 {
63 #if defined(GLFW_BUILD_WIN32_MAPPINGS)
64 "03000000fa2d00001000000000000000,3DRUDDER,leftx:a0,lefty:a1,rightx:a5,righty:a2,platform:Windows,",
65
      "03000000c82d00002038000000000000,8bitdo,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,le
66 "03000000c82d00009510000000000000,8BitDo Dogbone
      Modkit,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,start:b11,platform:Windows,",
67 "03000000c82d000011ab000000000000,8BitDo
      F30,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrigge
68 "03000000c82d00001038000000000000,8BitDo F30
      Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrigge
69 "03000000c82d00000090000000000000,8BitDo FC30
      Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrigge
70 "03000000c82d00000650000000000000,8BitDo
      M30,a:b0,b:b1,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:a4,lefttrigger:a5,leftx:a0,lefty:a1,rightshou
71 "03000000c82d00005106000000000000,8BitDo M30
      Gamepad,a:b1,b:b0,back:b10,guide:b2,leftshoulder:b6,lefttrigger:b8,leftx:a0,lefty:a1,rightshoulder:b7,righttrigger:b9,s
72 "03000000c82d00000151000000000000,8BitDo M30
      ModKit,a:b0,b:b1,back:b10,dpdown:+a2,dpleft:-a0,dpright:+a0,dpup:-a2,rightshoulder:b6,righttrigger:b7,start:b11,x:b3,y
73 "03000000c82d00000310000000000000,8BitDo
      N30,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b
74 "03000000c82d00002028000000000000,8BitDo
      N30,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrigge
75 "03000000c82d00008010000000000000,8BitDo
      N30,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b
76 "03000000c82d00000451000000000000,8BitDo N30
      Modkit,a:b1,b:b0,back:b10,dpdown:+a2,dpleft:-a0,dpright:+a0,dpup:-a2,start:b11,platform:Windows,",
77 "03000000c82d00001900000000000000,8BitDo N30
      Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrigge
78 "03000000c82d00001590000000000000,8BitDo N30 Pro
      2,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrigger
```

```
79 "03000000c82d00006528000000000000,8BitDo N30 Pro
       2,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrigger
80 "03000000022000000090000000000000,8Bitdo NES30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,left
81 "03000000203800000900000000000000,8Bitdo NES30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,left
82 "03000000c82d00000360000000000000,8BitDo Pro
       2,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,lefttrigger
83 "03000000c82d00002867000000000000,8BitDo S30
       Modkit,a:b0,b:b1,dpdown:+a2,dpleft:-a0,dpright:+a0,dpup:-a2,leftshoulder:b8,lefttrigger:b9,rightshoulder:b6,righttrigge
84 "03000000c82d00000130000000000000,8BitDo
       SF30,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrig
85 "03000000c82d00000060000000000000,8Bitdo SF30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,left
86 "03000000c82d00000061000000000000,8Bitdo SF30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrigge
87 "03000000c82d000021ab000000000000,8BitDo
       SFC30,a:b1,b:b0,back:b10,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b11,x:b4,y:b3,platform:Windows,",
88 "03000000102800000900000000000000,8Bitdo SFC30
       GamePad,a:b1,b:b0,back:b10,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b11,x:b4,y:b3,platform:Windows,",
89 "03000000c82d00003028000000000000,8Bitdo SFC30
       GamePad,a:b1,b:b0,back:b10,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b11,x:b4,y:b3,platform:Windows,",
90 "03000000c82d00000030000000000000,8BitDo
       SN30,a:b1,b:b0,back:b10,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b11,x:b4,y:b3,platform:Windows,",
91 "03000000c82d00001290000000000000,8BitDo
       SN30,a:b1,b:b0,back:b10,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b6,rightshoulder:b7,start:b11,x:b4,y:b
92 "03000000c82d000020ab000000000000,8BitDo
       SN30,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrig
93 "03000000c82d00004028000000000000,8BitDo
       SN30,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrig
94 "03000000c82d00006228000000000000,8BitDo
       SN30,a:b1,b:b0,back:b10,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b6,rightshoulder:b7,start:b11,x:b4,y:b
95 "03000000c82d00003510000000000000,8BitDo SN30
       Modkit,a:b1,b:b0,back:b10,dpdown:+a2,dpleft:-a0,dpright:+a0,dpup:-a2,leftshoulder:b6,rightshoulder:b7,start:b11,x:b4,y
96 "03000000c82d00000160000000000000,8BitDo SN30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrigge
97 "03000000c82d00000161000000000000,8BitDo SN30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrigge
98 "03000000c82d00000121000000000000,8BitDo SN30 Pro for
       Android,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,leftt
99 "03000000c82d00000260000000000000,8BitDo SN30
       Pro+,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrig
100 "03000000c82d00000261000000000000,8BitDo SN30
       Pro+,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrig
101 "03000000c82d00000031000000000000,8BitDo Wireless
       Adapter,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,
102 "03000000c82d00001890000000000000,8BitDo Zero
       2,a:b1,b:b0,back:b10,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b11,x:b4,y:b3,platform:Windows,",
103 "03000000c82d00003032000000000000,8BitDo Zero
       2,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,
104 "03000000a00500003232000000000000,8Bitdo Zero
       GamePad,a:b0,b:b1,back:b10,dpdown:+a2,dpleft:-a0,dpright:+a0,dpup:-a2,leftshoulder:b6,rightshoulder:b7,start:b11,x:b3,y
105 "03000000a30c00002700000000000000,Astro City
       Mini,a:b2,b:b1,back:b8,leftx:a3,lefty:a4,rightshoulder:b4,righttrigger:b5,start:b9,x:b3,y:b0,platform:Windows,",
106 "03000000a30c00002800000000000000,Astro City
       Mini,a:b2,b:b1,back:b8,leftx:a3,lefty:a4,rightshoulder:b4,righttrigger:b5,start:b9,x:b3,y:b0,platform:Windows,",
107 "030000008f0e00001200000000000000,Acme
       GA-02,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b5,le
108 "03000000c01100000355000011010000,ACRUX USB GAME
       PAD,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
109 "03000000fa190000f0ff000000000000,Acteck
       AGJ-3200,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,
110
       "030000006f0e00001413000000000000,Afterglow,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,
111 "03000000341a00003608000000000000,Afterglow PS3
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
112 "030000006f0e00000263000000000000,Afterglow PS3
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
113 "030000006f0e00001101000000000000,Afterglow PS3
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
114 "030000006f0e00001401000000000000,Afterglow PS3
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
115 "030000006f0e00001402000000000000,Afterglow PS3
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
116 "030000006f0e00001901000000000000,Afterglow PS3
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
117 "030000006f0e00001a01000000000000,Afterglow PS3
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
118 "03000000d62000001d57000000000000,Airflo PS3
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
119 "03000000491900001904000000000000,Amazon Luna
       Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b10,left
120 "03000000710100001904000000000000,Amazon Luna
       Controller,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b11,leftshoulder:b5,leftstick:b8,le
121
       "03000000ef050000030000000000000,AxisPad,a:b2,b:b3,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulde
122
```

```
    "03000000d6200000e557000000000000,Batarang,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,
123 "03000000c01100001352000000000000,Battalife
    Joystick,a:b6,b:b7,back:b2,leftshoulder:b0,leftx:a0,lefty:a1,rightshoulder:b1,start:b3,x:b4,y:b5,platform:Windows,",
124 "030000006f0e00003201000000000000,Battlefield 4 PS3
    Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
125 "03000000d62000002a79000000000000,BDA PS4
    Fightpad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,leftt
126 "03000000bc20000601200000000000,Betop
    2126F,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,le
127 "03000000bc20000005500000000000,Betop BFM
    Gamepad,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,leftt
128 "03000000bc20000631200000000000,Betop
    Controller,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b
129 "03000000bc20000632100000000000,BETOP
    CONTROLLER,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b
130 "03000000bc20000641200000000000,Betop
    Controller,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b
131 "03000000c01100000555000000000000,Betop
    Controller,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b
132 "03000000c01100000655000000000000,Betop
    Controller,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b
133 "03000000790000000700000000000000,Betop
    Gamepad,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,l
134 "03000000808300000300000000000000,Betop
    Gamepad,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,l
135 "030000006b1400000055000000000000,Bigben PS3
    Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
136 "030000006b1400000103000000000000,Bigben PS3
    Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
137 "03000000120c0000210e000000000000,Brook
    Mars,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
138
    "0300000066f700000500000000000000,BrutalLegendTest,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,le
139 "03000000d81d00000b00000000000000,BUFFALO BSGP1601 Series
    ,a:b5,b:b3,back:b12,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b8,leftstick:b10,lefttrigger:b6,leftx:a
140 "03000000e8200000605800000000000,Cideko
    AK08b,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,le
141
    "03000000457500000401000000000000,Cobra,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,left
142 "030000005e0400008e02000000000000,Controller (XBOX 360 For
    Windows),a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b8,lefttrigger:+a2,
143 "030000005e040000a102000000000000,Controller (Xbox 360 Wireless Receiver for
    Windows),a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b8,lefttrigger:+a2,
144 "030000005e040000ff02000000000000,Controller (Xbox One For Windows) -
    Wired,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b8,lefttrigger:+a2,le
145 "030000005e040000ea02000000000000,Controller (Xbox One For Windows) -
    Wireless,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b8,lefttrigger:+a2,
146 "03000000260900008888000000000000,Cyber Gadget GameCube
    Controller,a:b0,b:b1,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,lefttrigger:a5,leftx:a0,lefty:a1,rightshoulder:b6,r
147 "03000000a306000022f6000000000000,Cyborg V.3 Rumble
    Pad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:+a3,left
148 "03000000451300000830000000000000,Defender Game Racer
    X7,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,leftx
149 "030000007d0400000840000000000000,Destroyer
    Tiltpad,+leftx:h0.2,+lefty:h0.4,-leftx:h0.8,-lefty:h0.1,a:b1,b:b2,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulc
150 "03000000791d0000010300000000000,Dual Box
    WII,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b6,lefttrigger:b4,leftx:a0,
151 "03000000bd12000002e0000000000000,Dual USB Vibration
    Joystick,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b9,lefttrigger:b4,l
152 "030000008f0e0000091000000000000,DualShock
    2,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b9,lefttrigger:b4,leftx:a0
153 "030000006f0e00003001000000000000,EA SPORTS PS3
    Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
154 "03000000b8050000041000000000000,Elecom
    Gamepad,a:b2,b:b3,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,l
155 "03000000b8050000061000000000000,Elecom
    Gamepad,a:b2,b:b3,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,l
156
    "03000000120c0000f61c000000000000,Elite,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,left
157
    "030000008f0e00000f31000000000000,EXEQ,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4
158 "03000000341a0000010800000000000,EXEQ RF USB Gamepad
    8206,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b8,leftx:a0,lefty:a1,ri
159 "030000006f0e00008401000000000000,Faceoff Deluxe+ Audio Wired Controller for Nintendo
    Switch,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttri
160 "030000006f0e00008001000000000000,Faceoff Wired Pro Controller for Nintendo
    Switch,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttri
161
    "03000000852100000201000000000000,FF-GP1,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder
162 "030000000d0f00008500000000000000,Fighting Commander 2016
    PS3,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
163 "030000000d0f00008400000000000000,Fighting Commander
    5,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigger
164 "030000000d0f00008700000000000000,Fighting Stick mini
    4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,rightshoulc
165 "030000000d0f00008800000000000000,Fighting Stick mini
    4,a:b1,b:b2,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,rightshoulc
```

```
166 "030000000d0f00002700000000000000,FIGHTING STICK
       V3,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,rightshoul
167 "78696e7075740300000000000000000,Fightstick
       TES,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,lefttrigger:a2,rightshou
168 "03000000790000002201000000000000,Game Controller for
       PC,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,leftx
169 "0300000066f70000010000000000000,Game VIB
       Joystick,a:b2,b:b3,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b8,lefttrigger:b6,
170 "03000000260900002625000000000000,Gamecube
       Controller,a:b0,b:b1,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b6,lefttrigger:a4,leftx:a0,lefty:a1,righttrig
171 "03000000790000004618000000000000,GameCube Controller
       Adapter,a:b1,b:b2,dpdown:b14,dpleft:b15,dpright:b13,dpup:b12,lefttrigger:b4,leftx:a0,lefty:a1,rightshoulder:b7,righttri
172 "030000008f0e00000d31000000000000,GAMEPAD 3
       TURBO,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
173 "03000000280400000140000000000000,GamePad Pro
       USB,a:b1,b:b2,back:b8,leftshoulder:b4,lefttrigger:b6,leftx:a0,lefty:a1,rightshoulder:b5,righttrigger:b7,start:b9,x:b0,y
174
       "03000000ac0500003d03000000000000,GameSir,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulde
175
       "03000000ac0500004d04000000000000,GameSir,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulde
176 "03000000ffff0000000000000000000,GameStop
       Gamepad,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,l
177 "03000000c01100001400000000000000,GameStop PS4 Fun
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
178 "030000009b2800003200000000000000,GC/N64 to USB
       v3.4,a:b0,b:b7,dpdown:b11,dpleft:b12,dpright:b13,dpup:b10,lefttrigger:+a5,leftx:a0,lefty:a1,rightshoulder:b2,righttrigg
179 "030000009b2800006000000000000000,GC/N64 to USB
       v3.6,a:b0,b:b7,dpdown:b11,dpleft:b12,dpright:b13,dpup:b10,lefttrigger:+a5,leftx:a0,lefty:a1,rightshoulder:b2,righttrigg
180
       "03000000830500009a0000000000000,Genius,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder
181 "03000000830500031b0000000000000,Genius Maxfire Blaze
       3,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,leftx:a
182 "03000000451300000010000000000000,Genius Maxfire Grandias
       12,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,leftx
183 "030000005c1a00003330000000000000,Genius MaxFire Grandias
       12V,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b10,lefttrigger:b7,leftx
184 "03000000300f00000b0100000000000,GGE909 Recoil
       Pad,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,leftx
185
       "03000000f0250000c283000000000000,Gioteck,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,le
186 "03000000f025000021c1000000000000,Gioteck PS3
       Controller,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b
187 "03000000f0250000c383000000000000,Gioteck VX2
       Controller,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b
188 "03000000f0250000c483000000000000,Gioteck VX2
       Controller,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b
189 "030000007d0400000540000000000000,Gravis Eliminator GamePad
       Pro,a:b1,b:b2,back:b8,leftshoulder:b4,lefttrigger:b6,leftx:a0,lefty:a1,rightshoulder:b5,righttrigger:b7,start:b9,x:b0,y
190 "03000000341a00000302000000000000,Hama
       Scorpad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,leftt
191 "030000000d0f00004900000000000000,Hatsune Miku Sho
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
192 "03000000100800000e1000000000000,Havit
       HV-G60,a:b2,b:b1,back:b8,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b4,lefttrigger:b6,rightshoulder:b5,rig
193 "03000000d814000008620000000000,HitBox Edition
       Cthulhu+,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b5,lefttrigger:b4,righ
194
       "03000000632500002605000000000000,HJD-X,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,le
195 "030000000d0f00002d00000000000000,Hori Fighting Commander 3
       Pro,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,leftx:a0,
196 "030000000d0f00005f00000000000000,Hori Fighting Commander 4
       (PS3),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,leftx:a
197 "030000000d0f00005e00000000000000,Hori Fighting Commander 4
       (PS4),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:a3,leftx:a
198 "030000000d0f00004000000000000000,Hori Fighting Stick Mini
       3,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b5,lefttrigger:b4,rightshoulc
199 "030000000d0f00005400000000000000,Hori Pad
       3,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigger
200 "030000000d0f00009000000000000000,Hori Pad 3
       Turbo,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
201 "030000000d0f00004d00000000000000,Hori Pad
       A,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigger
202 "030000000d0f00009200000000000000,Hori Pokken Tournament DX Pro
       Pad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,lefttrigger:b6,rightshoulder:b5,ri
203 "030000000d0f00001600000000007803,HORI Real Arcade Pro EX-SE (Xbox
       360),a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,lefttrigger:a2,rightsho
204 "030000000d0f00009c00000000000000,Hori TAC
       Pro,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
205
       "030000000d0f0000c100000000000000,Horipad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,le
206 "030000000d0f00006e00000000000000,HORIPAD 4
       (PS3),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
207 "030000000d0f00006600000000000000,HORIPAD 4
       (PS4),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
208 "030000000d0f00005500000000000000,Horipad 4
       FPS,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
209 "030000000d0f0000ee00000000000000,HORIPAD
```

```
    mini4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
210 "03000000250900000017000000000000,HRAP2 on PS/SS/N64 Joypad to USB
    BOX,a:b2,b:b1,back:b9,leftshoulder:b5,lefttrigger:b4,leftx:a0,lefty:a1,rightshoulder:b7,righttrigger:b6,start:b8,x:b3,y
211 "030000008f0e00001330000000000000,HuiJia SNES
    Controller,a:b2,b:b1,back:b8,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b6,rightshoulder:b7,start:b9,x:b3,
212 "03000000d81d00000f00000000000000,iBUFFALO BSGP1204
    Series,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b10,lefttrigger:b4,le
213 "03000000d81d0000100000000000000,iBUFFALO BSGP1204P
    Series,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b10,lefttrigger:b4,le
214 "03000000830500006020000000000000,iBuffalo SNES
    Controller,a:b1,b:b0,back:b6,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b4,rightshoulder:b5,start:b7,x:b3,
215 "03000000b5070000140300000000000,Impact
    Black,a:b2,b:b3,back:b8,leftshoulder:b4,leftstick:b10,lefttrigger:b5,leftx:a0,lefty:a1,rightshoulder:b6,rightstick:b11,
216 "030000006f0e00002401000000000000,INJUSTICE FightStick PS3
    Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,ri
217
    "03000000ac0500002c02000000000000,IPEGA,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder
218 "03000000491900000204000000000000,Ipega
    PG-9023,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,
219 "03000000491900000304000000000000,Ipega PG-9087 - Bluetooth
    Gamepad,+righty:+a5,-righty:-a4,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,lefts
220 "030000006e0500000a20000000000000,JC-DUX60 ELECOM MMO
    Gamepad,a:b2,b:b3,back:b17,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b8,leftstick:b14,lefttrigger:b12
221
    "030000006e0500000520000000000000,JC-P301U,a:b2,b:b3,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulc
222 "030000006e0500000320000000000000,JC-U3613M
    (DInput),a:b2,b:b3,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b8,leftt
223
    "030000006e0500000720000000000000,JC-W01U,a:b2,b:b3,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder
224 "030000007e0500000620000000000000,Joy-Con
    (L),+leftx:h0.2,+lefty:h0.4,-leftx:h0.8,-lefty:h0.1,a:b0,b:b1,back:b13,leftshoulder:b4,leftstick:b10,rightshoulder:b5,s
225 "030000007e0500000620000001000000,Joy-Con
    (L),+leftx:h0.2,+lefty:h0.4,-leftx:h0.8,-lefty:h0.1,a:b0,b:b1,back:b13,leftshoulder:b4,leftstick:b10,rightshoulder:b5,s
226 "030000007e0500000720000000000000,Joy-Con
    (R),+leftx:h0.2,+lefty:h0.4,-leftx:h0.8,-lefty:h0.1,a:b0,b:b1,back:b12,leftshoulder:b4,leftstick:b11,rightshoulder:b5,s
227 "030000007e0500000720000001000000,Joy-Con
    (R),+leftx:h0.2,+lefty:h0.4,-leftx:h0.8,-lefty:h0.1,a:b0,b:b1,back:b12,leftshoulder:b4,leftstick:b11,rightshoulder:b5,s
228 "03000000bd12000003c0000010010000,Joypad Alpha
    Shock,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,le
229
    "03000000bd12000003c0000000000000,JY-P70UR,a:b1,b:b0,back:b5,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulde
230 "03000000242f00002d00000000000000,JYS Wireless
    Adapter,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttr
231 "03000000242f00008a00000000000000,JYS Wireless
    Adapter,a:b1,b:b4,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,leftt
232 "03000000790000000200000000000000,King PS3
    Controller,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b
233 "030000006d040000d1ca000000000000,Logitech
    ChillStream,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger
234 "030000006d040000d2ca000000000000,Logitech Cordless
    Precision,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,left
235 "030000006d04000011c2000000000000,Logitech Cordless
    Wingman,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b9,leftstick:b5,lefttrigger:b6,le
236 "030000006d04000016c2000000000000,Logitech Dual
    Action,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,le
237 "030000006d04000018c2000000000000,Logitech F510
    Gamepad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,l
238 "030000006d04000019c2000000000000,Logitech F710
    Gamepad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,l
239 "030000006d0400001ac2000000000000,Logitech Precision
    Gamepad,a:b1,b:b2,back:b8,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b4,lefttrigger:b6,rightshoulder:b5,ri
240 "030000006d0400000ac2000000000000,Logitech WingMan
    RumblePad,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,lefttrigger:b7,leftx:a0,left
241 "03000000380700006652000000000000,Mad Catz
    C.T.R.L.R,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,left
242 "03000000380700005032000000000000,Mad Catz FightPad PRO
    (PS3),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
243 "03000000380700005082000000000000,Mad Catz FightPad PRO
    (PS4),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
244 "03000000380700008433000000000000,Mad Catz FightStick TE S+
    (PS3),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
245 "03000000380700008483000000000000,Mad Catz FightStick TE S+
    (PS4),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
246 "03000000380700008134000000000000,Mad Catz FightStick TE2+
    PS3,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b7,leftstick:b10,lefttrigge
247 "03000000380700008184000000000000,Mad Catz FightStick TE2+
    PS4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b5,leftstick:b10,lefttrigge
248 "03000000380700006252000000000000,Mad Catz Micro
    C.T.R.L.R,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,left
249 "03000000380700008034000000000000,Mad Catz TE2 PS3
    Fightstick,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,le
250 "03000000380700008084000000000000,Mad Catz TE2 PS4
    Fightstick,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:a3,le
251 "03000000380700008532000000000000,Madcatz Arcade Fightstick TE S
    PS3,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,leftx:a0,
252 "03000000380700003888000000000000,Madcatz Arcade Fightstick TE S+
    PS3,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,leftx:a0,
```

```
253 "03000000380700001888000000000000,MadCatz SFIV FightStick
      PS3,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b5,lefttrigger:b7,leftx:a0,
254 "03000000380700008081000000000000,MADCATZ SFV Arcade FightStick Alpha
      PS4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,leftx:a0,
255
      "030000002a0600001024000000000000,Matricom,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,
256 "030000009f000000adbb000000000000,MaxJoypad Virtual
      Controller,a:b1,b:b2,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,lefttrigger:b6,leftx:a0,le
257 "03000000250900000128000000000000,Mayflash Arcade
      Stick,a:b1,b:b2,back:b8,leftshoulder:b0,lefttrigger:b4,leftx:a0,lefty:a1,rightshoulder:b3,righttrigger:b7,start:b9,x:b5
258 "03000000790000004418000000000000,Mayflash GameCube
      Controller,a:b1,b:b2,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,lefttrigger:a3,leftx:a0,lefty:a1,rightshoulder:b7,r
259 "03000000790000004318000000000000,Mayflash GameCube Controller
      Adapter,a:b1,b:b2,back:b0,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b0,leftshoulder:b4,leftstick:b0,lefttrig
260 "03000000242f00007300000000000000,Mayflash Magic
      NS,a:b1,b:b4,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,lefttrigge
261 "03000000790000000d218000000000000,Mayflash Magic
      NS,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigger
262 "03000000d620000010a7000000000000,Mayflash Magic
      NS,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigger
263 "030000008f0e0000103000000000000,Mayflash USB Adapter for original Sega Saturn
      controller,a:b0,b:b1,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,lefttrigger:b5,rightshoulder:b2,rig
264 "03000000250900000e803000000000000,Mayflash Wii Classic
      Controller,a:b1,b:b0,back:b8,dpdown:b13,dpleft:b12,dpright:b14,dpup:b11,guide:b10,leftshoulder:b4,lefttrigger:b6,leftx
265 "03000000790000000018000000000000,Mayflash WiiU Pro Game Controller Adapter
      (DInput),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,
266 "03000000790000002418000000000000,Mega
      Drive,a:b0,b:b1,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,rightshoulder:b2,start:b9,x:b3,y:b4,plat
267 "03000000380700006382000000000000,MLG GamePad PS3
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
268 "03000000c62400002a89000000000000,MOGA XP5-A
      Plus,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b15,leftshoulder:b6,leftstick:b13,lefttrig
269 "03000000c62400002b89000000000000,MOGA XP5-A
      Plus,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,lefttrig
270 "03000000c62400001a89000000000000,MOGA XP5-X
      Plus,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,lefttrig
271 "03000000c62400001b89000000000000,MOGA XP5-X
      Plus,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,lefttrig
272 "03000000efbe0000edfe000000000000,Monect Virtual
      Controller,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
273 "03000000250900006688000000000000,MP-8866 Super Dual
      Box,a:b2,b:b1,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b10,lefttrigger:b4,leftx
274 "030000006b140000010c000000000000,NACON
      GC-400ES,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,leftt
275 "03000000921200004b46000000000000,NES 2-port
      Adapter,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,start:b11,platform:Windows,",
276 "03000000790000004518000000000000,NEXILUX GAMECUBE Controller
      Adapter,platform:Windows,a:b1,b:b0,x:b2,y:b3,start:b9,rightshoulder:b7,dpup:h0.1,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,
277 "03000000100800000le5000000000000,NEXT SNES
      Controller,a:b2,b:b1,back:b8,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b4,rightshoulder:b5,righttrigger:b
278
      "03000000152000000182000000000000,NGDS,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4
279 "03000000bd12000015d0000000000000,Nintendo Retrolink USB Super SNES Classic
      Controller,a:b2,b:b1,back:b8,leftshoulder:b4,leftx:a0,lefty:a1,rightshoulder:b5,start:b9,x:b3,y:b0,platform:Windows,",
280 "030000007e0500009200000000000000,Nintendo Switch Pro
      Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
281 "030000000d0500000308000000000000,Nostromo
      N45,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b9,leftshoulder:b4,leftstick:b12,lefttrigger
282 "03000000550900001472000000000000,NVIDIA Controller
      v01.04,a:b11,b:b10,back:b13,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b7,leftstick:b5,leftt
283 "030000004b120000014d000000000000,NYKO
      AIRFLO,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:a3,leftstick:a0,lefttrig
284 "03000000d620000013a7000000000000,NSW wired
      controller,platform:Windows,a:b1,b:b2,x:b0,y:b3,back:b8,guide:b12,start:b9,leftstick:b10,rightstick:b11,leftshoulder:b4
285 "03000000782300000a10000000000000,Onlive Wireless
      Controller,a:b15,b:b14,back:b7,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b5,leftshoulder:b11,leftstick:b9,le
286 "03000000d62000006d57000000000000,OPP PS3
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
287 "030000006b14000001a1000000000000,Orange
      Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b6,left
288 "03000000362800000100000000000000,OUYA Game
      Controller,a:b0,b:b3,dpdown:b9,dpleft:b10,dpright:b11,dpup:b8,guide:b14,leftshoulder:b4,leftstick:b6,lefttrigger:a2,le
289 "03000000120c0000f60e000000000000,P4 Wired
      Gamepad,a:b1,b:b2,back:b12,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b5,lefttrigger:b7,right
290 "030000006f0e0000901000000000000,PDP Versus Fighting
      Pad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,rightshou
291 "030000008f0e0000300000000000000,Piranha
      xtreme,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b10,lefttrigger:b4,le
292 "030000004c050000da0c000000000000,PlayStation Classic
      Controller,a:b2,b:b1,back:b8,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b6,lefttrigger:b4,rightshoulder:b7
293 "030000004c050000371300000000000,PlayStation
      Vita,a:b1,b:b2,back:b8,dpdown:b13,dpleft:b15,dpright:b14,dpup:b12,leftshoulder:b4,leftx:a0,lefty:a1,rightshoulder:b5,ri
294 "03000000d62000006dca000000000000,PowerA Pro
      Ex,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigger
295 "03000000d62000009557000000000000,Pro Elite PS3
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
296 "03000000d62000009f31000000000000,Pro Ex mini PS3
```

```
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
297 "03000000d6200000c757000000000000,Pro Ex mini PS3
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
298 "03000000632500002306000000000000,PS
      Controller,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger
299 "03000000e30500009605000000000000,PS to USB convert
      cable,a:b2,b:b1,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b10,lefttrigger:b4,le
300 "03000000100800000100000000000000,PS1
      Controller,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b10,lefttrigger:
301 "030000008f0e00007530000000000000,PS1
      Controller,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
302 "03000000100800000300000000000000,PS2
      Controller,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b10,lefttrigger:
303 "03000000250900008888000000000000,PS2
      Controller,a:b2,b:b1,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b10,lefttrigger:
304 "03000000666600006706000000000000,PS2
      Controller,a:b2,b:b1,back:b8,dpdown:b14,dpleft:b15,dpright:b13,dpup:b12,leftshoulder:b6,leftstick:b9,lefttrigger:b4,le
305 "030000006b1400000303000000000000,PS2
      Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
306 "030000009d0d00001330000000000000,PS2
      Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:
307 "03000000250900000500000000000000,PS3
      Controller,a:b2,b:b1,back:b9,dpdown:h0.8,dpleft:h0.4,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b10,lefttrigger:
308 "030000004c0500006802000000000000,PS3
      Controller,a:b2,b:b1,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b10,le
309 "03000000632500007505000000000000,PS3
      Controller,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
310 "03000000888800000803000000000000,PS3
      Controller,a:b2,b:b1,back:b8,dpdown:h0.8,dpleft:h0.4,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b9,left
311 "030000008f0e00001431000000000000,PS3
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
312 "03000000380700056a8000000000000,PS3 RF
      pad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
313 "03000000100000008200000000000000,PS360+
      v1.66,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,leftx:h
314 "030000004c050000a00b000000000000,PS4
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
315 "030000004c050000c405000000000000,PS4
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
316 "030000004c050000cc09000000000000,PS4
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
317 "030000004c050000e60c000000000000,PS5
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
318
      "03000000ff000000cb01000000000000,PSP,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,
319 "03000000300f0000001100000000000,QanBa Arcade JoyStick
      1008,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,lefttrigger:b6,leftx:a0,lefty:a1,
320 "03000000300f0000161100000000000,QanBa Arcade JoyStick
      4018,a:b1,b:b2,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b9,leftshoulder:b4,lefttrigger:b6,rightsho
321 "03000000222c0000002000000000000,QANBA DRONE ARCADE
      JOYSTICK,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:a3,righ
322 "03000000300f0000121000000000000,QanBa Joystick
      Plus,a:b0,b:b1,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,rightshoulder:b5,start:b9,x:b2,y:b3,plat
323 "03000000341a0000104000000000000,QanBa Joystick
      Q4RAF,a:b5,b:b6,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b0,lefttrigger:b4,leftx:a
324 "03000000222c0000022300000000000,Qanba Obsidian Arcade Joystick PS3
      Mode,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
325 "03000000222c0000002300000000000,Qanba Obsidian Arcade Joystick PS4
      Mode,a:b1,b:b2,back:b13,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
326 "03000000321500000003000000000000,Razer
      Hydra,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b8,leftx:a0,lefty:a1,r
327 "03000000321500000204000000000000,Razer Panthera
      (PS3),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
328 "03000000321500000104000000000000,Razer Panthera
      (PS4),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
329 "03000000321500000507000000000000,Razer Raiju
      Mobile,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,l
330 "03000000321500000707000000000000,Razer Raiju
      Mobile,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,l
331 "03000000321500000011000000000000,Razer Raion Fightpad for
      PS4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
332 "03000000321500000009000000000000,Razer
      Serval,+lefty:+a2,-lefty:-a1,a:b0,b:b1,back:b12,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:h
333 "030000000d0f00001100000000000000,REAL ARCADE
      PRO.3,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
334 "030000000d0f00006a00000000000000,Real Arcade
      Pro.4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
335 "030000000d0f00006b00000000000000,Real Arcade
      Pro.4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
336 "030000000d0f00008a00000000000000,Real Arcade
      Pro.4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
337 "030000000d0f00008b00000000000000,Real Arcade
      Pro.4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
338 "030000000d0f00007000000000000000,REAL ARCADE PRO.4
      VLX,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
339 "030000000d0f00002200000000000000,REAL ARCADE
      Pro.V3,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,leftx
```

```
340 "030000000d0f00005b00000000000000,Real Arcade
      Pro.V4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttr
341 "030000000d0f00005c00000000000000,Real Arcade
      Pro.V4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttr
342 "03000000790000001100000000000000,Retrolink SNES
      Controller,a:b2,b:b1,back:b8,dpdown:+a4,dpleft:-a3,dpright:+a3,dpup:-a4,leftshoulder:b4,rightshoulder:b5,start:b9,x:b3,
343 "03000000bd12000013d0000000000000,Retrolink USB SEGA Saturn
      Classic,a:b0,b:b1,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b5,lefttrigger:b6,rightshoulder:b2,righttrigg
344 "0300000000f000000300000000000000,RetroUSB.com
      RetroPad,a:b1,b:b5,back:b2,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b3,x:b0,y:b4,platform:Windows,",
345 "0300000000f00000f100000000000000,RetroUSB.com Super
      RetroPort,a:b1,b:b5,back:b2,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b3,x:b0,y:b4,platform:Windows,",
346 "030000006b140000010d000000000000,Revolution Pro
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
347 "030000006b140000020d000000000000,Revolution Pro Controller
      2(1/2),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttr
348 "030000006b140000130d000000000000,Revolution Pro Controller
      3,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigger
349 "030000006f0e00001e01000000000000,Rock Candy PS3
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
350 "030000006f0e00002801000000000000,Rock Candy PS3
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
351 "030000006f0e00002f01000000000000,Rock Candy PS3
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
352
      "030000004f04000003d0000000000000,run'n'drive,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b
353 "03000000a30600001af5000000000000,Saitek
      Cyborg,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,le
354 "03000000a306000023f6000000000000,Saitek Cyborg V.1 Game
      pad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
355 "03000000300f0000120100000000000,Saitek Dual Analog
      Pad,a:b2,b:b3,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b5,leftx
356 "03000000a3060000701000000000000,Saitek
      P220,a:b2,b:b3,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b6,lefttrigger:b7,rightshoulder:b4,righttrigger
357 "03000000a30600000cff000000000000,Saitek P2500 Force Rumble
      Pad,a:b2,b:b3,back:b11,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b8,lefttrigger:b6,leftx
358 "03000000a30600000c04000000000000,Saitek
      P2900,a:b1,b:b2,back:b12,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b10,lefttrig
359 "03000000300f0000100100000000000,Saitek P480 Rumble
      Pad,a:b2,b:b3,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b5,leftx
360 "03000000a30600000b04000000000000,Saitek
      P990,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigg
361 "03000000a30600000b04000000010000,Saitek P990 Dual Analog
      Pad,a:b1,b:b2,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,leftx
362 "03000000a3060000210600000000000,Saitek
      PS1000,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttr
363 "03000000a306000020f6000000000000,Saitek
      PS2700,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttr
364 "03000000300f0000110100000000000,Saitek Rumble
      Pad,a:b2,b:b3,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b10,lefttrigger:b5,left
365 "03000000730700000401000000000000,Sanwa PlayOnline
      Mobile,a:b0,b:b1,back:b2,leftx:a0,lefty:a1,start:b3,platform:Windows,",
366
      "03000000000050000289b000000000000,Saturn_Adapter_2.0,a:b1,b:b2,leftshoulder:b6,lefttrigger:b4,leftx:a0,lefty:a1,rightsh
367
      "030000009b2800000500000000000000,Saturn_Adapter_2.0,a:b1,b:b2,leftshoulder:b6,lefttrigger:b4,leftx:a0,lefty:a1,rightsh
368 "03000000a30c0000250000000000000,Sega Genesis Mini 3B
      controller,a:b2,b:b1,dpdown:+a4,dpleft:-a3,dpright:+a3,dpup:-a4,righttrigger:b5,start:b9,platform:Windows,",
369 "03000000a30c0000240000000000000,Sega Mega Drive Mini 6B
      controller,a:b2,b:b1,dpdown:+a4,dpleft:-a3,dpright:+a3,dpup:-a4,rightshoulder:b4,righttrigger:b5,start:b9,x:b3,y:b0,pla
370
      "03000000341a0000020800000000000,SL-6555-SBK,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshou
371
      "03000000341a0000090800000000000,SL-6566,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder
372 "030000008f0e00000800000000000000,SpeedLink Strike
      FX,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,leftx
373 "03000000c01100000591000000000000,Speedlink
      Torid,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,le
374 "03000000d1180000009400000000000,Stadia
      Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b6,left
375
      "03000000110100001914000000000000,SteelSeries,a:b0,b:b1,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftstick:b13,le
376 "03000000381000001214000000000000,SteelSeries
      Free,a:b0,b:b1,back:b12,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder
377 "03000000110100003114000000000000,SteelSeries Stratus
      Duo,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,left
378 "03000000381000001814000000000000,SteelSeries Stratus
      XL,a:b0,b:b1,back:b18,dpdown:b13,dpleft:b14,dpright:b15,dpup:b12,guide:b19,leftshoulder:b4,leftstick:b10,lefttrigger:a1
379
      "03000000790000001c18000000000000,STK-7024X,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshou
380 "03000000ff1100003133000000000000,SVEN
      X-PAD,a:b2,b:b3,back:b4,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,lefttrigger:b8,leftx:a0,lefty:a1
381
      "03000000d620000011a7000000000000,Switch,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,le
382 "03000000457500002211000000000000,SZMY-POWER PC
      Gamepad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttr
383 "030000004f04000007d0000000000000,T Mini
```

```
       Wireless,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,leftt
384
       "030000004f0400000ab1000000000000,T.16000M,a:b0,b:b1,back:b12,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b11,
385 "03000000fa1900000706000000000000,Team
       5,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,leftx:a
386 "03000000b5070000120300000000000,Techmobility
       X6-38V,a:b2,b:b3,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b5,le
387 "030000004f04000015b3000000000000,Thrustmaster Dual Analog
       4,a:b0,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b5,leftx:a
388 "030000004f04000023b3000000000000,Thrustmaster Dual Trigger
       3-in-1,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,le
389 "030000004f0400000ed0000000000000,ThrustMaster eSwap PRO
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
390 "030000004f04000000b3000000000000,Thrustmaster Firestorm Dual
       Power,a:b0,b:b2,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b11,lefttrigg
391 "030000004f04000004b3000000000000,Thrustmaster Firestorm Dual Power
       3,a:b0,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b5,leftx:a
392 "030000006666000004880000000000000,TigerGame PS/PS2 Game Controller
       Adapter,a:b2,b:b1,back:b9,dpdown:b14,dpleft:b15,dpright:b13,dpup:b12,leftshoulder:b6,leftstick:b10,lefttrigger:b4,leftx
393 "03000000d62000006000000000000000,Tournament PS3
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
394 "030000005f140000c501000000000000,Trust
       Gamepad,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttr
395 "03000000b8050000021000000000000,Trust
       Gamepad,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttr
396 "030000004f04000087b6000000000000,TWCS
       Throttle,dpdown:b8,dpleft:b9,dpright:b7,dpup:b6,leftstick:b5,lefttrigger:-a5,leftx:a0,lefty:a1,righttrigger:+a5,platfor
397 "03000000d9040000020000000000000,TwinShock
       PS2,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b10,lefttrigger:b4,leftx
398
       "030000006e0500001320000000000000,U4113,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,left
399 "03000000101c0000171c000000000000,uRage
       Gamepad,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,l
400 "03000000300f0000070100000000000,USB 4-Axis 12-Button
       Gamepad,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,l
401 "03000000341a0000230800000000000,USB
       gamepad,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttr
402 "03000000550900000b4000000000000,USB
       gamepad,a:b10,b:b11,back:b5,dpdown:b1,dpleft:b2,dpright:b3,dpup:b0,guide:b14,leftshoulder:b8,leftstick:b6,lefttrigger:a
403 "030000006b1400000203000000000000,USB
       gamepad,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttr
404 "03000000790000000a00000000000000,USB
       gamepad,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,l
405 "03000000f0250000c183000000000000,USB
       gamepad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttr
406 "03000000ff1100004133000000000000,USB
       gamepad,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b10,lefttrigger:b4,l
407 "03000000632500002305000000000000,USB Vibration Joystick
       (BM),a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,left
408
       "03000000790000001a18000000000000,Venom,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,left
409 "03000000790000001b18000000000000,Venom Arcade
       Joystick,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,righ
410 "030000006f0e0000302000000000000,Victrix Pro Fight Stick for
       PS4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,rightshou
411 "030000006f0e0000702000000000000,Victrix Pro Fight Stick for
       PS4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
412 "03000000341200000adbe000000000000,vJoy
       Device,a:b0,b:b1,back:b15,dpdown:b6,dpleft:b7,dpright:b8,dpup:b5,guide:b16,leftshoulder:b9,leftstick:b13,lefttrigger:b1
413 "030000005e0400000a0b000000000000,Xbox Adaptive
       Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b8,lefttrigger:+a
414 "030000005e040000130b000000000000,Xbox Series
       Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b8,left
415
       "03000000341a0000060800000000000,Xeox,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,lefts
416 "03000000450c0000204300000000000,XEOX Gamepad
       SL-6556-BK,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b
417 "03000000ac0500005b05000000000000,Xiaoji
       Gamesir-G3w,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
418 "030000001727000044310000000000,XiaoMi Game
       Controller,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b20,leftshoulder:b6,leftstick:b13,le
419 "03000000786901066e70000000000000,XInput
       Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b8,left
420 "03000000790000004f18000000000000,ZD-T
       Android,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,leftt
421 "03000000120c0000101e000000000000,ZEROPLUS P4 Wired
       Gamepad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttr
422 "78696e7075740100000000000000000,XInput Gamepad
       (GLFW),platform:Windows,a:b0,b:b1,x:b2,y:b3,leftshoulder:b4,rightshoulder:b5,back:b6,start:b7,leftstick:b8,rightstick:b
423 "78696e7075740200000000000000000,XInput Wheel
       (GLFW),platform:Windows,a:b0,b:b1,x:b2,y:b3,leftshoulder:b4,rightshoulder:b5,back:b6,start:b7,leftstick:b8,rightstick:b
424 "78696e7075740300000000000000000,XInput Arcade Stick
       (GLFW),platform:Windows,a:b0,b:b1,x:b2,y:b3,leftshoulder:b4,rightshoulder:b5,back:b6,start:b7,leftstick:b8,rightstick:b
425 "78696e7075740400000000000000000,XInput Flight Stick
       (GLFW),platform:Windows,a:b0,b:b1,x:b2,y:b3,leftshoulder:b4,rightshoulder:b5,back:b6,start:b7,leftstick:b8,rightstick:b
426 "78696e7075740500000000000000000,XInput Dance Pad
       (GLFW),platform:Windows,a:b0,b:b1,x:b2,y:b3,leftshoulder:b4,rightshoulder:b5,back:b6,start:b7,leftstick:b8,rightstick:b
```

```
427 "78696e7075740600000000000000000,XInput Guitar
      (GLFW),platform:Windows,a:b0,b:b1,x:b2,y:b3,leftshoulder:b4,rightshoulder:b5,back:b6,start:b7,leftstick:b8,rightstick:b
428 "78696e7075740800000000000000000,XInput Drum Kit
      (GLFW),platform:Windows,a:b0,b:b1,x:b2,y:b3,leftshoulder:b4,rightshoulder:b5,back:b6,start:b7,leftstick:b8,rightstick:b
429 #endif // GLFW_BUILD_WIN32_MAPPINGS
430
431 #if defined(GLFW_BUILD_COCOA_MAPPINGS)
432 "030000008f0e00000300000009010000,2In1 USB
      Joystick,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,
      OS X,",
433 "03000000c82d00000090000001000000,8BitDo FC30
      Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:a4,left
      OS X,",
434 "03000000c82d00001038000000010000,8BitDo FC30
      Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:a5,left
      OS X,",
435 "03000000c82d00000650000001000000,8BitDo
      M30,a:b0,b:b1,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b8,lefttrigger:b9,leftx:a0,lefty:a1,rightshou
      OS X,",
436 "03000000c82d00005106000000010000,8BitDo M30
      Gamepad,a:b1,b:b0,back:b10,guide:b2,leftshoulder:b6,lefttrigger:a5,leftx:a0,lefty:a1,rightshoulder:b7,righttrigger:a4,s
      OS X,",
437 "03000000c82d00001590000001000000,8BitDo N30 Pro
      2,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrigger
      OS X,",
438 "03000000c82d00006528000000010000,8BitDo N30 Pro
      2,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrigger
      OS X,",
439 "030000003512000012ab000001000000,8BitDo NES30
      Gamepad,a:b1,b:b0,back:b10,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b6,rightshoulder:b7,start:b11,x:b4,y
      OS X,",
440 "03000000022000000090000001000000,8Bitdo NES30
      Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,left
      OS X,",
441 "03000000203800000900000000010000,8Bitdo NES30
      Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,left
      OS X,",
442 "03000000c82d00000190000001000000,8Bitdo NES30
      Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,left
      OS X,",
443 "03000000102800000900000000000000,8Bitdo SFC30 GamePad
      Joystick,a:b1,b:b0,back:b10,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b11,x:b4,y:b3,platform:Mac
      OS X,",
444 "03000000c82d00001290000001000000,8BitDo SN30
      Gamepad,a:b1,b:b0,back:b10,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b11,x:b4,y:b3,platform:Mac
      OS X,",
445 "03000000c82d00004028000000010000,8Bitdo SN30
      GamePad,a:b1,b:b0,x:b4,y:b3,back:b10,start:b11,leftshoulder:b6,rightshoulder:b7,dpup:-a1,dpdown:+a1,dpleft:-a0,dpright
      OS X,",
446 "03000000c82d00000160000001000000,8BitDo SN30
      Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:a4,left
      OS X,",
447 "03000000c82d00000161000000010000,8BitDo SN30
      Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrigge
      OS X,",
448 "03000000c82d00000260000001000000,8BitDo SN30
      Pro+,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrigg
      OS X,",
449 "03000000c82d00000261000000010000,8BitDo SN30
      Pro+,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrigg
      OS X,",
450 "03000000c82d00000031000001000000,8BitDo Wireless
      Adapter,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,
      OS X,",
451 "03000000c82d00001890000001000000,8BitDo Zero
      2,a:b1,b:b0,back:b10,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b11,x:b4,y:b3,platform:Mac
      OS X,",
452 "03000000c82d00003032000000010000,8BitDo Zero
      2,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,
      OS X,",
453 "03000000a00500003232000008010000,8Bitdo Zero
      GamePad,a:b0,b:b1,back:b10,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b6,rightshoulder:b7,start:b11,x:b3,y
      OS X,",
454 "03000000a00500003232000009010000,8Bitdo Zero
      GamePad,a:b0,b:b1,back:b10,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b6,rightshoulder:b7,start:b11,x:b3,y
      OS X,",
455 "03000000a30c00002700000003030000,Astro City
      Mini,a:b2,b:b1,back:b8,leftx:a3,lefty:a4,rightshoulder:b4,righttrigger:b5,start:b9,x:b3,y:b0,platform:Mac
      OS X,",
456 "03000000a30c00002800000003030000,Astro City
      Mini,a:b2,b:b1,back:b8,leftx:a3,lefty:a4,rightshoulder:b4,righttrigger:b5,start:b9,x:b3,y:b0,platform:Mac
      OS X,",
457 "03000000050b00000045000031000000,ASUS
      Gamepad,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b7,lefttrigger:a5,
      OS X,",
458
      "03000000ef05000003000000000020000,AxisPad,a:b2,b:b3,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulde
```

```
       OS X,",
459 "03000000491900001904000001010000,Amazon Luna
       Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b10,left
       OS X,",
460 "03000000710100001904000000010000,Amazon Luna
       Controller,a:b0,b:b1,back:b11,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b7,le
       OS X,",
461 "03000000c62400001a89000000010000,BDA MOGA XP5-X
       Plus,a:b0,b:b1,back:b12,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b14,leftshoulder:b6,leftstick:b15,lefttrig
       OS X,",
462 "03000000c62400001b89000000010000,BDA MOGA XP5-X
       Plus,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,lefttrig
       OS X,",
463 "03000000d62000002a79000000010000,BDA PS4
       Fightpad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,leftt
       OS X,",
464 "03000000120c0000200e000000010000,Brook
       Mars,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigg
       OS X,",
465 "03000000120c0000210e000000010000,Brook
       Mars,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigg
       OS X,",
466 "03000000830500003 1b0000000000000,Cideko
       AK08b,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,le
       OS X,",
467 "03000000260900008888000088020000,Cyber Gadget GameCube
       Controller,a:b0,b:b1,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,lefttrigger:a4,leftx:a0,lefty:a1,rightshoulder:b6,
       OS X,",
468 "03000000a306000022f6000001030000,Cyborg V.3 Rumble
       Pad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
       OS X,",
469 "03000000790000004618000000010000,GameCube Controller
       Adapter,a:b4,b:b0,dpdown:b56,dpleft:b60,dpright:b52,dpup:b48,lefttrigger:a12,leftx:a0,lefty:a4,rightshoulder:b28,rightt
       OS X,",
470 "03000000ad1b000001f9000000000000,Gamestop BB-070 X360
       Controller,a:b0,b:b1,back:b9,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,guide:b10,leftshoulder:b4,leftstick:b6,lefttrig
       OS X,",
471 "05000000475320476 16d657061640000,GameStop
       Gamepad,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,
       OS X,",
472 "03000000c01100000140000000010000,GameStop PS4 Fun
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
       OS X,",
473 "030000006f0e00000102000000000000,GameStop Xbox 360 Wired
       Controller,a:b0,b:b1,back:b9,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,guide:b10,leftshoulder:b4,leftstick:b6,lefttrig
       OS X,",
474 "030000007d0400000540000001010000,Gravis Eliminator GamePad
       Pro,a:b1,b:b2,back:b8,leftshoulder:b4,lefttrigger:b6,leftx:a0,lefty:a1,rightshoulder:b5,righttrigger:b7,start:b9,x:b0,y
       OS X,",
475 "03000000280400000140000000020000,Gravis Gamepad
       Pro,a:b1,b:b2,back:b8,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b4,lefttrigger:b6,rightshoulder:b5,righttt
       OS X,",
476 "030000008f0e00000300000007010000,GreenAsia Inc. USB
       Joystick,a:b2,b:b3,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,lefttrigger:b5,leftx:a0,lefty
       OS X,",
477 "030000000d0f00002d00000000100000,Hori Fighting Commander 3
       Pro,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,leftx:a0,
       OS X,",
478 "030000000d0f00005f00000000010000,Hori Fighting Commander 4
       (PS3),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,leftx:a
       OS X,",
479 "030000000d0f00005e00000000010000,Hori Fighting Commander 4
       (PS4),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:a3,leftx:a
       OS X,",
480 "030000000d0f00005f00000000000000,HORI Fighting Commander 4
       PS3,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
       OS X,",
481 "030000000d0f00005e00000000000000,HORI Fighting Commander 4
       PS4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
       OS X,",
482 "030000000d0f00004d00000000000000,HORI Gem Pad
       3,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigger
       OS X,",
483 "030000000d0f00009200000000010000,Hori Pokken Tournament DX Pro
       Pad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,lefttrigger:b6,rightshoulder:b5,r
       OS X,",
484 "030000000d0f00006e00000000010000,HORIPAD 4
       (PS3),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
       OS X,",
485 "030000000d0f00006600000000010000,HORIPAD 4
       (PS4),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
       OS X,",
486 "030000000d0f00006600000000000000,HORIPAD FPS PLUS
       4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigger
       OS X,",
487 "030000000d0f0000ee00000000010000,HORIPAD
       mini4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
```

```
      OS X,",
488 "030000008f0e00001330000011010000,HuiJia SNES
      Controller,a:b4,b:b2,back:b16,dpdown:+a2,dpleft:-a0,dpright:+a0,dpup:-a2,leftshoulder:b12,rightshoulder:b14,start:b18,x
      OS X,",
489 "030000008305000060200000000010000,iBuffalo SNES
      Controller,a:b1,b:b0,back:b6,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b4,rightshoulder:b5,start:b7,x:b3,
      OS X,",
490 "030000008305000060200000000000000,iBuffalo USB 2-axis 8-button
      Gamepad,a:b1,b:b0,back:b6,leftshoulder:b4,leftx:a0,lefty:a1,rightshoulder:b5,start:b7,x:b3,y:b2,platform:Mac
      OS X,",
491 "030000007e0500000620000001000000,Joy-Con
      (L),+leftx:h0.2,+lefty:h0.4,-leftx:h0.8,-lefty:h0.1,a:b0,b:b1,back:b13,leftshoulder:b4,leftstick:b10,rightshoulder:b5,s
      OS X,",
492 "030000007e0500000720000001000000,Joy-Con
      (R),+leftx:h0.2,+lefty:h0.4,-leftx:h0.8,-lefty:h0.1,a:b0,b:b1,back:b12,leftshoulder:b4,leftstick:b11,rightshoulder:b5,s
      OS X,",
493 "03000000242f00002d00000007010000,JYS Wireless
      Adapter,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttr
      OS X,",
494 "030000006d04000016c2000000020000,Logitech Dual
      Action,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,le
      OS X,",
495 "030000006d04000016c2000000030000,Logitech Dual
      Action,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,le
      OS X,",
496 "030000006d04000016c2000014040000,Logitech Dual
      Action,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,le
      OS X,",
497 "030000006d04000016c2000000000000,Logitech F310 Gamepad
      (DInput),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,
      OS X,",
498 "030000006d04000018c2000000000000,Logitech F510 Gamepad
      (DInput),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,
      OS X,",
499 "030000006d04000019c2000005030000,Logitech
      F710,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,left
      OS X,",
500 "030000006d0400001fc2000000000000,Logitech F710 Gamepad
      (XInput),a:b0,b:b1,back:b9,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,guide:b10,leftshoulder:b4,leftstick:b6,lefttrigge
      OS X,",
501 "030000006d04000018c2000000010000,Logitech RumblePad 2
      USB,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,leftx
      OS X,",
502 "030000006d04000019c2000000000000,Logitech Wireless Gamepad
      (DInput),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,
      OS X,",
503 "030000003807000050320000000010000,Mad Catz FightPad PRO
      (PS3),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
      OS X,",
504 "030000003807000050820000000010000,Mad Catz FightPad PRO
      (PS4),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
      OS X,",
505 "030000003807000084330000000010000,Mad Catz FightStick TE S+
      (PS3),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
      OS X,",
506 "030000003807000084830000000010000,Mad Catz FightStick TE S+
      (PS4),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
      OS X,",
507 "030000007900000006000000007010000,Marvo
      GT-004,a:b2,b:b1,x:b3,y:b0,back:b8,start:b9,leftstick:b10,rightstick:b11,leftshoulder:b4,rightshoulder:b5,dpup:h0.1,dpd
      OS X,",
508 "030000007900000044180000000010000,Mayflash GameCube
      Controller,a:b1,b:b2,dpdown:b14,dpleft:b15,dpright:b13,dpup:b12,lefttrigger:a3,leftx:a0,lefty:a1,rightshoulder:b7,right
      OS X,",
509 "03000000242f00007300000000020000,Mayflash Magic
      NS,a:b1,b:b4,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,lefttrigge
      OS X,",
510 "03000000790000000d218000026010000,Mayflash Magic
      NS,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigger
      OS X,",
511 "03000000d620000010a7000003010000,Mayflash Magic
      NS,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigger
      OS X,",
512 "03000000250900000e803000000000000,Mayflash Wii Classic
      Controller,a:b1,b:b0,back:b8,dpdown:b13,dpleft:b12,dpright:b14,dpup:b11,guide:b10,leftshoulder:b4,lefttrigger:b6,leftx
      OS X,",
513 "030000007900000000180000000010000,Mayflash Wii U Pro Controller
      Adapter,a:b4,b:b8,back:b32,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b16,leftstick:b40,lefttrigger:b2
      OS X,",
514 "030000007900000000180000000000000,Mayflash WiiU Pro Game Controller Adapter
      (DInput),a:b4,b:b8,back:b32,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b16,leftstick:b40,lefttrigger:b
      OS X,",
515 "03000000d8140000cecf000000000000,MC
      Cthulhu,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,lefttrigger:b6,rightshoulder:b
      OS X,",
516 "030000005e0400002700000001010000,Microsoft SideWinder Plug & Play Game
      Pad,a:b0,b:b1,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,lefttrigger:b4,leftx:a0,lefty:a1,righttrigger:b5,x:b2,y:b3,pla
```

```
      OS X,",
517 "03000000d6200000716200001000000,Moga Pro 2
      HID,a:b0,b:b1,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b7,lefttrigger:a5,leftx
      OS X,",
518 "03000000c62400002a89000000010000,MOGA XP5-A
      Plus,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b21,leftshoulder:b6,leftstick:b13,lefttrig
      OS X,",
519 "03000000c62400002b89000000010000,MOGA XP5-A
      Plus,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,lefttrig
      OS X,",
520 "03000000632500007505000000020000,NEOGEO mini PAD
      Controller,a:b1,b:b0,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,start:b9,x:b2,y:b3,platform:Mac
      OS X,",
521 "03000000921200004b46000003020000,NES 2-port
      Adapter,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,start:b11,platform:Mac OS
      X,",
522 "03000000100800001e5000006010000,NEXT SNES
      Controller,a:b2,b:b1,back:b8,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b4,rightshoulder:b5,righttrigger:b
      OS X,",
523 "03000000d620000011a7000000020000,Nintendo Switch Core (Plus) Wired
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
      OS X,",
524 "03000000d620000011a7000010050000,Nintendo Switch PowerA Wired
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
      OS X,",
525 "030000007e0500009200000000000000,Nintendo Switch Pro
      Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
      OS X,",
526 "030000007e0500009200000001000000,Nintendo Switch Pro
      Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
      OS X,",
527 "03000000550900001472000025050000,NVIDIA Controller
      v01.04,a:b0,b:b1,back:b17,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b15,leftshoulder:b4,leftstick:b7,lefttr
      OS X,",
528 "030000006f0e00000901000002010000,PDP Versus Fighting
      Pad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,rightshou
      OS X,",
529 "030000008f0e00000300000000000000,Piranha
      xtreme,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b10,lefttrigger:b4,le
      OS X,",
530 "030000004c050000da0c000000010000,Playstation Classic
      Controller,a:b2,b:b1,back:b8,leftshoulder:b6,lefttrigger:b4,leftx:a0,lefty:a1,rightshoulder:b7,righttrigger:b5,start:b9
      OS X,",
531 "030000004c050000371300000010000,PlayStation
      Vita,a:b1,b:b2,back:b8,dpdown:b13,dpleft:b15,dpright:b14,dpup:b12,leftshoulder:b4,leftx:a0,lefty:a1,rightshoulder:b5,ri
      OS X,",
532 "03000000d62000006dca000000010000,PowerA Pro
      Ex,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigger
      OS X,",
533 "03000000100800000300000006010000,PS2
      Adapter,a:b2,b:b1,back:b8,leftshoulder:b6,leftstick:b10,lefttrigger:b4,leftx:a0,lefty:a1,rightshoulder:b7,rightstick:b1
      OS X,",
534 "030000004c050000680200000000000,PS3
      Controller,a:b14,b:b13,back:b0,dpdown:b6,dpleft:b7,dpright:b5,dpup:b4,guide:b16,leftshoulder:b10,leftstick:b1,lefttrigg
      OS X,",
535 "030000004c0500006802000000010000,PS3
      Controller,a:b14,b:b13,back:b0,dpdown:b6,dpleft:b7,dpright:b5,dpup:b4,guide:b16,leftshoulder:b10,leftstick:b1,lefttrigg
      OS X,",
536 "030000004c050000a00b000000010000,PS4
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
      OS X,",
537 "030000004c050000c405000000000000,PS4
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
      OS X,",
538 "030000004c050000c405000000010000,PS4
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
      OS X,",
539 "030000004c050000cc09000000010000,PS4
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
      OS X,",
540 "050000004c050000e60c000000010000,PS5
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
      OS X,",
541 "03000000891600000fd000000000000,Razer Onza
      TE,a:b0,b:b1,back:b9,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,guide:b10,leftshoulder:b4,leftstick:b6,lefttrigger:a2,l
      OS X,",
542 "03000000321500000204000000010000,Razer Panthera
       (PS3),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
      OS X,",
543 "03000000321500000104000000010000,Razer Panthera
       (PS4),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
      OS X,",
544 "03000000321500000010000000010000,Razer
      RAIJU,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
      OS X,",
545 "03000000321500000507000001010000,Razer Raiju
      Mobile,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b21,leftshoulder:b6,leftstick:b13,leftt
```

```
      OS X,",
546 "03000000321500000011000000010000,Razer Raion Fightpad for
      PS4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
      OS X,",
547 "03000000321500000009000000020000,Razer
      Serval,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigc
      OS X,",
548 "03000000321500000090000163a0000,Razer
      Serval,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigc
      OS X,",
549 "0300000032150000030a000000000000,Razer
      Wildcat,a:b0,b:b1,back:b9,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,guide:b10,leftshoulder:b4,leftstick:b6,lefttrigger
      OS X,",
550 "03000000790000001100000000000000,Retrolink Classic
      Controller,a:b2,b:b1,back:b8,leftshoulder:b4,leftx:a3,lefty:a4,rightshoulder:b5,start:b9,x:b3,y:b0,platform:Mac
      OS X,",
551 "03000000790000001100000006010000,Retrolink SNES
      Controller,a:b2,b:b1,back:b8,dpdown:+a4,dpleft:-a3,dpright:+a3,dpup:-a4,leftshoulder:b4,rightshoulder:b5,start:b9,x:b3,
      OS X,",
552 "030000006b140000010d000000010000,Revolution Pro
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
      OS X,",
553 "030000006b140000130d000000010000,Revolution Pro Controller
      3,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigger
      OS X,",
554 "03000000c6240000fefa000000000000,Rock Candy Gamepad for
      PS3,a:b0,b:b1,back:b9,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,guide:b10,leftshoulder:b4,leftstick:b6,lefttrigger:a2,
      OS X,",
555 "03000000730700000401000000010000,Sanwa PlayOnline
      Mobile,a:b0,b:b1,back:b2,leftx:a0,lefty:a1,start:b3,platform:Mac OS X,",
556 "03000000811700007e05000000000000,Sega
      Saturn,a:b2,b:b4,dpdown:b16,dpleft:b15,dpright:b14,dpup:b17,leftshoulder:b8,lefttrigger:a5,leftx:a0,lefty:a2,rightshoul
      OS X,",
557 "03000000b40400000a01000000000000,Sega Saturn USB
      Gamepad,a:b0,b:b1,back:b5,guide:b2,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b8,x:b3,y:b4,platform:Mac
      OS X,",
558 "03000000351200002lab000000000000,SFC30
      Joystick,a:b1,b:b0,back:b10,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b11,x:b4,y:b3,platform:Mac
      OS X,",
559 "0300000000f00000f100000000000000,SNES
      RetroPort,a:b2,b:b3,back:b4,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b5,rightshoulder:b7,start:b6,x:b0,y
      OS X,",
560 "030000004c050000e60c000000010000,Sony
      DualSense,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,left
      OS X,",
561 "030000004c050000cc09000000000000,Sony DualShock 4
      V2,a:b1,b:b2,back:b13,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
      OS X,",
562 "030000004c050000a00b000000000000,Sony DualShock 4 Wireless
      Adaptor,a:b1,b:b2,back:b13,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,leftt
      OS X,",
563 "03000000d11800000094000000010000,Stadia
      Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
      OS X,",
564 "030000005e0400008e02000001000000,Steam Virtual
      Gamepad,a:b0,b:b1,back:b9,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,leftshoulder:b4,leftstick:b6,lefttrigger:a2,leftx
      OS X,",
565 "03000000110100002014000000000000,SteelSeries
      Nimbus,a:b0,b:b1,dpdown:b9,dpleft:b11,dpright:b10,dpup:b8,leftshoulder:b4,lefttrigger:b6,leftx:a0,lefty:a1,rightshoulde
      OS X,",
566 "03000000110100002014000001000000,SteelSeries
      Nimbus,a:b0,b:b1,dpdown:b9,dpleft:b11,dpright:b10,dpup:b8,guide:b12,leftshoulder:b4,lefttrigger:b6,leftx:a0,lefty:a1~,r
      OS X,",
567 "03000000381000002014000001000000,SteelSeries
      Nimbus,a:b0,b:b1,dpdown:b9,dpleft:b11,dpright:b10,dpup:b8,guide:b12,leftshoulder:b4,lefttrigger:b6,leftx:a0,lefty:a1~,r
      OS X,",
568 "050000004e696d6275732b0000000000,SteelSeries Nimbus
      Plus,a:b0,b:b1,back:b15,dpdown:b11,dpleft:b13,dpright:b12,dpup:b10,guide:b16,leftshoulder:b4,leftstick:b8,lefttrigger:b
      OS X,",
569 "03000000110100001714000000000000,SteelSeries Stratus
      XL,a:b0,b:b1,dpdown:b9,dpleft:b11,dpright:b10,dpup:b8,leftshoulder:b4,lefttrigger:b6,leftx:a0,lefty:a1~,rightshoulder:b
      OS X,",
570 "03000000110100001714000020010000,SteelSeries Stratus
      XL,a:b0,b:b1,dpdown:b9,dpleft:b11,dpright:b10,dpup:b8,leftshoulder:b4,lefttrigger:b6,leftx:a0,lefty:a1~,rightshoulder:b
      OS X,",
571 "03000000457500002211000000010000,SZMY-POWER PC
      Gamepad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,leftt
      OS X,",
572 "030000004f04000015b3000000000000,Thrustmaster Dual Analog
      3.2,a:b0,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b5,left
      OS X,",
573 "030000004f0400000ed0000000020000,ThrustMaster eSwap PRO
      Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
      OS X,",
574 "030000004f04000000b3000000000000,Thrustmaster Firestorm Dual
      Power,a:b0,b:b2,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b11,lefttrigc
      OS X,",
```

```
575 "03000000bd12000015d0000000000000,Tomee SNES USB
       Controller,a:b2,b:b1,back:b8,leftshoulder:b4,leftx:a0,lefty:a1,rightshoulder:b5,start:b9,x:b3,y:b0,platform:Mac
       OS X,",
576 "03000000bd12000015d0000000010000,Tomee SNES USB
       Controller,a:b2,b:b1,back:b8,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b4,rightshoulder:b5,start:b9,x:b3,
       OS X,",
577 "03000000100800000100000000000000,Twin USB
       Joystick,a:b4,b:b2,back:b16,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b12,leftstick:b20,lefttrigger:b
       OS X,",
578 "030000006f0e00000302000025040000,Victrix Pro Fight Stick for
       PS4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,rightshou
       OS X,",
579 "030000006f0e00000702000003060000,Victrix Pro Fight Stick for
       PS4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
       OS X,",
580 "03000000791d00000103000009010000,Wii Classic
       Controller,a:b2,b:b1,back:b8,dpdown:b14,dpleft:b15,dpright:b13,dpup:b12,guide:b10,leftshoulder:b6,lefttrigger:b4,leftx
       OS X,",
581 "050000005769696d6f74652028303000,Wii
       Remote,a:b4,b:b5,back:b7,dpdown:b3,dpleft:b0,dpright:b1,dpup:b2,guide:b8,leftshoulder:b11,lefttrigger:b12,leftx:a0,left
       OS X,",
582 "050000005769696d6f74652028313800,Wii U Pro
       Controller,a:b16,b:b15,back:b7,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,guide:b8,leftshoulder:b19,leftstick:b23,leftt
       OS X,",
583 "030000005e0400008e02000000000000,X360
       Controller,a:b0,b:b1,back:b9,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,guide:b10,leftshoulder:b4,leftstick:b6,lefttrig
       OS X,",
584 "030000006f0e00000104000000000000,Xbox 360 Wired
       Controller,a:b0,b:b1,back:b9,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,guide:b10,leftshoulder:b4,leftstick:b6,lefttrig
       OS X,",
585 "03000000c6240000045d000000000000,Xbox 360 Wired
       Controller,a:b0,b:b1,back:b9,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,guide:b10,leftshoulder:b4,leftstick:b6,lefttrig
       OS X,",
586 "030000005e0400000a0b000000000000,Xbox Adaptive
       Controller,a:b0,b:b1,back:b9,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,guide:b10,leftshoulder:b4,leftstick:b6,lefttrig
       OS X,",
587 "030000005e040000050b000003090000,Xbox Elite Wireless Controller Series
       2,a:b0,b:b1,back:b31,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b53,leftshoulder:b6,leftstick:b13,lefttrigger
       OS X,",
588 "03000000c62400003a54000000000000,Xbox One PowerA Wired
       Controller,a:b0,b:b1,back:b9,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,guide:b10,leftshoulder:b4,leftstick:b6,lefttrig
       OS X,",
589 "030000005e040000d102000000000000,Xbox One Wired
       Controller,a:b0,b:b1,back:b9,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,guide:b10,leftshoulder:b4,leftstick:b6,lefttrig
       OS X,",
590 "030000005e040000dd02000000000000,Xbox One Wired
       Controller,a:b0,b:b1,back:b9,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,guide:b10,leftshoulder:b4,leftstick:b6,lefttrig
       OS X,",
591 "030000005e040000e302000000000000,Xbox One Wired
       Controller,a:b0,b:b1,back:b9,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,guide:b10,leftshoulder:b4,leftstick:b6,lefttrig
       OS X,",
592 "030000005e040000130b000001050000,Xbox Series
       Controller,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,le
       OS X,",
593 "030000005e040000130b000005050000,Xbox Series
       Controller,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,le
       OS X,",
594 "030000005e040000e002000000000000,Xbox Wireless
       Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b8,left
       OS X,",
595 "030000005e040000e002000003090000,Xbox Wireless
       Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b8,left
       OS X,",
596 "030000005e040000ea02000000000000,Xbox Wireless
       Controller,a:b0,b:b1,back:b9,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,guide:b10,leftshoulder:b4,leftstick:b6,lefttrig
       OS X,",
597 "030000005e040000fd02000003090000,Xbox Wireless
       Controller,a:b0,b:b1,back:b16,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b15,leftshoulder:b6,leftstick:b13,le
       OS X,",
598 "030000001727000044310000290100000,XiaoMi Game
       Controller,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b15,leftshoulder:b6,leftstick:b13,le
       OS X,",
599 "03000000120c0000100e000000010000,ZEROPLUS P4
       Gamepad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,leftt
       OS X,",
600 "03000000120c0000101e000000010000,ZEROPLUS P4 Wired
       Gamepad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,leftt
       OS X,",
601 #endif // GLFW_BUILD_COCOA_MAPPINGS
602
603 #if defined(GLFW_BUILD_LINUX_MAPPINGS)
604 "03000000c82d00000090000011010000,8BitDo FC30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:a4,left
605 "05000000c82d00001038000000010000,8Bitdo FC30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,left
606 "05000000c82d00005106000000010000,8BitDo
       M30,a:b0,b:b1,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b8,lefttrigger:b9,leftx:a0,lefty:a1,rightshou
```

```
607 "03000000c82d00001590000011010000,8BitDo N30 Pro
       2,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,leftx
608 "05000000c82d00006528000000010000,8BitDo N30 Pro
       2,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,leftx
609 "03000000c82d00000310000011010000,8BitDo
       NES30,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b7,lefttrigger:b6,rightshoulder:b
610 "05000000c82d00008010000000010000,8BitDo
       NES30,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b7,lefttrigger:b6,rightshoulder:b
611 "03000000022000000090000011010000,8Bitdo NES30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,left
612 "05000000203800000900000000010000,8Bitdo NES30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,left
613 "05000000c82d00002038000000010000,8Bitdo NES30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrigge
614 "03000000c82d00000190000011010000,8Bitdo NES30 Pro 8Bitdo NES30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:a4,left
615 "05000000c82d00000060000000010000,8BitDo SF30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,left
616 "05000000c82d00000061000000010000,8Bitdo SF30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrigge
617 "03000000c82d000021ab000010010000,8BitDo
       SFC30,a:b1,b:b0,back:b10,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b11,x:b4,y:b3,platform:Linux,",
618 "030000003512000012ab000010010000,8Bitdo SFC30
       GamePad,a:b2,b:b1,back:b6,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b4,rightshoulder:b5,start:b7,x:b3,y:b
619 "05000000102800000900000000010000,8Bitdo SFC30
       GamePad,a:b1,b:b0,back:b10,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b11,x:b4,y:b3,platform:Linux,",
620 "05000000c82d00003028000000010000,8Bitdo SFC30
       GamePad,a:b1,b:b0,back:b10,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b11,x:b4,y:b3,platform:Linux,",
621 "03000000c82d00000160000000000000,8BitDo SN30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,left
622 "03000000c82d00000160000011010000,8BitDo SN30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,left
623 "03000000c82d00000161000000000000,8BitDo SN30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,left
624 "03000000c82d00001290000011010000,8BitDo SN30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,left
625 "05000000c82d00000161000000010000,8BitDo SN30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,lefttrigge
626 "05000000c82d00006228000000010000,8BitDo SN30
       Pro,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:b8,left
627 "03000000c82d00000260000011010000,8BitDo SN30
       Pro+,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,lefttric
628 "05000000c82d00000261000000010000,8BitDo SN30
       Pro+,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,lefttric
629 "05000000202800000900000000010000,8BitDo SNES30
       Gamepad,a:b1,b:b0,back:b10,dpdown:b122,dpleft:b119,dpright:b120,dpup:b117,leftshoulder:b6,rightshoulder:b7,start:b11,x
630 "03000000c82d00000031000011010000,8BitDo Wireless Adapter
       (DInput),a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b2,leftshoulder:b6,leftstick:b13,leftt
631 "030000005e0400008e02000020010000,8BitDo Wireless Adapter
       (XInput),a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttri
632 "03000000c82d00001890000011010000,8BitDo Zero
       2,a:b1,b:b0,back:b10,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b11,x:b4,y:b3,platform:Linux,",
633 "05000000c82d00003032000000010000,8BitDo Zero
       2,a:b1,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,
634 "050000005e040000e002000030110000,8BitDo Zero 2
       (XInput),a:b0,b:b1,back:b6,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b4,rightshoulder:b5,start:b7,x:b2,y
635 "05000000a00500003232000001000000,8Bitdo Zero
       GamePad,a:b0,b:b1,back:b10,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b11,x:b3,y:b4,platform:Linux,",
636 "05000000a00500003232000008010000,8Bitdo Zero
       GamePad,a:b0,b:b1,back:b10,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b6,rightshoulder:b7,start:b11,x:b3,y
637 "03000000c01100000355000011010000,ACRUX USB GAME
       PAD,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
638
       "030000006f0e00001302000000010000,Afterglow,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,l
639 "030000006f0e00003901000020060000,Afterglow Controller for Xbox
       One,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger
640 "030000006f0e00003901000000430000,Afterglow Prismatic
       Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
641 "030000006f0e00003901000013020000,Afterglow Prismatic Wired Controller
       048-007-NA,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
642 "03000000100000008200000011010000,Akishop Customs PS360+
       v1.66,a:b1,b:b2,back:b12,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,lefttrigger:b6,rightsh
643 "030000007c1800000006000010010000,Alienware Dual Compatible Game
       Pad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,leftx
644 "05000000491900000204000021000000,Amazon Fire Game
       Controller,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b17,leftshoulder:b6,leftstick:b13,le
645 "03000000491900001904000011010000,Amazon Luna
       Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b10,left
646 "05000000710100001904000000010000,Amazon Luna
       Controller,a:b0,b:b1,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b7,left
647 "03000000790000003018000011010000,Arcade Fightstick
       F300,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,leftx:a
648 "03000000a30c00002700000011010000,Astro City
       Mini,a:b2,b:b1,back:b8,leftx:a0,lefty:a1,rightshoulder:b4,righttrigger:b5,start:b9,x:b3,y:b0,platform:Linux,",
649 "03000000a30c00002800000011010000,Astro City
       Mini,a:b2,b:b1,back:b8,leftx:a0,lefty:a1,rightshoulder:b4,righttrigger:b5,start:b9,x:b3,y:b0,platform:Linux,",
650 "05000000050b00000045000031000000,ASUS
```

```
    Gamepad,a:b0,b:b1,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b6,leftshoulder:b4,leftstick:b7,lefttric
651 "05000000050b0000004500004000000,ASUS
    Gamepad,a:b0,b:b1,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b6,leftshoulder:b4,leftstick:b7,lefttric
652 "03000000503200000110000000000000,Atari Classic
    Controller,a:b0,back:b2,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b4,start:b3,x:b1,platform:Linux,",
653 "05000000503200000110000000000000,Atari Classic
    Controller,a:b0,back:b2,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b4,start:b3,x:b1,platform:Linux,",
654 "03000000503200000210000000000000,Atari Game
    Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
655 "05000000503200000210000000000000,Atari Game
    Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
656
    "03000000120c0000050000010010000,AxisPad,a:b2,b:b3,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulde
657
    "03000000ef0500000300000000010000,AxisPad,a:b2,b:b3,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulde
658 "03000000c62400001b89000011010000,BDA MOGA XP5-X
    Plus,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,lefttric
659 "03000000d62000002a79000011010000,BDA PS4
    Fightpad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,leftt
660 "03000000c21100000791000011010000,Be1 GC101 Controller 1.03
    mode,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,left
661 "03000000c31100000791000011010000,Be1 GC101 GAMEPAD 1.03
    mode,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,lefttric
662 "030000005e0400008e02000003030000,Be1 GC101 Xbox 360 Controller
    mode,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger
663 "05000000bc2000000055000001000000,BETOP AX1
    BFM,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,lefttrigg
664 "03000000666600006706000000010000,boom PSX to PC
    Converter,a:b2,b:b1,back:b8,dpdown:b14,dpleft:b15,dpright:b13,dpup:b12,leftshoulder:b6,leftstick:b9,lefttrigger:b4,left
665 "03000000120c0000200e000011010000,Brook
    Mars,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigg
666 "03000000120c0000210e000011010000,Brook
    Mars,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigg
667 "03000000120c0000f70e000011010000,Brook Universal Fighting
    Board,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
668 "03000000ffff0000ffff000000010000,Chinese-made Xbox
    Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b5,leftstick:b8,lefttrigger:a2
669 "03000000e82000006058000001010000,Cideko
    AK08b,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,le
670 "030000000b0400003365000000010000,Competition
    Pro,a:b0,b:b1,back:b2,leftx:a0,lefty:a1,start:b3,platform:Linux,",
671 "03000000260900008888000000010000,Cyber Gadget GameCube
    Controller,a:b0,b:b1,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,lefttrigger:a4,leftx:a0,lefty:a1,rightshoulder:b6,r
672 "03000000a306000022f6000011010000,Cyborg V.3 Rumble
    Pad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
673 "03000000b404000000a01000000010000,CYPRESS USB
    Gamepad,a:b0,b:b1,back:b5,guide:b2,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b8,x:b3,y:b4,platform:Linux
674 "03000000790000000600000010010000,DragonRise Inc. Generic USB
    Joystick,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,
675 "030000004f04000004b3000010010000,Dual Power
    2,a:b0,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b5,leftx:a
676 "030000006f0e0000300100001010000,EA Sports PS3
    Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
677 "03000000341a000005f7000010010000,GameCube {HuiJia USB
    box},a:b1,b:b2,dpdown:b14,dpleft:b15,dpright:b13,dpup:b12,lefttrigger:a3,leftx:a0,lefty:a1,rightshoulder:b7,righttrigge
678 "03000000bc2000000055000011010000,GameSir
    G3w,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:a5,left
679 "050000004753204761 6d657061640000,GameStop
    Gamepad,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,
680 "030000006f0e0000104000000010000,Gamestop Logic3
    Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
681 "030000008f0e0000080000010010000,Gasia Co. Ltd PS(R)
    Gamepad,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,
682 "030000006f0e0000130400000010000,Generic X-Box
    pad,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger
683 "03000000451300000010000010010000,Genius Maxfire Grandias
    12,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,leftx
684 "03000000f0250000c183000010010000,Goodbetterbest Ltd USB
    Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
685 "03000000790000000d418000000010000,GPD Win 2
    Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
686 "030000007d0400000540000000010000,Gravis Eliminator GamePad
    Pro,a:b1,b:b2,back:b8,leftshoulder:b4,lefttrigger:b6,leftx:a0,lefty:a1,rightshoulder:b5,righttrigger:b7,start:b9,x:b0,y
687 "03000000280400000140000000010000,Gravis GamePad Pro USB
    ,a:b1,b:b2,back:b8,leftshoulder:b4,lefttrigger:b6,leftx:a0,lefty:a1,rightshoulder:b5,righttrigger:b7,start:b9,x:b0,y:b1
688 "030000008f0e0000061000000010000,GreenAsia Electronics 4Axes 12Keys GamePad
    ,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b9,lefttrigger:b4,leftx:a0,
689 "030000008f0e0000120000010010000,GreenAsia Inc. USB
    Joystick,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b5,
690 "050000004753206761 6d657061640000,GS
    gamepad,a:b0,b:b1,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b10,lefttrigger:b6,
691 "03000000f0250000c383000010010000,GT
    VX2,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,leftx
692 "06000000adde0000efbe000002010000,Hidromancer Game
    Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
693 "03000000d8140000086200011010000,HitBox (PS3/PC) Analog
    Mode,a:b1,b:b2,back:b8,guide:b9,leftshoulder:b4,lefttrigger:b6,leftx:a0,lefty:a1,rightshoulder:b5,righttrigger:b7,start
```

```
694 "03000000c9110000f055000011010000,HJC Game
      GAMEPAD,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,
695
      "03000000632500002605000010010000,HJD-X,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,le
696
      "030000000d0f00000d00000000010000,hori,a:b0,b:b6,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,lefts
697 "030000000d0f00001000000011010000,HORI CO. LTD. FIGHTING STICK
      3,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,rightshould
698 "030000000d0f0000c100000011010000,HORI CO. LTD. HORIPAD
      S,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b13,leftshoulder:b4,leftstick:b10,lefttrigger
699 "030000000d0f00006a00000011010000,HORI CO. LTD. Real Arcade
      Pro.4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
700 "030000000d0f00006b00000011010000,HORI CO. LTD. Real Arcade
      Pro.4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
701 "030000000d0f00002200000011010000,HORI CO. LTD. REAL ARCADE
      Pro.V3,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,rights
702 "030000000d0f00008500000010010000,HORI Fighting
      Commander,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,left
703 "030000000d0f00008600000002010000,Hori Fighting
      Commander,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b11,left
704 "030000000d0f00005f00000011010000,Hori Fighting Commander 4
       (PS3),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,leftx:a
705 "030000000d0f00005e00000011010000,Hori Fighting Commander 4
       (PS4),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:a3,leftx:a
706 "03000000ad1b000001f5000033050000,Hori Pad EX Turbo
      2,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger:a2
707 "030000000d0f00009200000011010000,Hori Pokken Tournament DX Pro
      Pad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,lefttrigger:b6,rightshoulder:b5,ri
708 "030000000d0f0000aa00000011010000,HORI Real Arcade
      Pro,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
709 "030000000d0f0000d800000072056800,HORI Real Arcade Pro
      S,a:b0,b:b1,back:b4,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,guide:b5,leftshoulder:b9,leftstick:b7,lefttrigger:a4,le
710 "030000000d0f00001600000000010000,Hori Real Arcade Pro.EX-SE (Xbox
      360),a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,lefttrigger:b6,rightsho
711 "030000000d0f00006e00000011010000,HORIPAD 4
       (PS3),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
712 "030000000d0f00006600000011010000,HORIPAD 4
       (PS4),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
713 "030000000d0f0000ee00000011010000,HORIPAD
      mini4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
714 "030000000d0f00006700000001010000,HORIPAD
      ONE,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger
715 "030000008f0e00001330000010010000,HuiJia SNES
      Controller,a:b2,b:b1,back:b8,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b6,rightshoulder:b7,start:b9,x:b3,
716 "03000000242e00008816000001010000,Hyperkin
      X91,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger
717 "030000008305000060200000010010000,iBuffalo SNES
      Controller,a:b1,b:b0,back:b6,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b4,rightshoulder:b5,start:b7,x:b3,
718
      "050000006964726f69643a636f6e0000,idroid:con,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshou
719
      "03000000b5070000150300001010000,impact,a:b2,b:b3,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder
720 "03000000d8040000820000003000000,IMS PCU#0 Gamepad
      Interface,a:b1,b:b0,back:b4,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,start:b5,x:b3,y:b2,platform:Linux,",
721 "03000000fd0500000030000000010000,InterAct GoPad I-73000 (Fighting Game
      Layout),a:b3,b:b4,back:b6,leftx:a0,lefty:a1,rightshoulder:b2,righttrigger:b5,start:b7,x:b0,y:b1,platform:Linux,",
722 "05000000491900000204000001b010000,Ipega PG-9069 - Bluetooth
      Gamepad,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b161,leftshoulder:b6,leftstick:b13,left
723 "03000000632500007505000011010000,Ipega PG-9099 - Bluetooth
      Gamepad,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,
724 "030000006e0500000320000010010000,JC-U3613M - DirectInput
      Mode,a:b2,b:b3,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b8,lefttrigg
725 "03000000300f00001001000010010000,Jess Tech Dual Analog Rumble
      Pad,a:b2,b:b3,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b5,leftx
726 "03000000300f00000b01000010010000,Jess Tech GGE909 PC Recoil
      Pad,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,leftx
727 "03000000ba2200002010000001010000,Jess Technology USB Game
      Controller,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,lefttrigger:b6,leftx:a0,le
728 "030000007e0500006200000001000000,Joy-Con
       (L),+leftx:h0.2,+lefty:h0.4,-leftx:h0.8,-lefty:h0.1,a:b0,b:b1,back:b13,leftshoulder:b4,leftstick:b10,rightshoulder:b5,s
729 "050000007e0500006200000001000000,Joy-Con
       (L),+leftx:h0.2,+lefty:h0.4,-leftx:h0.8,-lefty:h0.1,a:b0,b:b1,back:b13,leftshoulder:b4,leftstick:b10,rightshoulder:b5,s
730 "030000007e0500007200000001000000,Joy-Con
       (R),+leftx:h0.2,+lefty:h0.4,-leftx:h0.8,-lefty:h0.1,a:b0,b:b1,back:b12,leftshoulder:b4,leftstick:b11,rightshoulder:b5,s
731 "050000007e0500007200000001000000,Joy-Con
       (R),+leftx:h0.2,+lefty:h0.4,-leftx:h0.8,-lefty:h0.1,a:b0,b:b1,back:b12,leftshoulder:b4,leftstick:b11,rightshoulder:b5,s
732 "03000000bd12000003c0000010010000,Joypad Alpha
      Shock,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,le
733 "03000000242f00002d00000011010000,JYS Wireless
      Adapter,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,leftt
734 "03000000242f00008a00000011010000,JYS Wireless
      Adapter,a:b1,b:b4,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,leftt
735 "030000006f0e00000103000000020000,Logic3
      Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
736 "030000006d040000d1ca000000000000,Logitech
      ChillStream,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger
737 "030000006d04000019c2000010010000,Logitech Cordless RumblePad
```

```
      2,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,leftx:a
738 "030000006d04000016c2000010010000,Logitech Dual
      Action,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,le
739 "030000006d04000016c2000011010000,Logitech Dual
      Action,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,le
740 "030000006d0400001dc2000014400000,Logitech F310 Gamepad
      (XInput),a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttri
741 "030000006d0400001ec2000019200000,Logitech F510 Gamepad
      (XInput),a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttri
742 "030000006d0400001ec2000020200000,Logitech F510 Gamepad
      (XInput),a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttri
743 "030000006d04000019c2000011010000,Logitech F710 Gamepad
      (DInput),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,
744 "030000006d0400001fc2000005030000,Logitech F710 Gamepad
      (XInput),a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttri
745 "030000006d0400000ac2000010010000,Logitech Inc. WingMan
      RumblePad,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,lefttrigger:b7,leftx:a0,left
746 "030000006d04000018c2000010010000,Logitech RumblePad
      2,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,leftx:a
747 "030000006d04000011c2000010010000,Logitech WingMan Cordless
      RumblePad,a:b0,b:b1,back:b2,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b5,leftshoulder:b6,lefttrigger:b9,left
748
      "050000004d4f435554452d3035305800,M54-PC,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder
749 "0500000038070000665200025010000,Mad Catz C.T.R.L.R
      ,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigger:b
750 "0300000038070000503200011010000,Mad Catz FightPad PRO
      (PS3),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
751 "0300000038070000508200011010000,Mad Catz FightPad PRO
      (PS4),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
752 "03000000ad1b00002ef0000090040000,Mad Catz Fightpad
      SFxT,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,lefttrigger:a2,rightshou
753 "0300000038070000803400011010000,Mad Catz fightstick
      (PS3),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,leftx:a
754 "0300000038070000808400011010000,Mad Catz fightstick
      (PS4),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:a3,leftx:a
755 "0300000038070000843300011010000,Mad Catz FightStick TE S+
      (PS3),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
756 "0300000038070000848300011010000,Mad Catz FightStick TE S+
      (PS4),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
757 "0300000038070000164700010040000,Mad Catz Wired Xbox 360
      Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
758 "0300000038070000384700090040000,Mad Catz Wired Xbox 360 Controller
      (SFIV),a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,lefttrigger:b6,leftx
759 "03000000ad1b000016f0000090040000,Mad Catz Xbox 360
      Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
760 "0300000038070000188800010010000,MadCatz PC USB Wired Stick
      8818,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,leftx:a0
761 "0300000038070000388800010010000,MadCatz PC USB Wired Stick
      8838,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:a0,lefttrigge
762 "03000000242f0000f700000001010000,Magic-S
      Pro,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger
763 "03000000120c0000050000000010000,Manta Dualshock
      2,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,leftx:a
764 "0300000079000000441800010010000,Mayflash GameCube
      Controller,a:b1,b:b0,dpdown:b14,dpleft:b15,dpright:b13,dpup:b12,lefttrigger:a3,leftx:a0,lefty:a1,rightshoulder:b7,right
765 "0300000079000000431800010010000,Mayflash GameCube Controller
      Adapter,a:b1,b:b2,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,lefttrigger:a3,leftx:a0,lefty:a1,rightshoulder:b7,righ
766 "03000000242f00007300000011010000,Mayflash Magic
      NS,a:b1,b:b4,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,lefttrigge
767 "0300000079000000d218000011010000,Mayflash Magic
      NS,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigger
768 "03000000d620000010a7000011010000,Mayflash Magic
      NS,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,leftx
769 "0300000025090000e803000001010000,Mayflash Wii Classic
      Controller,a:b1,b:b0,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:a4,lefttrigger:b6,le
770 "0300000078000000600000010010000,Microntek USB
      Joystick,a:b2,b:b1,back:b8,leftshoulder:b6,lefttrigger:b4,leftx:a0,lefty:a1,rightshoulder:b7,righttrigger:b5,start:b9,x
771 "030000005e0400000e00000000010000,Microsoft
      SideWinder,a:b0,b:b1,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,rightshoulder:b7,start:b8,x
772 "030000005e0400008e02000004010000,Microsoft X-Box 360
      pad,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger
773 "030000005e0400008e02000062230000,Microsoft X-Box 360
      pad,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger
774 "050000005e040000050b000003090000,Microsoft X-Box One Elite 2
      pad,a:b0,b:b1,back:b17,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,lefttrigg
775 "030000005e040000e302000003020000,Microsoft X-Box One Elite
      pad,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger
776 "030000005e040000d102000001010000,Microsoft X-Box One
      pad,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger
777 "030000005e040000dd02000003020000,Microsoft X-Box One pad (Firmware
      2015),a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigge
778 "030000005e040000d102000003020000,Microsoft X-Box One pad
      v2,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger:a
779 "030000005e0400008502000000010000,Microsoft X-Box pad
      (Japan),a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b5,leftstick:b8,lefttrigger:a2,le
780 "030000005e0400008902000021010000,Microsoft X-Box pad v2
      (US),a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b5,leftstick:b8,lefttrigger:a2,left
```

```
781 "030000005e040000000b000008040000,Microsoft Xbox One Elite 2 pad -
       Wired,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigge
782 "030000005e040000ea02000008040000,Microsoft Xbox One S pad -
       Wired,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigge
783 "03000000c62400001a53000000010000,Mini
       PE,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger:a
784
       "03000000030000000300000002000000,Miroof,a:b1,b:b0,back:b6,leftshoulder:b4,leftx:a0,lefty:a1,rightshoulder:b5,start:b7,
785 "05000000d6200000e589000001000000,Moga 2
       HID,a:b0,b:b1,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b7,lefttrigger:a5,leftx
786 "05000000d6200000ad0d000001000000,Moga
       Pro,a:b0,b:b1,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b7,lefttrigger:a5,leftx:a0,lefty
787 "05000000d62000007162000001000000,Moga Pro 2
       HID,a:b0,b:b1,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b7,lefttrigger:a5,leftx
788 "03000000c62400002b89000011010000,MOGA XP5-A
       Plus,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,lefttrig
789 "05000000c62400002a89000000010000,MOGA XP5-A
       Plus,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b22,leftshoulder:b6,leftstick:b13,lefttrig
790 "05000000c62400001a89000000010000,MOGA XP5-X
       Plus,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,lefttrig
791 "03000000250900006688000000010000,MP-8866 Super Dual
       Box,a:b2,b:b1,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b10,lefttrigger:b4,leftx
792 "030000006b140000010c000010010000,NACON
       GC-400ES,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,leftt
793 "030000000d0f00000900000010010000,Natec Genesis
       P44,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
794 "03000000790000004518000010010000,NEXILUX GAMECUBE Controller
       Adapter,a:b1,b:b0,x:b2,y:b3,start:b9,rightshoulder:b7,dpup:h0.1,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,leftx:a0,lefty:a1,
795 "03000001008000001e5000010010000,NEXT SNES
       Controller,a:b2,b:b1,back:b8,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b4,rightshoulder:b5,righttrigger:b
796 "060000007e0500003713000000000000,Nintendo
       3DS,a:b0,b:b1,back:b8,dpdown:b11,dpleft:b12,dpright:b13,dpup:b10,leftshoulder:b4,leftx:a0,lefty:a1,rightshoulder:b5,rig
797 "060000007e0500000820000000000000,Nintendo Combined Joy-Cons
       (joycond),a:b0,b:b1,back:b9,dpdown:b15,dpleft:b16,dpright:b17,dpup:b14,guide:b11,leftshoulder:b5,leftstick:b12,lefttrig
798 "030000007e0500003703000000016800,Nintendo GameCube
       Controller,a:b0,b:b2,dpdown:b6,dpleft:b4,dpright:b5,dpup:b7,lefttrigger:a4,leftx:a0,lefty:a1~,rightshoulder:b9,righttri
799 "03000000790000004618000010010000,Nintendo GameCube Controller
       Adapter,a:b1,b:b0,dpdown:b14,dpleft:b15,dpright:b13,dpup:b12,lefttrigger:b4,leftx:a0,lefty:a1,rightshoulder:b7,righttri
800 "050000007e0500006200000001800000,Nintendo Switch Left
       Joy-Con,a:b9,b:b8,back:b5,leftshoulder:b2,leftstick:b6,leftx:a1,lefty:a0~,rightshoulder:b4,start:b0,x:b7,y:b10,platform
801 "030000007e0500009200000011810000,Nintendo Switch Pro
       Controller,a:b0,b:b1,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b11,leftshoulder:b5,leftstick:b12,le
802 "050000007e0500009200000001000000,Nintendo Switch Pro
       Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
803 "050000007e0500009200000001800000,Nintendo Switch Pro
       Controller,a:b0,b:b1,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b11,leftshoulder:b5,leftstick:b12,le
804 "050000007e0500007200000001800000,Nintendo Switch Right
       Joy-Con,a:b1,b:b2,back:b9,leftshoulder:b4,leftstick:b10,leftx:a1~,lefty:a0~,rightshoulder:b6,start:b8,x:b0,y:b3,platfor
805 "050000007e0500001720000001000000,Nintendo Switch SNES
       Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,rightshoulder:b5,start:b9,x
806 "050000007e0500003003000001000000,Nintendo Wii Remote Pro
       Controller,a:b0,b:b1,back:b8,dpdown:b14,dpleft:b15,dpright:b16,dpup:b13,guide:b10,leftshoulder:b4,leftstick:b11,lefttri
807 "050000001000000010000000030000000,Nintendo
       Wiimote,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b11,leftti
808 "030000000d0500003080000010010000,Nostromo n45 Dual Analog
       Gamepad,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b9,leftshoulder:b4,leftstick:b12,lefttri
809 "03000000550900001072000011010000,NVIDIA
       Controller,a:b0,b:b1,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b13,leftshoulder:b4,leftstick:b8,lefttrigger
810 "03000000550900001472000011010000,NVIDIA Controller
       v01.04,a:b0,b:b1,back:b14,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b16,leftshoulder:b4,leftstick:b7,lefttri
811 "05000000550900001472000001000000,NVIDIA Controller
       v01.04,a:b0,b:b1,back:b14,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b16,leftshoulder:b4,leftstick:b7,lefttri
812 "03000000451300000830000010010000,NYKO
       CORE,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigg
813
       "19000000010000000100000001010000,odroidgo2_joypad,a:b1,b:b0,dpdown:b7,dpleft:b8,dpright:b9,dpup:b6,guide:b10,leftshou
814
       "19000000010000000200000011000000,odroidgo2_joypad_v11,a:b1,b:b0,dpdown:b9,dpleft:b10,dpright:b11,dpup:b8,guide:b12,le
815 "030000005e0400000202000000010000,Old Xbox
       pad,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b5,leftstick:b8,lefttrigger:a2,leftx
816 "03000000c0160000dc27000001010000,OnyxSoft Dual
       JoyDivision,a:b0,b:b1,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b4,rightshoulder:b5,start:b6,x:b2,y:b3,pl
817 "05000000362800000100000002010000,OUYA Game
       Controller,a:b0,b:b3,dpdown:b9,dpleft:b10,dpright:b11,dpup:b8,guide:b14,leftshoulder:b4,leftstick:b6,lefttrigger:a2,le
818 "05000000362800000100000003010000,OUYA Game
       Controller,a:b0,b:b3,dpdown:b9,dpleft:b10,dpright:b11,dpup:b8,guide:b14,leftshoulder:b4,leftstick:b6,lefttrigger:a2,le
819 "03000000830500005020000010010000,Padix Co. Ltd. Rockfire PSX/USB
       Bridge,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b8,lefttrigger:b6,le
820 "03000000790000001c18000011010000,PC Game
       Controller,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b
821 "03000000ff1100003133000010010000,PC Game
       Controller,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b
822 "030000006f0e0000b802000001010000,PDP AFTERGLOW Wired Xbox One
       Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
823 "030000006f0e0000b802000013020000,PDP AFTERGLOW Wired Xbox One
       Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
824 "030000006f0e00006401000001010000,PDP Battlefield
```

```
       One,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger
825 "030000006f0e0000800100001101000,PDP CO. LTD. Faceoff Wired Pro Controller for Nintendo
       Switch,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttri
826 "030000006f0e0000310100000010000,PDP EA Sports
       Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
827 "030000006f0e0000c802000012010000,PDP Kingdom Hearts
       Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
828 "030000006f0e0000870100001101000,PDP Rock Candy Wired Controller for Nintendo
       Switch,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b13,leftshoulder:b4,leftstick:b10,lefttri
829 "030000006f0e0000090100001101000,PDP Versus Fighting
       Pad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,rightshou
830 "030000006f0e0000a802000023020000,PDP Wired Controller for Xbox
       One,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b11,lefttrigge
831 "030000006f0e0000850100001101000,PDP Wired Fight Pad Pro for Nintendo
       Switch,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttri
832
       "05000000491900000304000001b010000,PG-9099,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulde
833
       "05000000491900000204000000000000,PG-9118,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulde
834 "030000004c050000da0c000011010000,Playstation
       Controller,a:b2,b:b1,back:b8,leftshoulder:b6,lefttrigger:b4,leftx:a0,lefty:a1,rightshoulder:b7,righttrigger:b5,start:b9
835 "030000004c0500003713000011010000,PlayStation
       Vita,a:b1,b:b2,back:b8,dpdown:b13,dpleft:b15,dpright:b14,dpup:b12,leftshoulder:b4,leftx:a0,lefty:a1,rightshoulder:b5,ri
836
       "03000000c624000000530000000010000,PowerA,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,left
837 "03000000c62400003a54000001010000,PowerA
       1428124-01,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
838 "03000000d62000006dca000011010000,PowerA Pro
       Ex,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigger
839 "03000000d6200000022800001010000,PowerA Wired Controller for Xbox
       One,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger
840 "03000000c62400001a58000001010000,PowerA Xbox One
       Cabled,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigg
841 "03000000c62400001a54000001010000,PowerA Xbox One Mini Wired
       Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
842 "030000006d040000d2ca000011010000,Precision
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b
843 "03000000ff1100004133000010010000,PS2
       Controller,a:b2,b:b1,back:b8,leftshoulder:b6,lefttrigger:b4,leftx:a0,lefty:a1,rightshoulder:b7,righttrigger:b5,start:b9
844 "03000000341a0000360800011010000,PS3
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
845 "030000004c0500006802000010010000,PS3
       Controller,a:b14,b:b13,back:b0,dpdown:b6,dpleft:b7,dpright:b5,dpup:b4,guide:b16,leftshoulder:b10,leftstick:b1,lefttrigg
846 "030000004c0500006802000010810000,PS3
       Controller,a:b0,b:b1,back:b8,dpdown:b14,dpleft:b15,dpright:b16,dpup:b13,guide:b10,leftshoulder:b4,leftstick:b11,lefttri
847 "030000004c0500006802000011010000,PS3
       Controller,a:b14,b:b13,back:b0,dpdown:b6,dpleft:b7,dpright:b5,dpup:b4,guide:b16,leftshoulder:b10,leftstick:b1,lefttrigg
848 "030000004c0500006802000011810000,PS3
       Controller,a:b0,b:b1,back:b8,dpdown:b14,dpleft:b15,dpright:b16,dpup:b13,guide:b10,leftshoulder:b4,leftstick:b11,lefttri
849 "030000006f0e0000140200001101000,PS3
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
850 "030000008f0e0000300000010010000,PS3
       Controller,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b
851 "050000004c0500006802000000000000,PS3
       Controller,a:b14,b:b13,back:b0,dpdown:b6,dpleft:b7,dpright:b5,dpup:b4,guide:b16,leftshoulder:b10,leftstick:b1,lefttrigg
852 "050000004c0500006802000000010000,PS3
       Controller,a:b14,b:b13,back:b0,dpdown:b6,dpleft:b7,dpright:b5,dpup:b4,guide:b16,leftshoulder:b10,leftstick:b1,lefttrigg
853 "050000004c0500006802000000800000,PS3
       Controller,a:b0,b:b1,back:b8,dpdown:b14,dpleft:b15,dpright:b16,dpup:b13,guide:b10,leftshoulder:b4,leftstick:b11,lefttri
854 "050000004c0500006802000000810000,PS3
       Controller,a:b0,b:b1,back:b8,dpdown:b14,dpleft:b15,dpright:b16,dpup:b13,guide:b10,leftshoulder:b4,leftstick:b11,lefttri
855 "05000000504c415953544154494f4e00,PS3
       Controller,a:b14,b:b13,back:b0,dpdown:b6,dpleft:b7,dpright:b5,dpup:b4,guide:b16,leftshoulder:b10,leftstick:b1,lefttrigg
856 "060000004c0500006802000000010000,PS3
       Controller,a:b14,b:b13,back:b0,dpdown:b6,dpleft:b7,dpright:b5,dpup:b4,guide:b16,leftshoulder:b10,leftstick:b1,lefttrigg
857 "030000004c050000a00b000011010000,PS4
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
858 "030000004c050000a00b000011810000,PS4
       Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b11,le
859 "030000004c050000c405000011010000,PS4
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
860 "030000004c050000c405000011810000,PS4
       Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b11,le
861 "030000004c050000cc09000000010000,PS4
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
862 "030000004c050000cc09000011010000,PS4
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
863 "030000004c050000cc09000011810000,PS4
       Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b11,le
864 "03000000c0110000140000011010000,PS4
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b
865 "050000004c050000c405000000010000,PS4
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
866 "050000004c050000c405000000810000,PS4
       Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b11,le
867 "050000004c050000c405000001800000,PS4
       Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b11,le
```

```
868 "050000004c050000cc09000000010000,PS4
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
869 "050000004c050000cc09000000810000,PS4
       Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b11,le
870 "050000004c050000cc09000001800000,PS4
       Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b11,le
871 "030000004c050000e60c000011010000,PS5
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
872 "050000004c050000e60c000000010000,PS5
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
873
       "03000000ff000000cb01000010010000,PSP,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,
874 "03000000300f000012110000011010000,QanBa Arcade
       JoyStick,a:b2,b:b0,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b5,lefttrigger:b4,left
875 "030000009b2800004200000001010000,Raphnet Technologies Dual NES to USB
       v2.0,a:b0,b:b1,back:b2,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,start:b3,platform:Linux,",
876 "030000009b2800003200000001010000,Raphnet Technologies GC/N64 to USB
       v3.4,a:b0,b:b7,dpdown:b11,dpleft:b12,dpright:b13,dpup:b10,lefttrigger:b4,leftx:a0,lefty:a1,rightshoulder:b2,righttrigge
877 "030000009b2800006000000001010000,Raphnet Technologies GC/N64 to USB
       v3.6,a:b0,b:b7,dpdown:b11,dpleft:b12,dpright:b13,dpup:b10,lefttrigger:b4,leftx:a0,lefty:a1,rightshoulder:b2,righttrigge
878 "030000009b2800000300000001010000,raphnet.net 4nes4snes
       v1.5,a:b0,b:b4,back:b2,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b3,x:b1,y:b5,platform:Linux,",
879 "030000008916000001fd000024010000,Razer Onza Classic
       Edition,a:b0,b:b1,back:b6,dpdown:b14,dpleft:b11,dpright:b12,dpup:b13,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger
880 "030000008916000000fd000024010000,Razer Onza Tournament
       Edition,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttric
881 "030000003215000002040000011010000,Razer Panthera
       (PS3),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
882 "030000003215000001040000011010000,Razer Panthera
       (PS4),a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
883 "030000003215000008100000011010000,Razer Panthera Evo Arcade Stick for
       PS4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b13,leftshoulder:b4,leftstick:b10,lefttrigge
884 "030000003215000000100000011010000,Razer
       RAIJU,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttric
885 "030000003215000005070000000010000,Razer Raiju
       Mobile,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b21,leftshoulder:b6,leftstick:b13,lefttr
886 "030000003215000000110000011010000,Razer Raion Fightpad for
       PS4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
887 "030000008916000000fe000024010000,Razer
       Sabertooth,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
888 "03000000c6240000045d000024010000,Razer
       Sabertooth,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
889 "03000000c6240000045d000025010000,Razer
       Sabertooth,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
890 "030000003215000000090000011010000,Razer
       Serval,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigg
891 "050000003215000000090000163a0000,Razer
       Serval,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigg
892 "03000000321500000030a000001010000,Razer
       Wildcat,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrig
893 "030000007900000011000000010010000,Retrolink SNES
       Controller,a:b2,b:b1,back:b8,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b4,rightshoulder:b5,start:b9,x:b3,
894 "030000008117000099a000001010000,Retronic Adapter,a:b0,leftx:a0,lefty:a1,platform:Linux,",
895
       "03000000000f000000300000000010000,RetroPad,a:b1,b:b5,back:b2,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:
896 "030000006b140000010d000011010000,Revolution Pro
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
897 "030000006b140000130d000011010000,Revolution Pro Controller
       3,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigger
898 "030000006f0e00001f01000000010000,Rock
       Candy,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigge
899 "030000006f0e00001e01000011010000,Rock Candy PS3
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
900 "030000006f0e00004601000001010000,Rock Candy Xbox One
       Controller,a:b0,b:b1,back:b6,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger:a2,leftx:a0,lefty:a1,rightshoulder:b5,ri
901 "03000000a306000023f6000011010000,Saitek Cyborg V.1 Game
       Pad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
902 "03000000a3060000100500000010000,Saitek
       P150,a:b0,b:b1,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b7,lefttrigger:b6,rightshoulder:b2,righttrigger
903 "03000000a3060000070100000010000,Saitek
       P220,a:b2,b:b3,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b6,lefttrigger:b7,rightshoulder:b4,righttrigger
904 "03000000a30600000cff000010010000,Saitek P2500 Force Rumble
       Pad,a:b2,b:b3,back:b11,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b8,lefttrigger:b6,leftx
905 "03000000a30600000c04000011010000,Saitek P2900 Wireless
       Pad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b9,leftshoulder:b6,leftstick:b10,lefttrigger
906 "03000000300f000012010000010010000,Saitek
       P380,a:b2,b:b3,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b5,left
907 "03000000a3060000090100000010000,Saitek
       P880,a:b2,b:b3,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b8,lefttrigger:b6,leftx:a0,left
908 "03000000a30600000b04000000010000,Saitek P990 Dual Analog
       Pad,a:b1,b:b2,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,leftx
909 "03000000a306000018f5000010010000,Saitek PLC Saitek P3200 Rumble
       Pad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:a2,leftx
910 "03000000a306000020f6000011010000,Saitek PS2700 Rumble
       Pad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
911
       "03000000d81d00000e00000010010000,Savior,a:b0,b:b1,back:b8,leftshoulder:b6,leftstick:b10,lefttrigger:b7,leftx:a0,lefty
```

```
912 "03000000f025000021c1000010010000,ShanWan Gioteck PS3 Wired
       Controller,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b
913 "03000000632500007505000010010000,SHANWAN PS3/PC
       Gamepad,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttr
914 "03000000bc2000000055000010010000,ShanWan PS3/PC Wired
       GamePad,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:a5,
915 "030000005f140000c501000010010000,SHANWAN Trust
       Gamepad,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttr
916 "03000000632500002305000010010000,ShanWan USB
       Gamepad,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,l
917
       "03000000341a00000908000010010000,SL-6566,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder
918 "030000004c050000e60c000011810000,Sony
       DualSense,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b11,left
919 "050000004c050000e60c000000810000,Sony DualSense
       ,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b11,lefttrigger:a
920 "03000000250900000500000000010000,Sony PS2 pad with SmartJoy
       adapter,a:b2,b:b1,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b10,lefttrigger:b4,l
921 "030000005e0400008e02000073050000,Speedlink TORID Wireless
       Gamepad,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrig
922 "030000005e0400008e02000020200000,SpeedLink XEOX Pro Analog Gamepad
       pad,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger
923 "03000000d11800000094000011010000,Stadia
       Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
924 "03000000de2800000112000001000000,Steam
       Controller,a:b0,b:b1,back:b6,dpdown:b14,dpleft:b15,dpright:b13,dpup:b12,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigg
925 "03000000de2800000211000001000000,Steam
       Controller,a:b0,b:b1,back:b6,dpdown:b14,dpleft:b15,dpright:b13,dpup:b12,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigg
926 "03000000de2800000211000011010000,Steam
       Controller,a:b2,b:b3,back:b10,dpdown:b18,dpleft:b19,dpright:b20,dpup:b17,guide:b12,leftshoulder:b6,leftstick:b13,lefttr
927 "03000000de2800004211000001000000,Steam
       Controller,a:b0,b:b1,back:b6,dpdown:b14,dpleft:b15,dpright:b13,dpup:b12,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigg
928 "03000000de2800004211000011010000,Steam
       Controller,a:b2,b:b3,back:b10,dpdown:b18,dpleft:b19,dpright:b20,dpup:b17,guide:b12,leftshoulder:b6,leftstick:b13,lefttr
929 "03000000de280000fc1100000010000,Steam
       Controller,a:b0,b:b1,back:b6,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger:a2,leftx:a0,lefty:a1,rightshoulder:b5,ri
930 "05000000de2800000212000001000000,Steam
       Controller,a:b0,b:b1,back:b6,dpdown:b14,dpleft:b15,dpright:b13,dpup:b12,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigg
931 "05000000de2800000511000001000000,Steam
       Controller,a:b0,b:b1,back:b6,dpdown:b14,dpleft:b15,dpright:b13,dpup:b12,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigg
932 "05000000de2800000611000001000000,Steam
       Controller,a:b0,b:b1,back:b6,dpdown:b14,dpleft:b15,dpright:b13,dpup:b12,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigg
933 "03000000de280000ff11000001000000,Steam Virtual
       Gamepad,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrig
934 "03000000381000003014000075010000,SteelSeries Stratus
       Duo,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger
935 "03000000381000003114000075010000,SteelSeries Stratus
       Duo,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger
936 "05000000110100003114000001b010000,SteelSeries Stratus
       Duo,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b32,leftshoulder:b6,leftstick:b13,lefttrigg
937 "05000000110100001914000009010000,SteelSeries Stratus
       XL,a:b0,b:b1,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:a5,leftx:a0,lefty
938 "03000000ad1b000038f0000090040000,Street Fighter IV FightStick
       TE,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,lefttrigger:a2,leftx:a0,le
939 "030000003b07000004a1000000010000,Suncom SFX Plus for
       USB,a:b0,b:b2,back:b7,leftshoulder:b6,lefttrigger:b4,leftx:a0,lefty:a1,rightshoulder:b9,righttrigger:b5,start:b8,x:b1,y
940 "03000000666600000488000000010000,Super Joy Box 5
       Pro,a:b2,b:b1,back:b9,dpdown:b14,dpleft:b15,dpright:b13,dpup:b12,leftshoulder:b6,leftstick:b10,lefttrigger:b4,leftx:a0,
941 "0300000000f00000f100000000010000,Super
       RetroPort,a:b1,b:b5,back:b2,leftshoulder:b6,leftx:a0,lefty:a1,rightshoulder:b7,start:b3,x:b0,y:b4,platform:Linux,",
942 "03000000457500002211000010010000,SZMY-POWER CO. LTD.
       GAMEPAD,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttr
943 "030000008f0e00000d31000010010000,SZMY-POWER CO. LTD. GAMEPAD 3
       TURBO,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrig
944 "030000008f0e00001431000010010000,SZMY-POWER CO. LTD. PS3
       gamepad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttr
945 "030000004f04000020b3000010010000,Thrustmaster 2 in 1
       DT,a:b0,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b5,leftx
946 "030000004f04000015b3000010010000,Thrustmaster Dual Analog
       4,a:b0,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b5,leftx:a
947 "030000004f04000023b3000000010000,Thrustmaster Dual Trigger
       3-in-1,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,le
948 "030000004f0400000ed0000011010000,ThrustMaster eSwap PRO
       Controller,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
949 "03000000b5070000039900000010000,Thrustmaster Firestorm Digital
       2,a:b2,b:b4,back:b11,leftshoulder:b6,leftstick:b10,lefttrigger:b7,leftx:a0,lefty:a1,rightshoulder:b8,rightstick:b0,righ
950 "030000004f04000003b3000010010000,Thrustmaster Firestorm Dual Analog
       2,a:b0,b:b2,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b8,leftx:a0,lefty
951 "030000004f04000000b3000010010000,Thrustmaster Firestorm Dual
       Power,a:b0,b:b2,back:b9,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b11,lefttrigg
952 "030000004f04000026b3000002040000,Thrustmaster Gamepad GP
       XID,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger
953 "03000000c6240000025b000002020000,Thrustmaster GPX
       Gamepad,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrig
954 "030000004f04000008d0000000010000,Thrustmaster Run N Drive
       Wireless,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,
955 "030000004f04000009d0000000010000,Thrustmaster Run N Drive Wireless
```

```
      PS3,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
956 "030000004f04000007d0000000010000,Thrustmaster T Mini
      Wireless,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,leftt
957 "030000004f04000012b3000010010000,Thrustmaster vibrating
      gamepad,a:b0,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b5,l
958 "03000000bd12000015d0000010010000,Tomee SNES USB
      Controller,a:b2,b:b1,back:b8,dpdown:+a1,dpleft:-a0,dpright:+a0,dpup:-a1,leftshoulder:b4,rightshoulder:b5,start:b9,x:b3,
959 "03000000d814000007cd000011010000,Toodles 2008 Chimp
      PC/PS3,a:b0,b:b1,back:b8,leftshoulder:b4,lefttrigger:b6,leftx:a0,lefty:a1,rightshoulder:b5,righttrigger:b7,start:b9,x:b
960
      "030000005e0400008e02000070050000,Torid,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,lefts
961
      "03000000c0110000059100001l010000,Torid,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b
962 "03000000100800000100000010010000,Twin USB PS2
      Adapter,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b10,lefttrigger:b4,l
963 "0300000010080000030000010010000,USB
      Gamepad,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b10,lefttrigger:b4,l
964 "03000000790000006000000007010000,USB
      gamepad,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b6,l
965 "0300000079000000110000000010000,USB
      Gamepad1,a:b2,b:b1,back:b8,dpdown:a0,dpleft:a1,dpright:a2,dpup:a4,start:b9,platform:Linux,",
966 "030000006f0e00000302000011010000,Victrix Pro Fight Stick for
      PS4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,lefttrigger:b6,rightshou
967 "030000006f0e00000702000011010000,Victrix Pro Fight Stick for
      PS4,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,lefttrigge
968
      "05000000ac05000032320000001000000,VR-BOX,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder
969 "03000000791d00000103000010010000,Wii Classic
      Controller,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b6,lefttrigger:b4,le
970 "050000000d0f0000f600000001000000,Wireless HORIPAD Switch Pro
      Controller,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
971 "030000005e0400008e02000010010000,X360
      Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
972 "030000005e0400008e02000014010000,X360
      Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
973 "030000005e0400001907000000010000,X360 Wireless
      Controller,a:b0,b:b1,back:b6,dpdown:b14,dpleft:b11,dpright:b12,dpup:b13,guide:b8,leftshoulder:b4,leftstick:b9,lefttrig
974 "030000005e0400009102000007010000,X360 Wireless
      Controller,a:b0,b:b1,back:b6,dpdown:b14,dpleft:b11,dpright:b12,dpup:b13,guide:b8,leftshoulder:b4,leftstick:b9,lefttrig
975 "030000005e040000a102000000010000,X360 Wireless
      Controller,a:b0,b:b1,back:b6,dpdown:b14,dpleft:b11,dpright:b12,dpup:b13,guide:b8,leftshoulder:b4,leftstick:b9,lefttrig
976 "030000005e040000a102000007010000,X360 Wireless
      Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
977 "00000000058626f782033363020576900,Xbox 360 Wireless
      Controller,a:b0,b:b1,back:b14,dpdown:b11,dpleft:b12,dpright:b13,dpup:b10,guide:b7,leftshoulder:b4,leftstick:b8,lefttrig
978 "030000005e040000a102000014010000,Xbox 360 Wireless Receiver
      (XBOX),a:b0,b:b1,back:b6,dpdown:b14,dpleft:b11,dpright:b12,dpup:b13,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger:a
979 "00000000058626f782047616d65706100,Xbox Gamepad (userspace
      driver),a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrig
980 "030000005e040000d102000002010000,Xbox One
      Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
981 "050000005e040000fd02000030110000,Xbox One
      Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
982 "050000005e040000050b000002090000,Xbox One Elite Series
      2,a:b0,b:b1,back:b136,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b6,leftstick:b13,lefttrigger:a6,leftx
983 "030000005e040000ea02000000000000,Xbox One Wireless
      Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
984 "050000005e040000e002000003090000,Xbox One Wireless
      Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b10,leftshoulder:b4,leftstick:b8,left
985 "050000005e040000fd02000003090000,Xbox One Wireless
      Controller,a:b0,b:b1,back:b15,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b16,leftshoulder:b6,leftstick:b13,le
986 "030000005e040000ea02000001030000,Xbox One Wireless Controller (Model
      1708),a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigge
987 "030000005e040000120b000001050000,Xbox Series
      Controller,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,leftt
988 "030000005e040000130b000005050000,Xbox Series
      Controller,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,le
989 "050000005e040000130b000001050000,Xbox Series
      Controller,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,le
990 "050000005e040000130b000005050000,Xbox Series
      Controller,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b6,leftstick:b13,le
991 "030000005e040000120b000005050000,XBox Series
      pad,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrigger
992 "030000005e0400008e02000000010000,xbox360 Wireless
      EasySMX,a:b0,b:b1,back:b6,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b8,leftshoulder:b4,leftstick:b9,lefttrig
993 "03000000450c0000204300001001000,XEOX Gamepad
      SL-6556-BK,a:b0,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,leftshoulder:b4,leftstick:b10,lefttrigger:b
994 "03000000ac0500005b05000010010000,Xiaoji
      Gamesir-G3w,a:b2,b:b1,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,le
995 "050000001727000044310000029010000,XiaoMi Game
      Controller,a:b0,b:b1,back:b10,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b20,leftshoulder:b6,leftstick:b13,le
996 "03000000c0160000e105000001010000,Xin-Mo Xin-Mo Dual
      Arcade,a:b4,b:b3,back:b6,dpdown:b12,dpleft:b13,dpright:b14,dpup:b11,guide:b9,leftshoulder:b2,leftx:a0,lefty:a1,rightsho
997 "03000000120c0000100e000011010000,ZEROPLUS P4
      Gamepad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,leftt
998 "03000000120c0000101e000011010000,ZEROPLUS P4 Wired
      Gamepad,a:b1,b:b2,back:b8,dpdown:h0.4,dpleft:h0.8,dpright:h0.2,dpup:h0.1,guide:b12,leftshoulder:b4,leftstick:b10,leftt
```

```
999 #endif // GLFW_BUILD_LINUX_MAPPINGS
1000 };
1001
```

27.25 null_joystick.h

```
1 //========================================================================
2 // GLFW 3.4 - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2006-2017 Camilla Löwy <elmindreda@glfw.org>
5 //
6 // This software is provided 'as-is', without any express or implied
7 // warranty. In no event will the authors be held liable for any damages
8 // arising from the use of this software.
9 //
10 // Permission is granted to anyone to use this software for any purpose,
11 // including commercial applications, and to alter it and redistribute it
12 // freely, subject to the following restrictions:
13 //
14 // 1. The origin of this software must not be misrepresented; you must not
15 //    claim that you wrote the original software. If you use this software
16 //    in a product, an acknowledgment in the product documentation would
17 //    be appreciated but is not required.
18 //
19 // 2. Altered source versions must be plainly marked as such, and must not
20 //    be misrepresented as being the original software.
21 //
22 // 3. This notice may not be removed or altered from any source
23 //    distribution.
24 //
25 //========================================================================
26
27 GLFWbool _glfwInitJoysticksNull(void);
28 void _glfwTerminateJoysticksNull(void);
29 int _glfwPollJoystickNull(_GLFWjoystick* js, int mode);
30 const char* _glfwGetMappingNameNull(void);
31 void _glfwUpdateGamepadGUIDNull(char* guid);
32
```

27.26 null_platform.h

```
1 //========================================================================
2 // GLFW 3.4 - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2016 Google Inc.
5 // Copyright (c) 2016-2017 Camilla Löwy <elmindreda@glfw.org>
6 //
7 // This software is provided 'as-is', without any express or implied
8 // warranty. In no event will the authors be held liable for any damages
9 // arising from the use of this software.
10 //
11 // Permission is granted to anyone to use this software for any purpose,
12 // including commercial applications, and to alter it and redistribute it
13 // freely, subject to the following restrictions:
14 //
15 // 1. The origin of this software must not be misrepresented; you must not
16 //    claim that you wrote the original software. If you use this software
17 //    in a product, an acknowledgment in the product documentation would
18 //    be appreciated but is not required.
19 //
20 // 2. Altered source versions must be plainly marked as such, and must not
21 //    be misrepresented as being the original software.
22 //
23 // 3. This notice may not be removed or altered from any source
24 //    distribution.
25 //
26 //========================================================================
27
28 #define GLFW_NULL_WINDOW_STATE          _GLFWwindowNull null;
29 #define GLFW_NULL_LIBRARY_WINDOW_STATE  _GLFWlibraryNull null;
30 #define GLFW_NULL_MONITOR_STATE         _GLFWmonitorNull null;
31
32 #define GLFW_NULL_CONTEXT_STATE
33 #define GLFW_NULL_CURSOR_STATE
34 #define GLFW_NULL_LIBRARY_CONTEXT_STATE
35
36
37 // Null-specific per-window data
38 //
```

```
39 typedef struct _GLFWwindowNull
40 {
41     int             xpos;
42     int             ypos;
43     int             width;
44     int             height;
45     char*           title;
46     GLFWbool        visible;
47     GLFWbool        iconified;
48     GLFWbool        maximized;
49     GLFWbool        resizable;
50     GLFWbool        decorated;
51     GLFWbool        floating;
52     GLFWbool        transparent;
53     float           opacity;
54 } _GLFWwindowNull;
55
56 // Null-specific per-monitor data
57 //
58 typedef struct _GLFWmonitorNull
59 {
60     GLFWgammaramp   ramp;
61 } _GLFWmonitorNull;
62
63 // Null-specific global data
64 //
65 typedef struct _GLFWlibraryNull
66 {
67     int             xcursor;
68     int             ycursor;
69     char*           clipboardString;
70     _GLFWwindow*    focusedWindow;
71 } _GLFWlibraryNull;
72
73 void _glfwPollMonitorsNull(void);
74
75 GLFWbool _glfwConnectNull(int platformID, _GLFWplatform* platform);
76 int _glfwInitNull(void);
77 void _glfwTerminateNull(void);
78
79 void _glfwFreeMonitorNull(_GLFWmonitor* monitor);
80 void _glfwGetMonitorPosNull(_GLFWmonitor* monitor, int* xpos, int* ypos);
81 void _glfwGetMonitorContentScaleNull(_GLFWmonitor* monitor, float* xscale, float* yscale);
82 void _glfwGetMonitorWorkareaNull(_GLFWmonitor* monitor, int* xpos, int* ypos, int* width, int* height);
83 GLFWvidmode* _glfwGetVideoModesNull(_GLFWmonitor* monitor, int* found);
84 void _glfwGetVideoModeNull(_GLFWmonitor* monitor, GLFWvidmode* mode);
85 GLFWbool _glfwGetGammaRampNull(_GLFWmonitor* monitor, GLFWgammaramp* ramp);
86 void _glfwSetGammaRampNull(_GLFWmonitor* monitor, const GLFWgammaramp* ramp);
87
88 int _glfwCreateWindowNull(_GLFWwindow* window, const _GLFWwndconfig* wndconfig, const _GLFWctxconfig*
       ctxconfig, const _GLFWfbconfig* fbconfig);
89 void _glfwDestroyWindowNull(_GLFWwindow* window);
90 void _glfwSetWindowTitleNull(_GLFWwindow* window, const char* title);
91 void _glfwSetWindowIconNull(_GLFWwindow* window, int count, const GLFWimage* images);
92 void _glfwSetWindowMonitorNull(_GLFWwindow* window, _GLFWmonitor* monitor, int xpos, int ypos, int width,
       int height, int refreshRate);
93 void _glfwGetWindowPosNull(_GLFWwindow* window, int* xpos, int* ypos);
94 void _glfwSetWindowPosNull(_GLFWwindow* window, int xpos, int ypos);
95 void _glfwGetWindowSizeNull(_GLFWwindow* window, int* width, int* height);
96 void _glfwSetWindowSizeNull(_GLFWwindow* window, int width, int height);
97 void _glfwSetWindowSizeLimitsNull(_GLFWwindow* window, int minwidth, int minheight, int maxwidth, int
       maxheight);
98 void _glfwSetWindowAspectRatioNull(_GLFWwindow* window, int n, int d);
99 void _glfwGetFramebufferSizeNull(_GLFWwindow* window, int* width, int* height);
100 void _glfwGetWindowFrameSizeNull(_GLFWwindow* window, int* left, int* top, int* right, int* bottom);
101 void _glfwGetWindowContentScaleNull(_GLFWwindow* window, float* xscale, float* yscale);
102 void _glfwIconifyWindowNull(_GLFWwindow* window);
103 void _glfwRestoreWindowNull(_GLFWwindow* window);
104 void _glfwMaximizeWindowNull(_GLFWwindow* window);
105 int _glfwWindowMaximizedNull(_GLFWwindow* window);
106 int _glfwWindowHoveredNull(_GLFWwindow* window);
107 int _glfwFramebufferTransparentNull(_GLFWwindow* window);
108 void _glfwSetWindowResizableNull(_GLFWwindow* window, GLFWbool enabled);
109 void _glfwSetWindowDecoratedNull(_GLFWwindow* window, GLFWbool enabled);
110 void _glfwSetWindowFloatingNull(_GLFWwindow* window, GLFWbool enabled);
111 void _glfwSetWindowMousePassthroughNull(_GLFWwindow* window, GLFWbool enabled);
112 float _glfwGetWindowOpacityNull(_GLFWwindow* window);
113 void _glfwSetWindowOpacityNull(_GLFWwindow* window, float opacity);
114 void _glfwSetRawMouseMotionNull(_GLFWwindow *window, GLFWbool enabled);
115 GLFWbool _glfwRawMouseMotionSupportedNull(void);
116 void _glfwShowWindowNull(_GLFWwindow* window);
117 void _glfwRequestWindowAttentionNull(_GLFWwindow* window);
118 void _glfwRequestWindowAttentionNull(_GLFWwindow* window);
119 void _glfwHideWindowNull(_GLFWwindow* window);
120 void _glfwFocusWindowNull(_GLFWwindow* window);
121 int _glfwWindowFocusedNull(_GLFWwindow* window);
122 int _glfwWindowIconifiedNull(_GLFWwindow* window);
```

```
123 int _glfwWindowVisibleNull(_GLFWwindow* window);
124 void _glfwPollEventsNull(void);
125 void _glfwWaitEventsNull(void);
126 void _glfwWaitEventsTimeoutNull(double timeout);
127 void _glfwPostEmptyEventNull(void);
128 void _glfwGetCursorPosNull(_GLFWwindow* window, double* xpos, double* ypos);
129 void _glfwSetCursorPosNull(_GLFWwindow* window, double x, double y);
130 void _glfwSetCursorModeNull(_GLFWwindow* window, int mode);
131 int _glfwCreateCursorNull(_GLFWcursor* cursor, const GLFWimage* image, int xhot, int yhot);
132 int _glfwCreateStandardCursorNull(_GLFWcursor* cursor, int shape);
133 void _glfwDestroyCursorNull(_GLFWcursor* cursor);
134 void _glfwSetCursorNull(_GLFWwindow* window, _GLFWcursor* cursor);
135 void _glfwSetClipboardStringNull(const char* string);
136 const char* _glfwGetClipboardStringNull(void);
137 const char* _glfwGetScancodeNameNull(int scancode);
138 int _glfwGetKeyScancodeNull(int key);
139
140 EGLenum _glfwGetEGLPlatformNull(EGLint** attribs);
141 EGLNativeDisplayType _glfwGetEGLNativeDisplayNull(void);
142 EGLNativeWindowType _glfwGetEGLNativeWindowNull(_GLFWwindow* window);
143
144 void _glfwGetRequiredInstanceExtensionsNull(char** extensions);
145 int _glfwGetPhysicalDevicePresentationSupportNull(VkInstance instance, VkPhysicalDevice device, uint32_t
       queuefamily);
146 VkResult _glfwCreateWindowSurfaceNull(VkInstance instance, _GLFWwindow* window, const
       VkAllocationCallbacks* allocator, VkSurfaceKHR* surface);
147
148 void _glfwPollMonitorsNull(void);
149
```

27.27 platform.h

```
1 //========================================================================
2 // GLFW 3.4 - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2002-2006 Marcus Geelnard
5 // Copyright (c) 2006-2018 Camilla Löwy <elmindreda@glfw.org>
6 //
7 // This software is provided 'as-is', without any express or implied
8 // warranty. In no event will the authors be held liable for any damages
9 // arising from the use of this software.
10 //
11 // Permission is granted to anyone to use this software for any purpose,
12 // including commercial applications, and to alter it and redistribute it
13 // freely, subject to the following restrictions:
14 //
15 // 1. The origin of this software must not be misrepresented; you must not
16 //    claim that you wrote the original software. If you use this software
17 //    in a product, an acknowledgment in the product documentation would
18 //    be appreciated but is not required.
19 //
20 // 2. Altered source versions must be plainly marked as such, and must not
21 //    be misrepresented as being the original software.
22 //
23 // 3. This notice may not be removed or altered from any source
24 //    distribution.
25 //
26 //========================================================================
27
28 #include "null_platform.h"
29
30 #if defined(_GLFW_WIN32)
31  #include "win32_platform.h"
32 #else
33  #define GLFW_WIN32_WINDOW_STATE
34  #define GLFW_WIN32_MONITOR_STATE
35  #define GLFW_WIN32_CURSOR_STATE
36  #define GLFW_WIN32_LIBRARY_WINDOW_STATE
37  #define GLFW_WGL_CONTEXT_STATE
38  #define GLFW_WGL_LIBRARY_CONTEXT_STATE
39 #endif
40
41 #if defined(_GLFW_COCOA)
42  #include "cocoa_platform.h"
43 #else
44  #define GLFW_COCOA_WINDOW_STATE
45  #define GLFW_COCOA_MONITOR_STATE
46  #define GLFW_COCOA_CURSOR_STATE
47  #define GLFW_COCOA_LIBRARY_WINDOW_STATE
48  #define GLFW_NSGL_CONTEXT_STATE
49  #define GLFW_NSGL_LIBRARY_CONTEXT_STATE
50 #endif
51
```

```
52 #if defined(_GLFW_WAYLAND)
53  #include "wl_platform.h"
54 #else
55  #define GLFW_WAYLAND_WINDOW_STATE
56  #define GLFW_WAYLAND_MONITOR_STATE
57  #define GLFW_WAYLAND_CURSOR_STATE
58  #define GLFW_WAYLAND_LIBRARY_WINDOW_STATE
59 #endif
60
61 #if defined(_GLFW_X11)
62  #include "x11_platform.h"
63 #else
64  #define GLFW_X11_WINDOW_STATE
65  #define GLFW_X11_MONITOR_STATE
66  #define GLFW_X11_CURSOR_STATE
67  #define GLFW_X11_LIBRARY_WINDOW_STATE
68  #define GLFW_GLX_CONTEXT_STATE
69  #define GLFW_GLX_LIBRARY_CONTEXT_STATE
70 #endif
71
72 #include "null_joystick.h"
73
74 #if defined(_GLFW_WIN32)
75  #include "win32_joystick.h"
76 #else
77  #define GLFW_WIN32_JOYSTICK_STATE
78  #define GLFW_WIN32_LIBRARY_JOYSTICK_STATE
79 #endif
80
81 #if defined(_GLFW_COCOA)
82  #include "cocoa_joystick.h"
83 #else
84  #define GLFW_COCOA_JOYSTICK_STATE
85  #define GLFW_COCOA_LIBRARY_JOYSTICK_STATE
86 #endif
87
88 #if (defined(_GLFW_X11) || defined(_GLFW_WAYLAND)) && defined(__linux__)
89  #include "linux_joystick.h"
90 #else
91  #define GLFW_LINUX_JOYSTICK_STATE
92  #define GLFW_LINUX_LIBRARY_JOYSTICK_STATE
93 #endif
94
95 #if defined(_WIN32)
96  #include "win32_thread.h"
97  #define GLFW_POSIX_TLS_STATE
98  #define GLFW_POSIX_MUTEX_STATE
99 #else
100  #include "posix_thread.h"
101  #define GLFW_WIN32_TLS_STATE
102  #define GLFW_WIN32_MUTEX_STATE
103 #endif
104
105 #if defined(_WIN32)
106  #include "win32_time.h"
107  #define GLFW_POSIX_LIBRARY_TIMER_STATE
108  #define GLFW_COCOA_LIBRARY_TIMER_STATE
109 #elif defined(__APPLE__)
110  #include "cocoa_time.h"
111  #define GLFW_WIN32_LIBRARY_TIMER_STATE
112  #define GLFW_POSIX_LIBRARY_TIMER_STATE
113 #else
114  #include "posix_time.h"
115  #define GLFW_WIN32_LIBRARY_TIMER_STATE
116  #define GLFW_COCOA_LIBRARY_TIMER_STATE
117 #endif
118
119 #define GLFW_PLATFORM_WINDOW_STATE \
120         GLFW_WIN32_WINDOW_STATE \
121         GLFW_COCOA_WINDOW_STATE \
122         GLFW_WAYLAND_WINDOW_STATE \
123         GLFW_X11_WINDOW_STATE \
124         GLFW_NULL_WINDOW_STATE \
125
126 #define GLFW_PLATFORM_MONITOR_STATE \
127         GLFW_WIN32_MONITOR_STATE \
128         GLFW_COCOA_MONITOR_STATE \
129         GLFW_WAYLAND_MONITOR_STATE \
130         GLFW_X11_MONITOR_STATE \
131         GLFW_NULL_MONITOR_STATE \
132
133 #define GLFW_PLATFORM_CURSOR_STATE \
134         GLFW_WIN32_CURSOR_STATE \
135         GLFW_COCOA_CURSOR_STATE \
136         GLFW_WAYLAND_CURSOR_STATE \
137         GLFW_X11_CURSOR_STATE \
138         GLFW_NULL_CURSOR_STATE \
```

```
139
140 #define GLFW_PLATFORM_JOYSTICK_STATE \
141         GLFW_WIN32_JOYSTICK_STATE \
142         GLFW_COCOA_JOYSTICK_STATE \
143         GLFW_LINUX_JOYSTICK_STATE
144
145 #define GLFW_PLATFORM_TLS_STATE \
146         GLFW_WIN32_TLS_STATE \
147         GLFW_POSIX_TLS_STATE \
148
149 #define GLFW_PLATFORM_MUTEX_STATE \
150         GLFW_WIN32_MUTEX_STATE \
151         GLFW_POSIX_MUTEX_STATE \
152
153 #define GLFW_PLATFORM_LIBRARY_WINDOW_STATE \
154         GLFW_WIN32_LIBRARY_WINDOW_STATE \
155         GLFW_COCOA_LIBRARY_WINDOW_STATE \
156         GLFW_WAYLAND_LIBRARY_WINDOW_STATE \
157         GLFW_X11_LIBRARY_WINDOW_STATE \
158         GLFW_NULL_LIBRARY_WINDOW_STATE \
159
160 #define GLFW_PLATFORM_LIBRARY_JOYSTICK_STATE \
161         GLFW_WIN32_LIBRARY_JOYSTICK_STATE \
162         GLFW_COCOA_LIBRARY_JOYSTICK_STATE \
163         GLFW_LINUX_LIBRARY_JOYSTICK_STATE
164
165 #define GLFW_PLATFORM_LIBRARY_TIMER_STATE \
166         GLFW_WIN32_LIBRARY_TIMER_STATE \
167         GLFW_COCOA_LIBRARY_TIMER_STATE \
168         GLFW_POSIX_LIBRARY_TIMER_STATE \
169
170 #define GLFW_PLATFORM_CONTEXT_STATE \
171         GLFW_WGL_CONTEXT_STATE \
172         GLFW_NSGL_CONTEXT_STATE \
173         GLFW_GLX_CONTEXT_STATE
174
175 #define GLFW_PLATFORM_LIBRARY_CONTEXT_STATE \
176         GLFW_WGL_LIBRARY_CONTEXT_STATE \
177         GLFW_NSGL_LIBRARY_CONTEXT_STATE \
178         GLFW_GLX_LIBRARY_CONTEXT_STATE
179
```

27.28 posix_poll.h

```
1 //========================================================================
2 // GLFW 3.4 POSIX - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2022 Camilla Löwy <elmindreda@glfw.org>
5 //
6 // This software is provided 'as-is', without any express or implied
7 // warranty. In no event will the authors be held liable for any damages
8 // arising from the use of this software.
9 //
10 // Permission is granted to anyone to use this software for any purpose,
11 // including commercial applications, and to alter it and redistribute it
12 // freely, subject to the following restrictions:
13 //
14 // 1. The origin of this software must not be misrepresented; you must not
15 //    claim that you wrote the original software. If you use this software
16 //    in a product, an acknowledgment in the product documentation would
17 //    be appreciated but is not required.
18 //
19 // 2. Altered source versions must be plainly marked as such, and must not
20 //    be misrepresented as being the original software.
21 //
22 // 3. This notice may not be removed or altered from any source
23 //    distribution.
24 //
25 //========================================================================
26 // It is fine to use C99 in this file because it will not be built with VS
27 //========================================================================
28
29 #include <poll.h>
30
31 GLFWbool _glfwPollPOSIX(struct pollfd* fds, nfds_t count, double* timeout);
32
```

27.29 posix_thread.h

```
1 //========================================================================
```

```
2 // GLFW 3.4 POSIX - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2002-2006 Marcus Geelnard
5 // Copyright (c) 2006-2017 Camilla Löwy <elmindreda@glfw.org>
6 //
7 // This software is provided 'as-is', without any express or implied
8 // warranty. In no event will the authors be held liable for any damages
9 // arising from the use of this software.
10 //
11 // Permission is granted to anyone to use this software for any purpose,
12 // including commercial applications, and to alter it and redistribute it
13 // freely, subject to the following restrictions:
14 //
15 // 1. The origin of this software must not be misrepresented; you must not
16 //    claim that you wrote the original software. If you use this software
17 //    in a product, an acknowledgment in the product documentation would
18 //    be appreciated but is not required.
19 //
20 // 2. Altered source versions must be plainly marked as such, and must not
21 //    be misrepresented as being the original software.
22 //
23 // 3. This notice may not be removed or altered from any source
24 //    distribution.
25 //
26 //========================================================================
27
28 #include <pthread.h>
29
30 #define GLFW_POSIX_TLS_STATE    _GLFWtlsPOSIX   posix;
31 #define GLFW_POSIX_MUTEX_STATE  _GLFWmutexPOSIX posix;
32
33
34 // POSIX-specific thread local storage data
35 //
36 typedef struct _GLFWtlsPOSIX
37 {
38     GLFWbool        allocated;
39     pthread_key_t   key;
40 } _GLFWtlsPOSIX;
41
42 // POSIX-specific mutex data
43 //
44 typedef struct _GLFWmutexPOSIX
45 {
46     GLFWbool        allocated;
47     pthread_mutex_t handle;
48 } _GLFWmutexPOSIX;
49
```

27.30 posix_time.h

```
1 //========================================================================
2 // GLFW 3.4 POSIX - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2002-2006 Marcus Geelnard
5 // Copyright (c) 2006-2017 Camilla Löwy <elmindreda@glfw.org>
6 //
7 // This software is provided 'as-is', without any express or implied
8 // warranty. In no event will the authors be held liable for any damages
9 // arising from the use of this software.
10 //
11 // Permission is granted to anyone to use this software for any purpose,
12 // including commercial applications, and to alter it and redistribute it
13 // freely, subject to the following restrictions:
14 //
15 // 1. The origin of this software must not be misrepresented; you must not
16 //    claim that you wrote the original software. If you use this software
17 //    in a product, an acknowledgment in the product documentation would
18 //    be appreciated but is not required.
19 //
20 // 2. Altered source versions must be plainly marked as such, and must not
21 //    be misrepresented as being the original software.
22 //
23 // 3. This notice may not be removed or altered from any source
24 //    distribution.
25 //
26 //========================================================================
27
28 #define GLFW_POSIX_LIBRARY_TIMER_STATE _GLFWtimerPOSIX posix;
29
30 #include <stdint.h>
31 #include <time.h>
32
```

```
33
34 // POSIX-specific global timer data
35 //
36 typedef struct _GLFWtimerPOSIX
37 {
38     clockid_t   clock;
39     uint64_t    frequency;
40 } _GLFWtimerPOSIX;
41
```

27.31 win32_joystick.h

```
1 //========================================================================
2 // GLFW 3.4 Win32 - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2006-2017 Camilla Löwy <elmindreda@glfw.org>
5 //
6 // This software is provided 'as-is', without any express or implied
7 // warranty. In no event will the authors be held liable for any damages
8 // arising from the use of this software.
9 //
10 // Permission is granted to anyone to use this software for any purpose,
11 // including commercial applications, and to alter it and redistribute it
12 // freely, subject to the following restrictions:
13 //
14 // 1. The origin of this software must not be misrepresented; you must not
15 //    claim that you wrote the original software. If you use this software
16 //    in a product, an acknowledgment in the product documentation would
17 //    be appreciated but is not required.
18 //
19 // 2. Altered source versions must be plainly marked as such, and must not
20 //    be misrepresented as being the original software.
21 //
22 // 3. This notice may not be removed or altered from any source
23 //    distribution.
24 //
25 //========================================================================
26
27 #define GLFW_WIN32_JOYSTICK_STATE         _GLFWjoystickWin32 win32;
28 #define GLFW_WIN32_LIBRARY_JOYSTICK_STATE
29
30 #define GLFW_BUILD_WIN32_MAPPINGS
31
32 // Joystick element (axis, button or slider)
33 //
34 typedef struct _GLFWjoyobjectWin32
35 {
36     int                     offset;
37     int                     type;
38 } _GLFWjoyobjectWin32;
39
40 // Win32-specific per-joystick data
41 //
42 typedef struct _GLFWjoystickWin32
43 {
44     _GLFWjoyobjectWin32*    objects;
45     int                     objectCount;
46     IDirectInputDevice8W*   device;
47     DWORD                   index;
48     GUID                    guid;
49 } _GLFWjoystickWin32;
50
51 void _glfwDetectJoystickConnectionWin32(void);
52 void _glfwDetectJoystickDisconnectionWin32(void);
53
```

27.32 win32_platform.h

```
1 //========================================================================
2 // GLFW 3.4 Win32 - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2002-2006 Marcus Geelnard
5 // Copyright (c) 2006-2019 Camilla Löwy <elmindreda@glfw.org>
6 //
7 // This software is provided 'as-is', without any express or implied
8 // warranty. In no event will the authors be held liable for any damages
9 // arising from the use of this software.
10 //
11 // Permission is granted to anyone to use this software for any purpose,
```

```
12 // including commercial applications, and to alter it and redistribute it
13 // freely, subject to the following restrictions:
14 //
15 // 1. The origin of this software must not be misrepresented; you must not
16 //    claim that you wrote the original software. If you use this software
17 //    in a product, an acknowledgment in the product documentation would
18 //    be appreciated but is not required.
19 //
20 // 2. Altered source versions must be plainly marked as such, and must not
21 //    be misrepresented as being the original software.
22 //
23 // 3. This notice may not be removed or altered from any source
24 //    distribution.
25 //
26 //========================================================================
27
28 // We don't need all the fancy stuff
29 #ifndef NOMINMAX
30  #define NOMINMAX
31 #endif
32
33 #ifndef VC_EXTRALEAN
34  #define VC_EXTRALEAN
35 #endif
36
37 #ifndef WIN32_LEAN_AND_MEAN
38  #define WIN32_LEAN_AND_MEAN
39 #endif
40
41 // This is a workaround for the fact that glfw3.h needs to export APIENTRY (for
42 // example to allow applications to correctly declare a GL_KHR_debug callback)
43 // but windows.h assumes no one will define APIENTRY before it does
44 #undef APIENTRY
45
46 // GLFW on Windows is Unicode only and does not work in MBCS mode
47 #ifndef UNICODE
48  #define UNICODE
49 #endif
50
51 // GLFW requires Windows XP or later
52 #if WINVER < 0x0501
53  #undef WINVER
54  #define WINVER 0x0501
55 #endif
56 #if _WIN32_WINNT < 0x0501
57  #undef _WIN32_WINNT
58  #define _WIN32_WINNT 0x0501
59 #endif
60
61 // GLFW uses DirectInput8 interfaces
62 #define DIRECTINPUT_VERSION 0x0800
63
64 // GLFW uses OEM cursor resources
65 #define OEMRESOURCE
66
67 #include <wctype.h>
68 #include <windows.h>
69 #include <dinput.h>
70 #include <xinput.h>
71 #include <dbt.h>
72
73 // HACK: Define macros that some windows.h variants don't
74 #ifndef WM_MOUSEHWHEEL
75  #define WM_MOUSEHWHEEL 0x020E
76 #endif
77 #ifndef WM_DWMCOMPOSITIONCHANGED
78  #define WM_DWMCOMPOSITIONCHANGED 0x031E
79 #endif
80 #ifndef WM_DWMCOLORIZATIONCOLORCHANGED
81  #define WM_DWMCOLORIZATIONCOLORCHANGED 0x0320
82 #endif
83 #ifndef WM_COPYGLOBALDATA
84  #define WM_COPYGLOBALDATA 0x0049
85 #endif
86 #ifndef WM_UNICHAR
87  #define WM_UNICHAR 0x0109
88 #endif
89 #ifndef UNICODE_NOCHAR
90  #define UNICODE_NOCHAR 0xFFFF
91 #endif
92 #ifndef WM_DPICHANGED
93  #define WM_DPICHANGED 0x02E0
94 #endif
95 #ifndef GET_XBUTTON_WPARAM
96  #define GET_XBUTTON_WPARAM(w) (HIWORD(w))
97 #endif
98 #ifndef EDS_ROTATEDMODE
```

```
99  #define EDS_ROTATEDMODE 0x00000004
100 #endif
101 #ifndef DISPLAY_DEVICE_ACTIVE
102  #define DISPLAY_DEVICE_ACTIVE 0x00000001
103 #endif
104 #ifndef _WIN32_WINNT_WINBLUE
105  #define _WIN32_WINNT_WINBLUE 0x0603
106 #endif
107 #ifndef _WIN32_WINNT_WIN8
108  #define _WIN32_WINNT_WIN8 0x0602
109 #endif
110 #ifndef WM_GETDPISCALEDSIZE
111  #define WM_GETDPISCALEDSIZE 0x02e4
112 #endif
113 #ifndef USER_DEFAULT_SCREEN_DPI
114  #define USER_DEFAULT_SCREEN_DPI 96
115 #endif
116 #ifndef OCR_HAND
117  #define OCR_HAND 32649
118 #endif
119
120 #if WINVER < 0x0601
121 typedef struct
122 {
123     DWORD cbSize;
124     DWORD ExtStatus;
125 } CHANGEFILTERSTRUCT;
126 #ifndef MSGFLT_ALLOW
127  #define MSGFLT_ALLOW 1
128 #endif
129 #endif /*Windows 7*/
130
131 #if WINVER < 0x0600
132 #define DWM_BB_ENABLE 0x00000001
133 #define DWM_BB_BLURREGION 0x00000002
134 typedef struct
135 {
136     DWORD dwFlags;
137     BOOL fEnable;
138     HRGN hRgnBlur;
139     BOOL fTransitionOnMaximized;
140 } DWM_BLURBEHIND;
141 #else
142  #include <dwmapi.h>
143 #endif /*Windows Vista*/
144
145 #ifndef DPI_ENUMS_DECLARED
146 typedef enum
147 {
148     PROCESS_DPI_UNAWARE = 0,
149     PROCESS_SYSTEM_DPI_AWARE = 1,
150     PROCESS_PER_MONITOR_DPI_AWARE = 2
151 } PROCESS_DPI_AWARENESS;
152 typedef enum
153 {
154     MDT_EFFECTIVE_DPI = 0,
155     MDT_ANGULAR_DPI = 1,
156     MDT_RAW_DPI = 2,
157     MDT_DEFAULT = MDT_EFFECTIVE_DPI
158 } MONITOR_DPI_TYPE;
159 #endif /*DPI_ENUMS_DECLARED*/
160
161 #ifndef DPI_AWARENESS_CONTEXT_PER_MONITOR_AWARE_V2
162 #define DPI_AWARENESS_CONTEXT_PER_MONITOR_AWARE_V2 ((HANDLE) -4)
163 #endif /*DPI_AWARENESS_CONTEXT_PER_MONITOR_AWARE_V2*/
164
165 // Replacement for versionhelpers.h macros, as we cannot rely on the
166 // application having a correct embedded manifest
167 //
168 #define IsWindowsVistaOrGreater()                                   \
169     _glfwIsWindowsVersionOrGreaterWin32(HIBYTE(_WIN32_WINNT_VISTA),    \
170                                         LOBYTE(_WIN32_WINNT_VISTA), 0)
171 #define IsWindows7OrGreater()                                       \
172     _glfwIsWindowsVersionOrGreaterWin32(HIBYTE(_WIN32_WINNT_WIN7),     \
173                                         LOBYTE(_WIN32_WINNT_WIN7), 0)
174 #define IsWindows8OrGreater()                                       \
175     _glfwIsWindowsVersionOrGreaterWin32(HIBYTE(_WIN32_WINNT_WIN8),     \
176                                         LOBYTE(_WIN32_WINNT_WIN8), 0)
177 #define IsWindows8Point1OrGreater()                                 \
178     _glfwIsWindowsVersionOrGreaterWin32(HIBYTE(_WIN32_WINNT_WINBLUE), \
179                                         LOBYTE(_WIN32_WINNT_WINBLUE), 0)
180
181 // Windows 10 Anniversary Update
182 #define _glfwIsWindows10Version1607OrGreaterWin32() \
183     _glfwIsWindows10BuildOrGreaterWin32(14393)
184 // Windows 10 Creators Update
185 #define _glfwIsWindows10Version1703OrGreaterWin32() \
```

```
186     _glfwIsWindows10BuildOrGreaterWin32(15063)
187
188 // HACK: Define macros that some xinput.h variants don't
189 #ifndef XINPUT_CAPS_WIRELESS
190  #define XINPUT_CAPS_WIRELESS 0x0002
191 #endif
192 #ifndef XINPUT_DEVSUBTYPE_WHEEL
193  #define XINPUT_DEVSUBTYPE_WHEEL 0x02
194 #endif
195 #ifndef XINPUT_DEVSUBTYPE_ARCADE_STICK
196  #define XINPUT_DEVSUBTYPE_ARCADE_STICK 0x03
197 #endif
198 #ifndef XINPUT_DEVSUBTYPE_FLIGHT_STICK
199  #define XINPUT_DEVSUBTYPE_FLIGHT_STICK 0x04
200 #endif
201 #ifndef XINPUT_DEVSUBTYPE_DANCE_PAD
202  #define XINPUT_DEVSUBTYPE_DANCE_PAD 0x05
203 #endif
204 #ifndef XINPUT_DEVSUBTYPE_GUITAR
205  #define XINPUT_DEVSUBTYPE_GUITAR 0x06
206 #endif
207 #ifndef XINPUT_DEVSUBTYPE_DRUM_KIT
208  #define XINPUT_DEVSUBTYPE_DRUM_KIT 0x08
209 #endif
210 #ifndef XINPUT_DEVSUBTYPE_ARCADE_PAD
211  #define XINPUT_DEVSUBTYPE_ARCADE_PAD 0x13
212 #endif
213 #ifndef XUSER_MAX_COUNT
214  #define XUSER_MAX_COUNT 4
215 #endif
216
217 // HACK: Define macros that some dinput.h variants don't
218 #ifndef DIDFT_OPTIONAL
219  #define DIDFT_OPTIONAL 0x80000000
220 #endif
221
222 #define WGL_NUMBER_PIXEL_FORMATS_ARB 0x2000
223 #define WGL_SUPPORT_OPENGL_ARB 0x2010
224 #define WGL_DRAW_TO_WINDOW_ARB 0x2001
225 #define WGL_PIXEL_TYPE_ARB 0x2013
226 #define WGL_TYPE_RGBA_ARB 0x202b
227 #define WGL_ACCELERATION_ARB 0x2003
228 #define WGL_NO_ACCELERATION_ARB 0x2025
229 #define WGL_RED_BITS_ARB 0x2015
230 #define WGL_RED_SHIFT_ARB 0x2016
231 #define WGL_GREEN_BITS_ARB 0x2017
232 #define WGL_GREEN_SHIFT_ARB 0x2018
233 #define WGL_BLUE_BITS_ARB 0x2019
234 #define WGL_BLUE_SHIFT_ARB 0x201a
235 #define WGL_ALPHA_BITS_ARB 0x201b
236 #define WGL_ALPHA_SHIFT_ARB 0x201c
237 #define WGL_ACCUM_BITS_ARB 0x201d
238 #define WGL_ACCUM_RED_BITS_ARB 0x201e
239 #define WGL_ACCUM_GREEN_BITS_ARB 0x201f
240 #define WGL_ACCUM_BLUE_BITS_ARB 0x2020
241 #define WGL_ACCUM_ALPHA_BITS_ARB 0x2021
242 #define WGL_DEPTH_BITS_ARB 0x2022
243 #define WGL_STENCIL_BITS_ARB 0x2023
244 #define WGL_AUX_BUFFERS_ARB 0x2024
245 #define WGL_STEREO_ARB 0x2012
246 #define WGL_DOUBLE_BUFFER_ARB 0x2011
247 #define WGL_SAMPLES_ARB 0x2042
248 #define WGL_FRAMEBUFFER_SRGB_CAPABLE_ARB 0x20a9
249 #define WGL_CONTEXT_DEBUG_BIT_ARB 0x00000001
250 #define WGL_CONTEXT_FORWARD_COMPATIBLE_BIT_ARB 0x00000002
251 #define WGL_CONTEXT_PROFILE_MASK_ARB 0x9126
252 #define WGL_CONTEXT_CORE_PROFILE_BIT_ARB 0x00000001
253 #define WGL_CONTEXT_COMPATIBILITY_PROFILE_BIT_ARB 0x00000002
254 #define WGL_CONTEXT_MAJOR_VERSION_ARB 0x2091
255 #define WGL_CONTEXT_MINOR_VERSION_ARB 0x2092
256 #define WGL_CONTEXT_FLAGS_ARB 0x2094
257 #define WGL_CONTEXT_ES2_PROFILE_BIT_EXT 0x00000004
258 #define WGL_CONTEXT_ROBUST_ACCESS_BIT_ARB 0x00000004
259 #define WGL_LOSE_CONTEXT_ON_RESET_ARB 0x8252
260 #define WGL_CONTEXT_RESET_NOTIFICATION_STRATEGY_ARB 0x8256
261 #define WGL_NO_RESET_NOTIFICATION_ARB 0x8261
262 #define WGL_CONTEXT_RELEASE_BEHAVIOR_ARB 0x2097
263 #define WGL_CONTEXT_RELEASE_BEHAVIOR_NONE_ARB 0
264 #define WGL_CONTEXT_RELEASE_BEHAVIOR_FLUSH_ARB 0x2098
265 #define WGL_CONTEXT_OPENGL_NO_ERROR_ARB 0x31b3
266 #define WGL_COLORSPACE_EXT 0x309d
267 #define WGL_COLORSPACE_SRGB_EXT 0x3089
268
269 #define ERROR_INVALID_VERSION_ARB 0x2095
270 #define ERROR_INVALID_PROFILE_ARB 0x2096
271 #define ERROR_INCOMPATIBLE_DEVICE_CONTEXTS_ARB 0x2054
272
```

```
273 // xinput.dll function pointer typedefs
274 typedef DWORD (WINAPI * PFN_XInputGetCapabilities)(DWORD,DWORD,XINPUT_CAPABILITIES*);
275 typedef DWORD (WINAPI * PFN_XInputGetState)(DWORD,XINPUT_STATE*);
276 #define XInputGetCapabilities _glfw.win32.xinput.GetCapabilities
277 #define XInputGetState _glfw.win32.xinput.GetState
278
279 // dinput8.dll function pointer typedefs
280 typedef HRESULT (WINAPI * PFN_DirectInput8Create)(HINSTANCE,DWORD,REFIID,LPVOID*,LPUNKNOWN);
281 #define DirectInput8Create _glfw.win32.dinput8.Create
282
283 // user32.dll function pointer typedefs
284 typedef BOOL (WINAPI * PFN_SetProcessDPIAware)(void);
285 typedef BOOL (WINAPI * PFN_ChangeWindowMessageFilterEx)(HWND,UINT,DWORD,CHANGEFILTERSTRUCT*);
286 typedef BOOL (WINAPI * PFN_EnableNonClientDpiScaling)(HWND);
287 typedef BOOL (WINAPI * PFN_SetProcessDpiAwarenessContext)(HANDLE);
288 typedef UINT (WINAPI * PFN_GetDpiForWindow)(HWND);
289 typedef BOOL (WINAPI * PFN_AdjustWindowRectExForDpi)(LPRECT,DWORD,BOOL,DWORD,UINT);
290 typedef int (WINAPI * PFN_GetSystemMetricsForDpi)(int,UINT);
291 #define SetProcessDPIAware _glfw.win32.user32.SetProcessDPIAware_
292 #define ChangeWindowMessageFilterEx _glfw.win32.user32.ChangeWindowMessageFilterEx_
293 #define EnableNonClientDpiScaling _glfw.win32.user32.EnableNonClientDpiScaling_
294 #define SetProcessDpiAwarenessContext _glfw.win32.user32.SetProcessDpiAwarenessContext_
295 #define GetDpiForWindow _glfw.win32.user32.GetDpiForWindow_
296 #define AdjustWindowRectExForDpi _glfw.win32.user32.AdjustWindowRectExForDpi_
297 #define GetSystemMetricsForDpi _glfw.win32.user32.GetSystemMetricsForDpi_
298
299 // dwmapi.dll function pointer typedefs
300 typedef HRESULT (WINAPI * PFN_DwmIsCompositionEnabled)(BOOL*);
301 typedef HRESULT (WINAPI * PFN_DwmFlush)(VOID);
302 typedef HRESULT(WINAPI * PFN_DwmEnableBlurBehindWindow)(HWND,const DWM_BLURBEHIND*);
303 typedef HRESULT (WINAPI * PFN_DwmGetColorizationColor)(DWORD*,BOOL*);
304 #define DwmIsCompositionEnabled _glfw.win32.dwmapi.IsCompositionEnabled
305 #define DwmFlush _glfw.win32.dwmapi.Flush
306 #define DwmEnableBlurBehindWindow _glfw.win32.dwmapi.EnableBlurBehindWindow
307 #define DwmGetColorizationColor _glfw.win32.dwmapi.GetColorizationColor
308
309 // shcore.dll function pointer typedefs
310 typedef HRESULT (WINAPI * PFN_SetProcessDpiAwareness)(PROCESS_DPI_AWARENESS);
311 typedef HRESULT (WINAPI * PFN_GetDpiForMonitor)(HMONITOR,MONITOR_DPI_TYPE,UINT*,UINT*);
312 #define SetProcessDpiAwareness _glfw.win32.shcore.SetProcessDpiAwareness_
313 #define GetDpiForMonitor _glfw.win32.shcore.GetDpiForMonitor_
314
315 // ntdll.dll function pointer typedefs
316 typedef LONG (WINAPI * PFN_RtlVerifyVersionInfo)(OSVERSIONINFOEXW*,ULONG,ULONGLONG);
317 #define RtlVerifyVersionInfo _glfw.win32.ntdll.RtlVerifyVersionInfo_
318
319 // WGL extension pointer typedefs
320 typedef BOOL (WINAPI * PFNWGLSWAPINTERVALEXTPROC)(int);
321 typedef BOOL (WINAPI * PFNWGLGETPIXELFORMATATTRIBIVARBPROC)(HDC,int,int,UINT,const int*,int*);
322 typedef const char* (WINAPI * PFNWGLGETEXTENSIONSSTRINGEXTPROC)(void);
323 typedef const char* (WINAPI * PFNWGLGETEXTENSIONSSTRINGARBPROC)(HDC);
324 typedef HGLRC (WINAPI * PFNWGLCREATECONTEXTATTRIBSARBPROC)(HDC,HGLRC,const int*);
325 #define wglSwapIntervalEXT _glfw.wgl.SwapIntervalEXT
326 #define wglGetPixelFormatAttribivARB _glfw.wgl.GetPixelFormatAttribivARB
327 #define wglGetExtensionsStringEXT _glfw.wgl.GetExtensionsStringEXT
328 #define wglGetExtensionsStringARB _glfw.wgl.GetExtensionsStringARB
329 #define wglCreateContextAttribsARB _glfw.wgl.CreateContextAttribsARB
330
331 // opengl32.dll function pointer typedefs
332 typedef HGLRC (WINAPI * PFN_wglCreateContext)(HDC);
333 typedef BOOL (WINAPI * PFN_wglDeleteContext)(HGLRC);
334 typedef PROC (WINAPI * PFN_wglGetProcAddress)(LPCSTR);
335 typedef HDC (WINAPI * PFN_wglGetCurrentDC)(void);
336 typedef HGLRC (WINAPI * PFN_wglGetCurrentContext)(void);
337 typedef BOOL (WINAPI * PFN_wglMakeCurrent)(HDC,HGLRC);
338 typedef BOOL (WINAPI * PFN_wglShareLists)(HGLRC,HGLRC);
339 #define wglCreateContext _glfw.wgl.CreateContext
340 #define wglDeleteContext _glfw.wgl.DeleteContext
341 #define wglGetProcAddress _glfw.wgl.GetProcAddress
342 #define wglGetCurrentDC _glfw.wgl.GetCurrentDC
343 #define wglGetCurrentContext _glfw.wgl.GetCurrentContext
344 #define wglMakeCurrent _glfw.wgl.MakeCurrent
345 #define wglShareLists _glfw.wgl.ShareLists
346
347 typedef VkFlags VkWin32SurfaceCreateFlagsKHR;
348
349 typedef struct VkWin32SurfaceCreateInfoKHR
350 {
351     VkStructureType                 sType;
352     const void*                     pNext;
353     VkWin32SurfaceCreateFlagsKHR    flags;
354     HINSTANCE                       hinstance;
355     HWND                            hwnd;
356 } VkWin32SurfaceCreateInfoKHR;
357
358 typedef VkResult (APIENTRY *PFN_vkCreateWin32SurfaceKHR)(VkInstance,const
       VkWin32SurfaceCreateInfoKHR*,const VkAllocationCallbacks*,VkSurfaceKHR*);
```

```
359 typedef VkBool32 (APIENTRY
       *PFN_vkGetPhysicalDeviceWin32PresentationSupportKHR)(VkPhysicalDevice,uint32_t);
360
361 #if !defined(_GLFW_WNDCLASSNAME)
362  #define _GLFW_WNDCLASSNAME L"GLFW30"
363 #endif
364
365 #define GLFW_WIN32_WINDOW_STATE         _GLFWwindowWin32  win32;
366 #define GLFW_WIN32_LIBRARY_WINDOW_STATE _GLFWlibraryWin32 win32;
367 #define GLFW_WIN32_MONITOR_STATE        _GLFWmonitorWin32 win32;
368 #define GLFW_WIN32_CURSOR_STATE         _GLFWcursorWin32  win32;
369
370 #define GLFW_WGL_CONTEXT_STATE          _GLFWcontextWGL wgl;
371 #define GLFW_WGL_LIBRARY_CONTEXT_STATE  _GLFWlibraryWGL wgl;
372
373
374 // WGL-specific per-context data
375 //
376 typedef struct _GLFWcontextWGL
377 {
378     HDC       dc;
379     HGLRC     handle;
380     int       interval;
381 } _GLFWcontextWGL;
382
383 // WGL-specific global data
384 //
385 typedef struct _GLFWlibraryWGL
386 {
387     HINSTANCE                           instance;
388     PFN_wglCreateContext                CreateContext;
389     PFN_wglDeleteContext                DeleteContext;
390     PFN_wglGetProcAddress               GetProcAddress;
391     PFN_wglGetCurrentDC                 GetCurrentDC;
392     PFN_wglGetCurrentContext            GetCurrentContext;
393     PFN_wglMakeCurrent                  MakeCurrent;
394     PFN_wglShareLists                   ShareLists;
395
396     PFNWGLSWAPINTERVALEXTPROC           SwapIntervalEXT;
397     PFNWGLGETPIXELFORMATATTRIBIVARBPROC GetPixelFormatAttribivARB;
398     PFNWGLGETEXTENSIONSSTRINGEXTPROC    GetExtensionsStringEXT;
399     PFNWGLGETEXTENSIONSSTRINGARBPROC    GetExtensionsStringARB;
400     PFNWGLCREATECONTEXTATTRIBSARBPROC   CreateContextAttribsARB;
401     GLFWbool                            EXT_swap_control;
402     GLFWbool                            EXT_colorspace;
403     GLFWbool                            ARB_multisample;
404     GLFWbool                            ARB_framebuffer_sRGB;
405     GLFWbool                            EXT_framebuffer_sRGB;
406     GLFWbool                            ARB_pixel_format;
407     GLFWbool                            ARB_create_context;
408     GLFWbool                            ARB_create_context_profile;
409     GLFWbool                            EXT_create_context_es2_profile;
410     GLFWbool                            ARB_create_context_robustness;
411     GLFWbool                            ARB_create_context_no_error;
412     GLFWbool                            ARB_context_flush_control;
413 } _GLFWlibraryWGL;
414
415 // Win32-specific per-window data
416 //
417 typedef struct _GLFWwindowWin32
418 {
419     HWND                handle;
420     HICON               bigIcon;
421     HICON               smallIcon;
422
423     GLFWbool            cursorTracked;
424     GLFWbool            frameAction;
425     GLFWbool            iconified;
426     GLFWbool            maximized;
427     // Whether to enable framebuffer transparency on DWM
428     GLFWbool            transparent;
429     GLFWbool            scaleToMonitor;
430     GLFWbool            keymenu;
431
432     // Cached size used to filter out duplicate events
433     int                 width, height;
434
435     // The last received cursor position, regardless of source
436     int                 lastCursorPosX, lastCursorPosY;
437     // The last received high surrogate when decoding pairs of UTF-16 messages
438     WCHAR               highSurrogate;
439 } _GLFWwindowWin32;
440
441 // Win32-specific global data
442 //
443 typedef struct _GLFWlibraryWin32
444 {
```

```
445     HINSTANCE           instance;
446     HWND                helperWindowHandle;
447     HDEVNOTIFY          deviceNotificationHandle;
448     int                 acquiredMonitorCount;
449     char*               clipboardString;
450     short int           keycodes[512];
451     short int           scancodes[GLFW_KEY_LAST + 1];
452     char                keynames[GLFW_KEY_LAST + 1][5];
453     // Where to place the cursor when re-enabled
454     double              restoreCursorPosX, restoreCursorPosY;
455     // The window whose disabled cursor mode is active
456     _GLFWwindow*        disabledCursorWindow;
457     RAWINPUT*           rawInput;
458     int                 rawInputSize;
459     UINT                mouseTrailSize;
460
461     struct {
462         HINSTANCE                       instance;
463         PFN_DirectInput8Create          Create;
464         IDirectInput8W*                 api;
465     } dinput8;
466
467     struct {
468         HINSTANCE                       instance;
469         PFN_XInputGetCapabilities       GetCapabilities;
470         PFN_XInputGetState              GetState;
471     } xinput;
472
473     struct {
474         HINSTANCE                       instance;
475         PFN_SetProcessDPIAware          SetProcessDPIAware_;
476         PFN_ChangeWindowMessageFilterEx ChangeWindowMessageFilterEx_;
477         PFN_EnableNonClientDpiScaling   EnableNonClientDpiScaling_;
478         PFN_SetProcessDpiAwarenessContext SetProcessDpiAwarenessContext_;
479         PFN_GetDpiForWindow             GetDpiForWindow_;
480         PFN_AdjustWindowRectExForDpi    AdjustWindowRectExForDpi_;
481         PFN_GetSystemMetricsForDpi      GetSystemMetricsForDpi_;
482     } user32;
483
484     struct {
485         HINSTANCE                       instance;
486         PFN_DwmIsCompositionEnabled     IsCompositionEnabled;
487         PFN_DwmFlush                    Flush;
488         PFN_DwmEnableBlurBehindWindow   EnableBlurBehindWindow;
489         PFN_DwmGetColorizationColor     GetColorizationColor;
490     } dwmapi;
491
492     struct {
493         HINSTANCE                       instance;
494         PFN_SetProcessDpiAwareness      SetProcessDpiAwareness_;
495         PFN_GetDpiForMonitor            GetDpiForMonitor_;
496     } shcore;
497
498     struct {
499         HINSTANCE                       instance;
500         PFN_RtlVerifyVersionInfo        RtlVerifyVersionInfo_;
501     } ntdll;
502 } _GLFWlibraryWin32;
503
504 // Win32-specific per-monitor data
505 //
506 typedef struct _GLFWmonitorWin32
507 {
508     HMONITOR            handle;
509     // This size matches the static size of DISPLAY_DEVICE.DeviceName
510     WCHAR               adapterName[32];
511     WCHAR               displayName[32];
512     char                publicAdapterName[32];
513     char                publicDisplayName[32];
514     GLFWbool            modesPruned;
515     GLFWbool            modeChanged;
516 } _GLFWmonitorWin32;
517
518 // Win32-specific per-cursor data
519 //
520 typedef struct _GLFWcursorWin32
521 {
522     HCURSOR             handle;
523 } _GLFWcursorWin32;
524
525
526 GLFWbool _glfwConnectWin32(int platformID, _GLFWplatform* platform);
527 int _glfwInitWin32(void);
528 void _glfwTerminateWin32(void);
529
530 GLFWbool _glfwRegisterWindowClassWin32(void);
531 void _glfwUnregisterWindowClassWin32(void);
```

```
532
533 WCHAR* _glfwCreateWideStringFromUTF8Win32(const char* source);
534 char* _glfwCreateUTF8FromWideStringWin32(const WCHAR* source);
535 BOOL _glfwIsWindowsVersionOrGreaterWin32(WORD major, WORD minor, WORD sp);
536 BOOL _glfwIsWindows10BuildOrGreaterWin32(WORD build);
537 void _glfwInputErrorWin32(int error, const char* description);
538 void _glfwUpdateKeyNamesWin32(void);
539
540 void _glfwPollMonitorsWin32(void);
541 void _glfwSetVideoModeWin32(_GLFWmonitor* monitor, const GLFWvidmode* desired);
542 void _glfwRestoreVideoModeWin32(_GLFWmonitor* monitor);
543 void _glfwGetHMONITORContentScaleWin32(HMONITOR handle, float* xscale, float* yscale);
544
545 int _glfwCreateWindowWin32(_GLFWwindow* window, const _GLFWwndconfig* wndconfig, const _GLFWctxconfig*
       ctxconfig, const _GLFWfbconfig* fbconfig);
546 void _glfwDestroyWindowWin32(_GLFWwindow* window);
547 void _glfwSetWindowTitleWin32(_GLFWwindow* window, const char* title);
548 void _glfwSetWindowIconWin32(_GLFWwindow* window, int count, const GLFWimage* images);
549 void _glfwGetWindowPosWin32(_GLFWwindow* window, int* xpos, int* ypos);
550 void _glfwSetWindowPosWin32(_GLFWwindow* window, int xpos, int ypos);
551 void _glfwGetWindowSizeWin32(_GLFWwindow* window, int* width, int* height);
552 void _glfwSetWindowSizeWin32(_GLFWwindow* window, int width, int height);
553 void _glfwSetWindowSizeLimitsWin32(_GLFWwindow* window, int minwidth, int minheight, int maxwidth, int
       maxheight);
554 void _glfwSetWindowAspectRatioWin32(_GLFWwindow* window, int numer, int denom);
555 void _glfwGetFramebufferSizeWin32(_GLFWwindow* window, int* width, int* height);
556 void _glfwGetWindowFrameSizeWin32(_GLFWwindow* window, int* left, int* top, int* right, int* bottom);
557 void _glfwGetWindowContentScaleWin32(_GLFWwindow* window, float* xscale, float* yscale);
558 void _glfwIconifyWindowWin32(_GLFWwindow* window);
559 void _glfwRestoreWindowWin32(_GLFWwindow* window);
560 void _glfwMaximizeWindowWin32(_GLFWwindow* window);
561 void _glfwShowWindowWin32(_GLFWwindow* window);
562 void _glfwHideWindowWin32(_GLFWwindow* window);
563 void _glfwRequestWindowAttentionWin32(_GLFWwindow* window);
564 void _glfwFocusWindowWin32(_GLFWwindow* window);
565 void _glfwSetWindowMonitorWin32(_GLFWwindow* window, _GLFWmonitor* monitor, int xpos, int ypos, int
       width, int height, int refreshRate);
566 int _glfwWindowFocusedWin32(_GLFWwindow* window);
567 int _glfwWindowIconifiedWin32(_GLFWwindow* window);
568 int _glfwWindowVisibleWin32(_GLFWwindow* window);
569 int _glfwWindowMaximizedWin32(_GLFWwindow* window);
570 int _glfwWindowHoveredWin32(_GLFWwindow* window);
571 int _glfwFramebufferTransparentWin32(_GLFWwindow* window);
572 void _glfwSetWindowResizableWin32(_GLFWwindow* window, GLFWbool enabled);
573 void _glfwSetWindowDecoratedWin32(_GLFWwindow* window, GLFWbool enabled);
574 void _glfwSetWindowFloatingWin32(_GLFWwindow* window, GLFWbool enabled);
575 void _glfwSetWindowMousePassthroughWin32(_GLFWwindow* window, GLFWbool enabled);
576 float _glfwGetWindowOpacityWin32(_GLFWwindow* window);
577 void _glfwSetWindowOpacityWin32(_GLFWwindow* window, float opacity);
578
579 void _glfwSetRawMouseMotionWin32(_GLFWwindow *window, GLFWbool enabled);
580 GLFWbool _glfwRawMouseMotionSupportedWin32(void);
581
582 void _glfwPollEventsWin32(void);
583 void _glfwWaitEventsWin32(void);
584 void _glfwWaitEventsTimeoutWin32(double timeout);
585 void _glfwPostEmptyEventWin32(void);
586
587 void _glfwGetCursorPosWin32(_GLFWwindow* window, double* xpos, double* ypos);
588 void _glfwSetCursorPosWin32(_GLFWwindow* window, double xpos, double ypos);
589 void _glfwSetCursorModeWin32(_GLFWwindow* window, int mode);
590 const char* _glfwGetScancodeNameWin32(int scancode);
591 int _glfwGetKeyScancodeWin32(int key);
592 int _glfwCreateCursorWin32(_GLFWcursor* cursor, const GLFWimage* image, int xhot, int yhot);
593 int _glfwCreateStandardCursorWin32(_GLFWcursor* cursor, int shape);
594 void _glfwDestroyCursorWin32(_GLFWcursor* cursor);
595 void _glfwSetCursorWin32(_GLFWwindow* window, _GLFWcursor* cursor);
596 void _glfwSetClipboardStringWin32(const char* string);
597 const char* _glfwGetClipboardStringWin32(void);
598
599 EGLenum _glfwGetEGLPlatformWin32(EGLint** attribs);
600 EGLNativeDisplayType _glfwGetEGLNativeDisplayWin32(void);
601 EGLNativeWindowType _glfwGetEGLNativeWindowWin32(_GLFWwindow* window);
602
603 void _glfwGetRequiredInstanceExtensionsWin32(char** extensions);
604 int _glfwGetPhysicalDevicePresentationSupportWin32(VkInstance instance, VkPhysicalDevice device,
       uint32_t queuefamily);
605 VkResult _glfwCreateWindowSurfaceWin32(VkInstance instance, _GLFWwindow* window, const
       VkAllocationCallbacks* allocator, VkSurfaceKHR* surface);
606
607 void _glfwFreeMonitorWin32(_GLFWmonitor* monitor);
608 void _glfwGetMonitorPosWin32(_GLFWmonitor* monitor, int* xpos, int* ypos);
609 void _glfwGetMonitorContentScaleWin32(_GLFWmonitor* monitor, float* xscale, float* yscale);
610 void _glfwGetMonitorWorkareaWin32(_GLFWmonitor* monitor, int* xpos, int* ypos, int* width, int* height);
611 GLFWvidmode* _glfwGetVideoModesWin32(_GLFWmonitor* monitor, int* count);
612 void _glfwGetVideoModeWin32(_GLFWmonitor* monitor, GLFWvidmode* mode);
613 GLFWbool _glfwGetGammaRampWin32(_GLFWmonitor* monitor, GLFWgammaramp* ramp);
```

```
614 void _glfwSetGammaRampWin32(_GLFWmonitor* monitor, const GLFWgammaramp* ramp);
615
616 GLFWbool _glfwInitJoysticksWin32(void);
617 void _glfwTerminateJoysticksWin32(void);
618 int _glfwPollJoystickWin32(_GLFWjoystick* js, int mode);
619 const char* _glfwGetMappingNameWin32(void);
620 void _glfwUpdateGamepadGUIDWin32(char* guid);
621
622 GLFWbool _glfwInitWGL(void);
623 void _glfwTerminateWGL(void);
624 GLFWbool _glfwCreateContextWGL(_GLFWwindow* window,
625                                const _GLFWctxconfig* ctxconfig,
626                                const _GLFWfbconfig* fbconfig);
627
```

27.33 win32_thread.h

```
1 //========================================================================
2 // GLFW 3.4 Win32 - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2002-2006 Marcus Geelnard
5 // Copyright (c) 2006-2017 Camilla Löwy <elmindreda@glfw.org>
6 //
7 // This software is provided 'as-is', without any express or implied
8 // warranty. In no event will the authors be held liable for any damages
9 // arising from the use of this software.
10 //
11 // Permission is granted to anyone to use this software for any purpose,
12 // including commercial applications, and to alter it and redistribute it
13 // freely, subject to the following restrictions:
14 //
15 // 1. The origin of this software must not be misrepresented; you must not
16 //    claim that you wrote the original software. If you use this software
17 //    in a product, an acknowledgment in the product documentation would
18 //    be appreciated but is not required.
19 //
20 // 2. Altered source versions must be plainly marked as such, and must not
21 //    be misrepresented as being the original software.
22 //
23 // 3. This notice may not be removed or altered from any source
24 //    distribution.
25 //
26 //========================================================================
27
28 #include <windows.h>
29
30 #define GLFW_WIN32_TLS_STATE            _GLFWtlsWin32     win32;
31 #define GLFW_WIN32_MUTEX_STATE          _GLFWmutexWin32   win32;
32
33 // Win32-specific thread local storage data
34 //
35 typedef struct _GLFWtlsWin32
36 {
37     GLFWbool            allocated;
38     DWORD               index;
39 } _GLFWtlsWin32;
40
41 // Win32-specific mutex data
42 //
43 typedef struct _GLFWmutexWin32
44 {
45     GLFWbool            allocated;
46     CRITICAL_SECTION    section;
47 } _GLFWmutexWin32;
48
```

27.34 win32_time.h

```
1 //========================================================================
2 // GLFW 3.4 Win32 - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2002-2006 Marcus Geelnard
5 // Copyright (c) 2006-2017 Camilla Löwy <elmindreda@glfw.org>
6 //
7 // This software is provided 'as-is', without any express or implied
8 // warranty. In no event will the authors be held liable for any damages
9 // arising from the use of this software.
10 //
11 // Permission is granted to anyone to use this software for any purpose,
```

```
12 // including commercial applications, and to alter it and redistribute it
13 // freely, subject to the following restrictions:
14 //
15 // 1. The origin of this software must not be misrepresented; you must not
16 //    claim that you wrote the original software. If you use this software
17 //    in a product, an acknowledgment in the product documentation would
18 //    be appreciated but is not required.
19 //
20 // 2. Altered source versions must be plainly marked as such, and must not
21 //    be misrepresented as being the original software.
22 //
23 // 3. This notice may not be removed or altered from any source
24 //    distribution.
25 //
26 //========================================================================
27
28 #include <windows.h>
29
30 #define GLFW_WIN32_LIBRARY_TIMER_STATE  _GLFWtimerWin32   win32;
31
32 // Win32-specific global timer data
33 //
34 typedef struct _GLFWtimerWin32
35 {
36     uint64_t            frequency;
37 } _GLFWtimerWin32;
38
```

27.35 wl_platform.h

```
1 //========================================================================
2 // GLFW 3.4 Wayland - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2014 Jonas Ådahl <jadahl@gmail.com>
5 //
6 // This software is provided 'as-is', without any express or implied
7 // warranty. In no event will the authors be held liable for any damages
8 // arising from the use of this software.
9 //
10 // Permission is granted to anyone to use this software for any purpose,
11 // including commercial applications, and to alter it and redistribute it
12 // freely, subject to the following restrictions:
13 //
14 // 1. The origin of this software must not be misrepresented; you must not
15 //    claim that you wrote the original software. If you use this software
16 //    in a product, an acknowledgment in the product documentation would
17 //    be appreciated but is not required.
18 //
19 // 2. Altered source versions must be plainly marked as such, and must not
20 //    be misrepresented as being the original software.
21 //
22 // 3. This notice may not be removed or altered from any source
23 //    distribution.
24 //
25 //========================================================================
26
27 #include <wayland-client-core.h>
28 #include <xkbcommon/xkbcommon.h>
29 #include <xkbcommon/xkbcommon-compose.h>
30
31 typedef VkFlags VkWaylandSurfaceCreateFlagsKHR;
32
33 typedef struct VkWaylandSurfaceCreateInfoKHR
34 {
35     VkStructureType                 sType;
36     const void*                     pNext;
37     VkWaylandSurfaceCreateFlagsKHR  flags;
38     struct wl_display*              display;
39     struct wl_surface*              surface;
40 } VkWaylandSurfaceCreateInfoKHR;
41
42 typedef VkResult (APIENTRY *PFN_vkCreateWaylandSurfaceKHR)(VkInstance,const
      VkWaylandSurfaceCreateInfoKHR*,const VkAllocationCallbacks*,VkSurfaceKHR*);
43 typedef VkBool32 (APIENTRY
      *PFN_vkGetPhysicalDeviceWaylandPresentationSupportKHR)(VkPhysicalDevice,uint32_t,struct wl_display*);
44
45 #include "xkb_unicode.h"
46 #include "posix_poll.h"
47
48 typedef int (* PFN_wl_display_flush)(struct wl_display *display);
49 typedef void (* PFN_wl_display_cancel_read)(struct wl_display *display);
50 typedef int (* PFN_wl_display_dispatch_pending)(struct wl_display *display);
51 typedef int (* PFN_wl_display_read_events)(struct wl_display *display);
```

```
52 typedef struct wl_display* (* PFN_wl_display_connect)(const char*);
53 typedef void (* PFN_wl_display_disconnect)(struct wl_display*);
54 typedef int (* PFN_wl_display_roundtrip)(struct wl_display*);
55 typedef int (* PFN_wl_display_get_fd)(struct wl_display*);
56 typedef int (* PFN_wl_display_prepare_read)(struct wl_display*);
57 typedef void (* PFN_wl_proxy_marshal)(struct wl_proxy*,uint32_t,...);
58 typedef int (* PFN_wl_proxy_add_listener)(struct wl_proxy*,void(**)(void),void*);
59 typedef void (* PFN_wl_proxy_destroy)(struct wl_proxy*);
60 typedef struct wl_proxy* (* PFN_wl_proxy_marshal_constructor)(struct wl_proxy*,uint32_t,const struct
       wl_interface*,...);
61 typedef struct wl_proxy* (* PFN_wl_proxy_marshal_constructor_versioned)(struct wl_proxy*,uint32_t,const
       struct wl_interface*,uint32_t,...);
62 typedef void* (* PFN_wl_proxy_get_user_data)(struct wl_proxy*);
63 typedef void (* PFN_wl_proxy_set_user_data)(struct wl_proxy*,void*);
64 typedef uint32_t (* PFN_wl_proxy_get_version)(struct wl_proxy*);
65 typedef struct wl_proxy* (* PFN_wl_proxy_marshal_flags)(struct wl_proxy*,uint32_t,const struct
       wl_interface*,uint32_t,uint32_t,...);
66 #define wl_display_flush _glfw.wl.client.display_flush
67 #define wl_display_cancel_read _glfw.wl.client.display_cancel_read
68 #define wl_display_dispatch_pending _glfw.wl.client.display_dispatch_pending
69 #define wl_display_read_events _glfw.wl.client.display_read_events
70 #define wl_display_disconnect _glfw.wl.client.display_disconnect
71 #define wl_display_roundtrip _glfw.wl.client.display_roundtrip
72 #define wl_display_get_fd _glfw.wl.client.display_get_fd
73 #define wl_display_prepare_read _glfw.wl.client.display_prepare_read
74 #define wl_proxy_marshal _glfw.wl.client.proxy_marshal
75 #define wl_proxy_add_listener _glfw.wl.client.proxy_add_listener
76 #define wl_proxy_destroy _glfw.wl.client.proxy_destroy
77 #define wl_proxy_marshal_constructor _glfw.wl.client.proxy_marshal_constructor
78 #define wl_proxy_marshal_constructor_versioned _glfw.wl.client.proxy_marshal_constructor_versioned
79 #define wl_proxy_get_user_data _glfw.wl.client.proxy_get_user_data
80 #define wl_proxy_set_user_data _glfw.wl.client.proxy_set_user_data
81 #define wl_proxy_get_version _glfw.wl.client.proxy_get_version
82 #define wl_proxy_marshal_flags _glfw.wl.client.proxy_marshal_flags
83
84 struct wl_shm;
85
86 #define wl_display_interface _glfw_wl_display_interface
87 #define wl_subcompositor_interface _glfw_wl_subcompositor_interface
88 #define wl_compositor_interface _glfw_wl_compositor_interface
89 #define wl_shm_interface _glfw_wl_shm_interface
90 #define wl_data_device_manager_interface _glfw_wl_data_device_manager_interface
91 #define wl_shell_interface _glfw_wl_shell_interface
92 #define wl_buffer_interface _glfw_wl_buffer_interface
93 #define wl_callback_interface _glfw_wl_callback_interface
94 #define wl_data_device_interface _glfw_wl_data_device_interface
95 #define wl_data_offer_interface _glfw_wl_data_offer_interface
96 #define wl_data_source_interface _glfw_wl_data_source_interface
97 #define wl_keyboard_interface _glfw_wl_keyboard_interface
98 #define wl_output_interface _glfw_wl_output_interface
99 #define wl_pointer_interface _glfw_wl_pointer_interface
100 #define wl_region_interface _glfw_wl_region_interface
101 #define wl_registry_interface _glfw_wl_registry_interface
102 #define wl_seat_interface _glfw_wl_seat_interface
103 #define wl_shell_surface_interface _glfw_wl_shell_surface_interface
104 #define wl_shm_pool_interface _glfw_wl_shm_pool_interface
105 #define wl_subsurface_interface _glfw_wl_subsurface_interface
106 #define wl_surface_interface _glfw_wl_surface_interface
107 #define wl_touch_interface _glfw_wl_touch_interface
108 #define zwp_idle_inhibitor_v1_interface _glfw_zwp_idle_inhibitor_v1_interface
109 #define zwp_idle_inhibit_manager_v1_interface _glfw_zwp_idle_inhibit_manager_v1_interface
110 #define zwp_confined_pointer_v1_interface _glfw_zwp_confined_pointer_v1_interface
111 #define zwp_locked_pointer_v1_interface _glfw_zwp_locked_pointer_v1_interface
112 #define zwp_pointer_constraints_v1_interface _glfw_zwp_pointer_constraints_v1_interface
113 #define zwp_relative_pointer_v1_interface _glfw_zwp_relative_pointer_v1_interface
114 #define zwp_relative_pointer_manager_v1_interface _glfw_zwp_relative_pointer_manager_v1_interface
115 #define wp_viewport_interface _glfw_wp_viewport_interface
116 #define wp_viewporter_interface _glfw_wp_viewporter_interface
117 #define xdg_toplevel_interface _glfw_xdg_toplevel_interface
118 #define zxdg_toplevel_decoration_v1_interface _glfw_zxdg_toplevel_decoration_v1_interface
119 #define zxdg_decoration_manager_v1_interface _glfw_zxdg_decoration_manager_v1_interface
120 #define xdg_popup_interface _glfw_xdg_popup_interface
121 #define xdg_positioner_interface _glfw_xdg_positioner_interface
122 #define xdg_surface_interface _glfw_xdg_surface_interface
123 #define xdg_toplevel_interface _glfw_xdg_toplevel_interface
124 #define xdg_wm_base_interface _glfw_xdg_wm_base_interface
125
126 #define GLFW_WAYLAND_WINDOW_STATE         _GLFWwindowWayland  wl;
127 #define GLFW_WAYLAND_LIBRARY_WINDOW_STATE _GLFWlibraryWayland wl;
128 #define GLFW_WAYLAND_MONITOR_STATE        _GLFWmonitorWayland wl;
129 #define GLFW_WAYLAND_CURSOR_STATE         _GLFWcursorWayland  wl;
130
131 struct wl_cursor_image {
132     uint32_t width;
133     uint32_t height;
134     uint32_t hotspot_x;
135     uint32_t hotspot_y;
```

```
136     uint32_t delay;
137 };
138 struct wl_cursor {
139     unsigned int image_count;
140     struct wl_cursor_image** images;
141     char* name;
142 };
143 typedef struct wl_cursor_theme* (* PFN_wl_cursor_theme_load)(const char*, int, struct wl_shm*);
144 typedef void (* PFN_wl_cursor_theme_destroy)(struct wl_cursor_theme*);
145 typedef struct wl_cursor* (* PFN_wl_cursor_theme_get_cursor)(struct wl_cursor_theme*, const char*);
146 typedef struct wl_buffer* (* PFN_wl_cursor_image_get_buffer)(struct wl_cursor_image*);
147 #define wl_cursor_theme_load _glfw.wl.cursor.theme_load
148 #define wl_cursor_theme_destroy _glfw.wl.cursor.theme_destroy
149 #define wl_cursor_theme_get_cursor _glfw.wl.cursor.theme_get_cursor
150 #define wl_cursor_image_get_buffer _glfw.wl.cursor.image_get_buffer
151
152 typedef struct wl_egl_window* (* PFN_wl_egl_window_create)(struct wl_surface*, int, int);
153 typedef void (* PFN_wl_egl_window_destroy)(struct wl_egl_window*);
154 typedef void (* PFN_wl_egl_window_resize)(struct wl_egl_window*, int, int, int, int);
155 #define wl_egl_window_create _glfw.wl.egl.window_create
156 #define wl_egl_window_destroy _glfw.wl.egl.window_destroy
157 #define wl_egl_window_resize _glfw.wl.egl.window_resize
158
159 typedef struct xkb_context* (* PFN_xkb_context_new)(enum xkb_context_flags);
160 typedef void (* PFN_xkb_context_unref)(struct xkb_context*);
161 typedef struct xkb_keymap* (* PFN_xkb_keymap_new_from_string)(struct xkb_context*, const char*, enum
        xkb_keymap_format, enum xkb_keymap_compile_flags);
162 typedef void (* PFN_xkb_keymap_unref)(struct xkb_keymap*);
163 typedef xkb_mod_index_t (* PFN_xkb_keymap_mod_get_index)(struct xkb_keymap*, const char*);
164 typedef int (* PFN_xkb_keymap_key_repeats)(struct xkb_keymap*, xkb_keycode_t);
165 typedef int (* PFN_xkb_keymap_key_get_syms_by_level)(struct
        xkb_keymap*,xkb_keycode_t,xkb_layout_index_t,xkb_level_index_t,const xkb_keysym_t**);
166 typedef struct xkb_state* (* PFN_xkb_state_new)(struct xkb_keymap*);
167 typedef void (* PFN_xkb_state_unref)(struct xkb_state*);
168 typedef int (* PFN_xkb_state_key_get_syms)(struct xkb_state*, xkb_keycode_t, const xkb_keysym_t**);
169 typedef enum xkb_state_component (* PFN_xkb_state_update_mask)(struct xkb_state*, xkb_mod_mask_t,
        xkb_mod_mask_t, xkb_mod_mask_t, xkb_layout_index_t, xkb_layout_index_t, xkb_layout_index_t);
170 typedef xkb_mod_mask_t (* PFN_xkb_state_serialize_mods)(struct xkb_state*, enum xkb_state_component);
171 typedef xkb_layout_index_t (* PFN_xkb_state_key_get_layout)(struct xkb_state*,xkb_keycode_t);
172 #define xkb_context_new _glfw.wl.xkb.context_new
173 #define xkb_context_unref _glfw.wl.xkb.context_unref
174 #define xkb_keymap_new_from_string _glfw.wl.xkb.keymap_new_from_string
175 #define xkb_keymap_unref _glfw.wl.xkb.keymap_unref
176 #define xkb_keymap_mod_get_index _glfw.wl.xkb.keymap_mod_get_index
177 #define xkb_keymap_key_repeats _glfw.wl.xkb.keymap_key_repeats
178 #define xkb_keymap_key_get_syms_by_level _glfw.wl.xkb.keymap_key_get_syms_by_level
179 #define xkb_state_new _glfw.wl.xkb.state_new
180 #define xkb_state_unref _glfw.wl.xkb.state_unref
181 #define xkb_state_key_get_syms _glfw.wl.xkb.state_key_get_syms
182 #define xkb_state_update_mask _glfw.wl.xkb.state_update_mask
183 #define xkb_state_serialize_mods _glfw.wl.xkb.state_serialize_mods
184 #define xkb_state_key_get_layout _glfw.wl.xkb.state_key_get_layout
185
186 typedef struct xkb_compose_table* (* PFN_xkb_compose_table_new_from_locale)(struct xkb_context*, const
        char*, enum xkb_compose_compile_flags);
187 typedef void (* PFN_xkb_compose_table_unref)(struct xkb_compose_table*);
188 typedef struct xkb_compose_state* (* PFN_xkb_compose_state_new)(struct xkb_compose_table*, enum
        xkb_compose_state_flags);
189 typedef void (* PFN_xkb_compose_state_unref)(struct xkb_compose_state*);
190 typedef enum xkb_compose_feed_result (* PFN_xkb_compose_state_feed)(struct xkb_compose_state*,
        xkb_keysym_t);
191 typedef enum xkb_compose_status (* PFN_xkb_compose_state_get_status)(struct xkb_compose_state*);
192 typedef xkb_keysym_t (* PFN_xkb_compose_state_get_one_sym)(struct xkb_compose_state*);
193 #define xkb_compose_table_new_from_locale _glfw.wl.xkb.compose_table_new_from_locale
194 #define xkb_compose_table_unref _glfw.wl.xkb.compose_table_unref
195 #define xkb_compose_state_new _glfw.wl.xkb.compose_state_new
196 #define xkb_compose_state_unref _glfw.wl.xkb.compose_state_unref
197 #define xkb_compose_state_feed _glfw.wl.xkb.compose_state_feed
198 #define xkb_compose_state_get_status _glfw.wl.xkb.compose_state_get_status
199 #define xkb_compose_state_get_one_sym _glfw.wl.xkb.compose_state_get_one_sym
200
201 #define _GLFW_DECORATION_WIDTH 4
202 #define _GLFW_DECORATION_TOP 24
203 #define _GLFW_DECORATION_VERTICAL (_GLFW_DECORATION_TOP + _GLFW_DECORATION_WIDTH)
204 #define _GLFW_DECORATION_HORIZONTAL (2 * _GLFW_DECORATION_WIDTH)
205
206 typedef enum _GLFWdecorationSideWayland
207 {
208     mainWindow,
209     topDecoration,
210     leftDecoration,
211     rightDecoration,
212     bottomDecoration,
213 } _GLFWdecorationSideWayland;
214
215 typedef struct _GLFWdecorationWayland
216 {
```

```
217     struct wl_surface*          surface;
218     struct wl_subsurface*       subsurface;
219     struct wp_viewport*         viewport;
220 } _GLFWdecorationWayland;
221
222 // Wayland-specific per-window data
223 //
224 typedef struct _GLFWwindowWayland
225 {
226     int                         width, height;
227     GLFWbool                    visible;
228     GLFWbool                    maximized;
229     GLFWbool                    hovered;
230     GLFWbool                    transparent;
231     struct wl_surface*          surface;
232     struct wl_egl_window*       native;
233     struct wl_callback*         callback;
234
235     struct {
236         struct xdg_surface*     surface;
237         struct xdg_toplevel*    toplevel;
238         struct zxdg_toplevel_decoration_v1* decoration;
239     } xdg;
240
241     _GLFWcursor*                currentCursor;
242     double                      cursorPosX, cursorPosY;
243
244     char*                       title;
245
246     // We need to track the monitors the window spans on to calculate the
247     // optimal scaling factor.
248     int                         scale;
249     _GLFWmonitor**              monitors;
250     int                         monitorsCount;
251     int                         monitorsSize;
252
253     struct {
254         struct zwp_relative_pointer_v1*    relativePointer;
255         struct zwp_locked_pointer_v1*      lockedPointer;
256     } pointerLock;
257
258     struct zwp_idle_inhibitor_v1*          idleInhibitor;
259
260     GLFWbool                    wasFullscreen;
261
262     struct {
263         GLFWbool                           serverSide;
264         struct wl_buffer*                  buffer;
265         _GLFWdecorationWayland             top, left, right, bottom;
266         int                                focus;
267     } decorations;
268 } _GLFWwindowWayland;
269
270 // Wayland-specific global data
271 //
272 typedef struct _GLFWlibraryWayland
273 {
274     struct wl_display*          display;
275     struct wl_registry*         registry;
276     struct wl_compositor*       compositor;
277     struct wl_subcompositor*    subcompositor;
278     struct wl_shm*              shm;
279     struct wl_seat*             seat;
280     struct wl_pointer*          pointer;
281     struct wl_keyboard*         keyboard;
282     struct wl_data_device_manager*          dataDeviceManager;
283     struct wl_data_device*      dataDevice;
284     struct wl_data_offer*       dataOffer;
285     struct wl_data_source*      dataSource;
286     struct xdg_wm_base*         wmBase;
287     struct zxdg_decoration_manager_v1*      decorationManager;
288     struct wp_viewporter*       viewporter;
289     struct zwp_relative_pointer_manager_v1* relativePointerManager;
290     struct zwp_pointer_constraints_v1*      pointerConstraints;
291     struct zwp_idle_inhibit_manager_v1*     idleInhibitManager;
292
293     int                         compositorVersion;
294     int                         seatVersion;
295
296     struct wl_cursor_theme*     cursorTheme;
297     struct wl_cursor_theme*     cursorThemeHiDPI;
298     struct wl_surface*          cursorSurface;
299     const char*                 cursorPreviousName;
300     int                         cursorTimerfd;
301     uint32_t                    serial;
302     uint32_t                    pointerEnterSerial;
303
```

```
304     int32_t                     keyboardRepeatRate;
305     int32_t                     keyboardRepeatDelay;
306     int                         keyboardLastKey;
307     int                         keyboardLastScancode;
308     char*                       clipboardString;
309     size_t                      clipboardSize;
310     char*                       clipboardSendString;
311     size_t                      clipboardSendSize;
312     int                         timerfd;
313     short int                   keycodes[256];
314     short int                   scancodes[GLFW_KEY_LAST + 1];
315     char                        keynames[GLFW_KEY_LAST + 1][5];
316
317     struct {
318         void*                   handle;
319         struct xkb_context*     context;
320         struct xkb_keymap*      keymap;
321         struct xkb_state*       state;
322
323         struct xkb_compose_state* composeState;
324
325         xkb_mod_mask_t          controlMask;
326         xkb_mod_mask_t          altMask;
327         xkb_mod_mask_t          shiftMask;
328         xkb_mod_mask_t          superMask;
329         xkb_mod_mask_t          capsLockMask;
330         xkb_mod_mask_t          numLockMask;
331         unsigned int            modifiers;
332
333         PFN_xkb_context_new context_new;
334         PFN_xkb_context_unref context_unref;
335         PFN_xkb_keymap_new_from_string keymap_new_from_string;
336         PFN_xkb_keymap_unref keymap_unref;
337         PFN_xkb_keymap_mod_get_index keymap_mod_get_index;
338         PFN_xkb_keymap_key_repeats keymap_key_repeats;
339         PFN_xkb_keymap_key_get_syms_by_level keymap_key_get_syms_by_level;
340         PFN_xkb_state_new state_new;
341         PFN_xkb_state_unref state_unref;
342         PFN_xkb_state_key_get_syms state_key_get_syms;
343         PFN_xkb_state_update_mask state_update_mask;
344         PFN_xkb_state_serialize_mods state_serialize_mods;
345         PFN_xkb_state_key_get_layout state_key_get_layout;
346
347         PFN_xkb_compose_table_new_from_locale compose_table_new_from_locale;
348         PFN_xkb_compose_table_unref compose_table_unref;
349         PFN_xkb_compose_state_new compose_state_new;
350         PFN_xkb_compose_state_unref compose_state_unref;
351         PFN_xkb_compose_state_feed compose_state_feed;
352         PFN_xkb_compose_state_get_status compose_state_get_status;
353         PFN_xkb_compose_state_get_one_sym compose_state_get_one_sym;
354     } xkb;
355
356     _GLFWwindow*                pointerFocus;
357     _GLFWwindow*                keyboardFocus;
358
359     struct {
360         void*                                       handle;
361         PFN_wl_display_flush                        display_flush;
362         PFN_wl_display_cancel_read                  display_cancel_read;
363         PFN_wl_display_dispatch_pending             display_dispatch_pending;
364         PFN_wl_display_read_events                  display_read_events;
365         PFN_wl_display_disconnect                   display_disconnect;
366         PFN_wl_display_roundtrip                    display_roundtrip;
367         PFN_wl_display_get_fd                       display_get_fd;
368         PFN_wl_display_prepare_read                 display_prepare_read;
369         PFN_wl_proxy_marshal                        proxy_marshal;
370         PFN_wl_proxy_add_listener                   proxy_add_listener;
371         PFN_wl_proxy_destroy                        proxy_destroy;
372         PFN_wl_proxy_marshal_constructor            proxy_marshal_constructor;
373         PFN_wl_proxy_marshal_constructor_versioned  proxy_marshal_constructor_versioned;
374         PFN_wl_proxy_get_user_data                  proxy_get_user_data;
375         PFN_wl_proxy_set_user_data                  proxy_set_user_data;
376         PFN_wl_proxy_get_version                    proxy_get_version;
377         PFN_wl_proxy_marshal_flags                  proxy_marshal_flags;
378     } client;
379
380     struct {
381         void*                   handle;
382
383         PFN_wl_cursor_theme_load theme_load;
384         PFN_wl_cursor_theme_destroy theme_destroy;
385         PFN_wl_cursor_theme_get_cursor theme_get_cursor;
386         PFN_wl_cursor_image_get_buffer image_get_buffer;
387     } cursor;
388
389     struct {
390         void*                   handle;
```

```
391
392         PFN_wl_egl_window_create window_create;
393         PFN_wl_egl_window_destroy window_destroy;
394         PFN_wl_egl_window_resize window_resize;
395     } egl;
396 } _GLFWlibraryWayland;
397
398 // Wayland-specific per-monitor data
399 //
400 typedef struct _GLFWmonitorWayland
401 {
402     struct wl_output*           output;
403     uint32_t                    name;
404     int                         currentMode;
405
406     int                         x;
407     int                         y;
408     int                         scale;
409 } _GLFWmonitorWayland;
410
411 // Wayland-specific per-cursor data
412 //
413 typedef struct _GLFWcursorWayland
414 {
415     struct wl_cursor*           cursor;
416     struct wl_cursor*           cursorHiDPI;
417     struct wl_buffer*           buffer;
418     int                         width, height;
419     int                         xhot, yhot;
420     int                         currentImage;
421 } _GLFWcursorWayland;
422
423 GLFWbool _glfwConnectWayland(int platformID, _GLFWplatform* platform);
424 int _glfwInitWayland(void);
425 void _glfwTerminateWayland(void);
426
427 int _glfwCreateWindowWayland(_GLFWwindow* window, const _GLFWwndconfig* wndconfig, const _GLFWctxconfig*
        ctxconfig, const _GLFWfbconfig* fbconfig);
428 void _glfwDestroyWindowWayland(_GLFWwindow* window);
429 void _glfwSetWindowTitleWayland(_GLFWwindow* window, const char* title);
430 void _glfwSetWindowIconWayland(_GLFWwindow* window, int count, const GLFWimage* images);
431 void _glfwGetWindowPosWayland(_GLFWwindow* window, int* xpos, int* ypos);
432 void _glfwSetWindowPosWayland(_GLFWwindow* window, int xpos, int ypos);
433 void _glfwGetWindowSizeWayland(_GLFWwindow* window, int* width, int* height);
434 void _glfwSetWindowSizeWayland(_GLFWwindow* window, int width, int height);
435 void _glfwSetWindowSizeLimitsWayland(_GLFWwindow* window, int minwidth, int minheight, int maxwidth, int
        maxheight);
436 void _glfwSetWindowAspectRatioWayland(_GLFWwindow* window, int numer, int denom);
437 void _glfwGetFramebufferSizeWayland(_GLFWwindow* window, int* width, int* height);
438 void _glfwGetWindowFrameSizeWayland(_GLFWwindow* window, int* left, int* top, int* right, int* bottom);
439 void _glfwGetWindowContentScaleWayland(_GLFWwindow* window, float* xscale, float* yscale);
440 void _glfwIconifyWindowWayland(_GLFWwindow* window);
441 void _glfwRestoreWindowWayland(_GLFWwindow* window);
442 void _glfwMaximizeWindowWayland(_GLFWwindow* window);
443 void _glfwShowWindowWayland(_GLFWwindow* window);
444 void _glfwHideWindowWayland(_GLFWwindow* window);
445 void _glfwRequestWindowAttentionWayland(_GLFWwindow* window);
446 void _glfwFocusWindowWayland(_GLFWwindow* window);
447 void _glfwSetWindowMonitorWayland(_GLFWwindow* window, _GLFWmonitor* monitor, int xpos, int ypos, int
        width, int height, int refreshRate);
448 int _glfwWindowFocusedWayland(_GLFWwindow* window);
449 int _glfwWindowIconifiedWayland(_GLFWwindow* window);
450 int _glfwWindowVisibleWayland(_GLFWwindow* window);
451 int _glfwWindowMaximizedWayland(_GLFWwindow* window);
452 int _glfwWindowHoveredWayland(_GLFWwindow* window);
453 int _glfwFramebufferTransparentWayland(_GLFWwindow* window);
454 void _glfwSetWindowResizableWayland(_GLFWwindow* window, GLFWbool enabled);
455 void _glfwSetWindowDecoratedWayland(_GLFWwindow* window, GLFWbool enabled);
456 void _glfwSetWindowFloatingWayland(_GLFWwindow* window, GLFWbool enabled);
457 float _glfwGetWindowOpacityWayland(_GLFWwindow* window);
458 void _glfwSetWindowOpacityWayland(_GLFWwindow* window, float opacity);
459 void _glfwSetWindowMousePassthroughWayland(_GLFWwindow* window, GLFWbool enabled);
460
461 void _glfwSetRawMouseMotionWayland(_GLFWwindow *window, GLFWbool enabled);
462 GLFWbool _glfwRawMouseMotionSupportedWayland(void);
463
464 void _glfwPollEventsWayland(void);
465 void _glfwWaitEventsWayland(void);
466 void _glfwWaitEventsTimeoutWayland(double timeout);
467 void _glfwPostEmptyEventWayland(void);
468
469 void _glfwGetCursorPosWayland(_GLFWwindow* window, double* xpos, double* ypos);
470 void _glfwSetCursorPosWayland(_GLFWwindow* window, double xpos, double ypos);
471 void _glfwSetCursorModeWayland(_GLFWwindow* window, int mode);
472 const char* _glfwGetScancodeNameWayland(int scancode);
473 int _glfwGetKeyScancodeWayland(int key);
474 int _glfwCreateCursorWayland(_GLFWcursor* cursor, const GLFWimage* image, int xhot, int yhot);
```

```
475 int _glfwCreateStandardCursorWayland(_GLFWcursor* cursor, int shape);
476 void _glfwDestroyCursorWayland(_GLFWcursor* cursor);
477 void _glfwSetCursorWayland(_GLFWwindow* window, _GLFWcursor* cursor);
478 void _glfwSetClipboardStringWayland(const char* string);
479 const char* _glfwGetClipboardStringWayland(void);
480
481 EGLenum _glfwGetEGLPlatformWayland(EGLint** attribs);
482 EGLNativeDisplayType _glfwGetEGLNativeDisplayWayland(void);
483 EGLNativeWindowType _glfwGetEGLNativeWindowWayland(_GLFWwindow* window);
484
485 void _glfwGetRequiredInstanceExtensionsWayland(char** extensions);
486 int _glfwGetPhysicalDevicePresentationSupportWayland(VkInstance instance, VkPhysicalDevice device,
       uint32_t queuefamily);
487 VkResult _glfwCreateWindowSurfaceWayland(VkInstance instance, _GLFWwindow* window, const
       VkAllocationCallbacks* allocator, VkSurfaceKHR* surface);
488
489 void _glfwFreeMonitorWayland(_GLFWmonitor* monitor);
490 void _glfwGetMonitorPosWayland(_GLFWmonitor* monitor, int* xpos, int* ypos);
491 void _glfwGetMonitorContentScaleWayland(_GLFWmonitor* monitor, float* xscale, float* yscale);
492 void _glfwGetMonitorWorkareaWayland(_GLFWmonitor* monitor, int* xpos, int* ypos, int* width, int*
       height);
493 GLFWvidmode* _glfwGetVideoModesWayland(_GLFWmonitor* monitor, int* count);
494 void _glfwGetVideoModeWayland(_GLFWmonitor* monitor, GLFWvidmode* mode);
495 GLFWbool _glfwGetGammaRampWayland(_GLFWmonitor* monitor, GLFWgammaramp* ramp);
496 void _glfwSetGammaRampWayland(_GLFWmonitor* monitor, const GLFWgammaramp* ramp);
497
498 void _glfwAddOutputWayland(uint32_t name, uint32_t version);
499 GLFWbool _glfwInputTextWayland(_GLFWwindow* window, uint32_t scancode);
500
```

27.36 x11_platform.h

```
1 //========================================================================
2 // GLFW 3.4 X11 - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2002-2006 Marcus Geelnard
5 // Copyright (c) 2006-2019 Camilla Löwy <elmindreda@glfw.org>
6 //
7 // This software is provided 'as-is', without any express or implied
8 // warranty. In no event will the authors be held liable for any damages
9 // arising from the use of this software.
10 //
11 // Permission is granted to anyone to use this software for any purpose,
12 // including commercial applications, and to alter it and redistribute it
13 // freely, subject to the following restrictions:
14 //
15 // 1. The origin of this software must not be misrepresented; you must not
16 //    claim that you wrote the original software. If you use this software
17 //    in a product, an acknowledgment in the product documentation would
18 //    be appreciated but is not required.
19 //
20 // 2. Altered source versions must be plainly marked as such, and must not
21 //    be misrepresented as being the original software.
22 //
23 // 3. This notice may not be removed or altered from any source
24 //    distribution.
25 //
26 //========================================================================
27
28 #include <unistd.h>
29 #include <signal.h>
30 #include <stdint.h>
31
32 #include <X11/Xlib.h>
33 #include <X11/keysym.h>
34 #include <X11/Xatom.h>
35 #include <X11/Xresource.h>
36 #include <X11/Xcursor/Xcursor.h>
37
38 // The XRandR extension provides mode setting and gamma control
39 #include <X11/extensions/Xrandr.h>
40
41 // The Xkb extension provides improved keyboard support
42 #include <X11/XKBlib.h>
43
44 // The Xinerama extension provides legacy monitor indices
45 #include <X11/extensions/Xinerama.h>
46
47 // The XInput extension provides raw mouse motion input
48 #include <X11/extensions/XInput2.h>
49
50 // The Shape extension provides custom window shapes
51 #include <X11/extensions/shape.h>
```

```
52
53 #define GLX_VENDOR 1
54 #define GLX_RGBA_BIT 0x00000001
55 #define GLX_WINDOW_BIT 0x00000001
56 #define GLX_DRAWABLE_TYPE 0x8010
57 #define GLX_RENDER_TYPE 0x8011
58 #define GLX_RGBA_TYPE 0x8014
59 #define GLX_DOUBLEBUFFER 5
60 #define GLX_STEREO 6
61 #define GLX_AUX_BUFFERS 7
62 #define GLX_RED_SIZE 8
63 #define GLX_GREEN_SIZE 9
64 #define GLX_BLUE_SIZE 10
65 #define GLX_ALPHA_SIZE 11
66 #define GLX_DEPTH_SIZE 12
67 #define GLX_STENCIL_SIZE 13
68 #define GLX_ACCUM_RED_SIZE 14
69 #define GLX_ACCUM_GREEN_SIZE 15
70 #define GLX_ACCUM_BLUE_SIZE 16
71 #define GLX_ACCUM_ALPHA_SIZE 17
72 #define GLX_SAMPLES 0x186a1
73 #define GLX_VISUAL_ID 0x800b
74
75 #define GLX_FRAMEBUFFER_SRGB_CAPABLE_ARB 0x20b2
76 #define GLX_CONTEXT_DEBUG_BIT_ARB 0x00000001
77 #define GLX_CONTEXT_COMPATIBILITY_PROFILE_BIT_ARB 0x00000002
78 #define GLX_CONTEXT_CORE_PROFILE_BIT_ARB 0x00000001
79 #define GLX_CONTEXT_PROFILE_MASK_ARB 0x9126
80 #define GLX_CONTEXT_FORWARD_COMPATIBLE_BIT_ARB 0x00000002
81 #define GLX_CONTEXT_MAJOR_VERSION_ARB 0x2091
82 #define GLX_CONTEXT_MINOR_VERSION_ARB 0x2092
83 #define GLX_CONTEXT_FLAGS_ARB 0x2094
84 #define GLX_CONTEXT_ES2_PROFILE_BIT_EXT 0x00000004
85 #define GLX_CONTEXT_ROBUST_ACCESS_BIT_ARB 0x00000004
86 #define GLX_LOSE_CONTEXT_ON_RESET_ARB 0x8252
87 #define GLX_CONTEXT_RESET_NOTIFICATION_STRATEGY_ARB 0x8256
88 #define GLX_NO_RESET_NOTIFICATION_ARB 0x8261
89 #define GLX_CONTEXT_RELEASE_BEHAVIOR_ARB 0x2097
90 #define GLX_CONTEXT_RELEASE_BEHAVIOR_NONE_ARB 0
91 #define GLX_CONTEXT_RELEASE_BEHAVIOR_FLUSH_ARB 0x2098
92 #define GLX_CONTEXT_OPENGL_NO_ERROR_ARB 0x31b3
93
94 typedef XID GLXWindow;
95 typedef XID GLXDrawable;
96 typedef struct __GLXFBConfig* GLXFBConfig;
97 typedef struct __GLXcontext* GLXContext;
98 typedef void (*__GLXextproc)(void);
99
100 typedef XClassHint* (* PFN_XAllocClassHint)(void);
101 typedef XSizeHints* (* PFN_XAllocSizeHints)(void);
102 typedef XWMHints* (* PFN_XAllocWMHints)(void);
103 typedef int (* PFN_XChangeProperty)(Display*,Window,Atom,Atom,int,int,const unsigned char*,int);
104 typedef int (* PFN_XChangeWindowAttributes)(Display*,Window,unsigned long,XSetWindowAttributes*);
105 typedef Bool (* PFN_XCheckIfEvent)(Display*,XEvent*,Bool(*)(Display*,XEvent*,XPointer),XPointer);
106 typedef Bool (* PFN_XCheckTypedWindowEvent)(Display*,Window,int,XEvent*);
107 typedef int (* PFN_XCloseDisplay)(Display*);
108 typedef Status (* PFN_XCloseIM)(XIM);
109 typedef int (* PFN_XConvertSelection)(Display*,Atom,Atom,Atom,Window,Time);
110 typedef Colormap (* PFN_XCreateColormap)(Display*,Window,Visual*,int);
111 typedef Cursor (* PFN_XCreateFontCursor)(Display*,unsigned int);
112 typedef XIC (* PFN_XCreateIC)(XIM,...);
113 typedef Region (* PFN_XCreateRegion)(void);
114 typedef Window (* PFN_XCreateWindow)(Display*,Window,int,int,unsigned int,unsigned int,unsigned
       int,int,unsigned int,Visual*,unsigned long,XSetWindowAttributes*);
115 typedef int (* PFN_XDefineCursor)(Display*,Window,Cursor);
116 typedef int (* PFN_XDeleteContext)(Display*,XID,XContext);
117 typedef int (* PFN_XDeleteProperty)(Display*,Window,Atom);
118 typedef void (* PFN_XDestroyIC)(XIC);
119 typedef int (* PFN_XDestroyRegion)(Region);
120 typedef int (* PFN_XDestroyWindow)(Display*,Window);
121 typedef int (* PFN_XDisplayKeycodes)(Display*,int*,int*);
122 typedef int (* PFN_XEventsQueued)(Display*,int);
123 typedef Bool (* PFN_XFilterEvent)(XEvent*,Window);
124 typedef int (* PFN_XFindContext)(Display*,XID,XContext,XPointer*);
125 typedef int (* PFN_XFlush)(Display*);
126 typedef int (* PFN_XFree)(void*);
127 typedef int (* PFN_XFreeColormap)(Display*,Colormap);
128 typedef int (* PFN_XFreeCursor)(Display*,Cursor);
129 typedef void (* PFN_XFreeEventData)(Display*,XGenericEventCookie*);
130 typedef int (* PFN_XGetErrorText)(Display*,int,char*,int);
131 typedef Bool (* PFN_XGetEventData)(Display*,XGenericEventCookie*);
132 typedef char* (* PFN_XGetICValues)(XIC,...);
133 typedef char* (* PFN_XGetIMValues)(XIM,...);
134 typedef int (* PFN_XGetInputFocus)(Display*,Window*,int*);
135 typedef KeySym* (* PFN_XGetKeyboardMapping)(Display*,KeyCode,int,int*);
136 typedef int (* PFN_XGetScreenSaver)(Display*,int*,int*,int*,int*);
137 typedef Window (* PFN_XGetSelectionOwner)(Display*,Atom);
```

```
138 typedef XVisualInfo* (* PFN_XGetVisualInfo)(Display*,long,XVisualInfo*,int*);
139 typedef Status (* PFN_XGetWMNormalHints)(Display*,Window,XSizeHints*,long*);
140 typedef Status (* PFN_XGetWindowAttributes)(Display*,Window,XWindowAttributes*);
141 typedef int (* PFN_XGetWindowProperty)(Display*,Window,Atom,long,long,Bool,Atom,Atom*,int*,unsigned
      long*,unsigned long*,unsigned char**);
142 typedef int (* PFN_XGrabPointer)(Display*,Window,Bool,unsigned int,int,int,Window,Cursor,Time);
143 typedef Status (* PFN_XIconifyWindow)(Display*,Window,int);
144 typedef Status (* PFN_XInitThreads)(void);
145 typedef Atom (* PFN_XInternAtom)(Display*,const char*,Bool);
146 typedef int (* PFN_XLookupString)(XKeyEvent*,char*,int,KeySym*,XComposeStatus*);
147 typedef int (* PFN_XMapRaised)(Display*,Window);
148 typedef int (* PFN_XMapWindow)(Display*,Window);
149 typedef int (* PFN_XMoveResizeWindow)(Display*,Window,int,int,unsigned int,unsigned int);
150 typedef int (* PFN_XMoveWindow)(Display*,Window,int,int);
151 typedef int (* PFN_XNextEvent)(Display*,XEvent*);
152 typedef Display* (* PFN_XOpenDisplay)(const char*);
153 typedef XIM (* PFN_XOpenIM)(Display*,XrmDatabase*,char*,char*);
154 typedef int (* PFN_XPeekEvent)(Display*,XEvent*);
155 typedef int (* PFN_XPending)(Display*);
156 typedef Bool (* PFN_XQueryExtension)(Display*,const char*,int*,int*,int*);
157 typedef Bool (* PFN_XQueryPointer)(Display*,Window,Window*,Window*,int*,int*,int*,int*,unsigned int*);
158 typedef int (* PFN_XRaiseWindow)(Display*,Window);
159 typedef Bool (* PFN_XRegisterIMInstantiateCallback)(Display*,void*,char*,char*,XIDProc,XPointer);
160 typedef int (* PFN_XResizeWindow)(Display*,Window,unsigned int,unsigned int);
161 typedef char* (* PFN_XResourceManagerString)(Display*);
162 typedef int (* PFN_XSaveContext)(Display*,XID,XContext,const char*);
163 typedef int (* PFN_XSelectInput)(Display*,Window,long);
164 typedef Status (* PFN_XSendEvent)(Display*,Window,Bool,long,XEvent*);
165 typedef int (* PFN_XSetClassHint)(Display*,Window,XClassHint*);
166 typedef XErrorHandler (* PFN_XSetErrorHandler)(XErrorHandler);
167 typedef void (* PFN_XSetICFocus)(XIC);
168 typedef char* (* PFN_XSetIMValues)(XIM,...);
169 typedef int (* PFN_XSetInputFocus)(Display*,Window,int,Time);
170 typedef char* (* PFN_XSetLocaleModifiers)(const char*);
171 typedef int (* PFN_XSetScreenSaver)(Display*,int,int,int,int);
172 typedef int (* PFN_XSetSelectionOwner)(Display*,Atom,Window,Time);
173 typedef int (* PFN_XSetWMHints)(Display*,Window,XWMHints*);
174 typedef void (* PFN_XSetWMNormalHints)(Display*,Window,XSizeHints*);
175 typedef Status (* PFN_XSetWMProtocols)(Display*,Window,Atom*,int);
176 typedef Bool (* PFN_XSupportsLocale)(void);
177 typedef int (* PFN_XSync)(Display*,Bool);
178 typedef Bool (* PFN_XTranslateCoordinates)(Display*,Window,Window,int,int,int*,int*,Window*);
179 typedef int (* PFN_XUndefineCursor)(Display*,Window);
180 typedef int (* PFN_XUngrabPointer)(Display*,Time);
181 typedef int (* PFN_XUnmapWindow)(Display*,Window);
182 typedef void (* PFN_XUnsetICFocus)(XIC);
183 typedef VisualID (* PFN_XVisualIDFromVisual)(Visual*);
184 typedef int (* PFN_XWarpPointer)(Display*,Window,Window,int,int,unsigned int,unsigned int,int,int);
185 typedef void (* PFN_XkbFreeKeyboard)(XkbDescPtr,unsigned int,Bool);
186 typedef void (* PFN_XkbFreeNames)(XkbDescPtr,unsigned int,Bool);
187 typedef XkbDescPtr (* PFN_XkbGetMap)(Display*,unsigned int,unsigned int);
188 typedef Status (* PFN_XkbGetNames)(Display*,unsigned int,XkbDescPtr);
189 typedef Status (* PFN_XkbGetState)(Display*,unsigned int,XkbStatePtr);
190 typedef KeySym (* PFN_XkbKeycodeToKeysym)(Display*,KeyCode,int,int);
191 typedef Bool (* PFN_XkbQueryExtension)(Display*,int*,int*,int*,int*,int*);
192 typedef Bool (* PFN_XkbSelectEventDetails)(Display*,unsigned int,unsigned int,unsigned long,unsigned
      long);
193 typedef Bool (* PFN_XkbSetDetectableAutoRepeat)(Display*,Bool,Bool*);
194 typedef void (* PFN_XrmDestroyDatabase)(XrmDatabase);
195 typedef Bool (* PFN_XrmGetResource)(XrmDatabase,const char*,const char*,char**,XrmValue*);
196 typedef XrmDatabase (* PFN_XrmGetStringDatabase)(const char*);
197 typedef void (* PFN_XrmInitialize)(void);
198 typedef XrmQuark (* PFN_XrmUniqueQuark)(void);
199 typedef Bool (* PFN_XUnregisterIMInstantiateCallback)(Display*,void*,char*,char*,XIDProc,XPointer);
200 typedef int (* PFN_Xutf8LookupString)(XIC,XKeyPressedEvent*,char*,int,KeySym*,Status*);
201 typedef void (* PFN_Xutf8SetWMProperties)(Display*,Window,const char*,const
      char*,char**,int,XSizeHints*,XWMHints*,XClassHint*);
202 #define XAllocClassHint _glfw.x11.xlib.AllocClassHint
203 #define XAllocSizeHints _glfw.x11.xlib.AllocSizeHints
204 #define XAllocWMHints _glfw.x11.xlib.AllocWMHints
205 #define XChangeProperty _glfw.x11.xlib.ChangeProperty
206 #define XChangeWindowAttributes _glfw.x11.xlib.ChangeWindowAttributes
207 #define XCheckIfEvent _glfw.x11.xlib.CheckIfEvent
208 #define XCheckTypedWindowEvent _glfw.x11.xlib.CheckTypedWindowEvent
209 #define XCloseDisplay _glfw.x11.xlib.CloseDisplay
210 #define XCloseIM _glfw.x11.xlib.CloseIM
211 #define XConvertSelection _glfw.x11.xlib.ConvertSelection
212 #define XCreateColormap _glfw.x11.xlib.CreateColormap
213 #define XCreateFontCursor _glfw.x11.xlib.CreateFontCursor
214 #define XCreateIC _glfw.x11.xlib.CreateIC
215 #define XCreateRegion _glfw.x11.xlib.CreateRegion
216 #define XCreateWindow _glfw.x11.xlib.CreateWindow
217 #define XDefineCursor _glfw.x11.xlib.DefineCursor
218 #define XDeleteContext _glfw.x11.xlib.DeleteContext
219 #define XDeleteProperty _glfw.x11.xlib.DeleteProperty
220 #define XDestroyIC _glfw.x11.xlib.DestroyIC
221 #define XDestroyRegion _glfw.x11.xlib.DestroyRegion
```

```
222 #define XDestroyWindow _glfw.x11.xlib.DestroyWindow
223 #define XDisplayKeycodes _glfw.x11.xlib.DisplayKeycodes
224 #define XEventsQueued _glfw.x11.xlib.EventsQueued
225 #define XFilterEvent _glfw.x11.xlib.FilterEvent
226 #define XFindContext _glfw.x11.xlib.FindContext
227 #define XFlush _glfw.x11.xlib.Flush
228 #define XFree _glfw.x11.xlib.Free
229 #define XFreeColormap _glfw.x11.xlib.FreeColormap
230 #define XFreeCursor _glfw.x11.xlib.FreeCursor
231 #define XFreeEventData _glfw.x11.xlib.FreeEventData
232 #define XGetErrorText _glfw.x11.xlib.GetErrorText
233 #define XGetEventData _glfw.x11.xlib.GetEventData
234 #define XGetICValues _glfw.x11.xlib.GetICValues
235 #define XGetIMValues _glfw.x11.xlib.GetIMValues
236 #define XGetInputFocus _glfw.x11.xlib.GetInputFocus
237 #define XGetKeyboardMapping _glfw.x11.xlib.GetKeyboardMapping
238 #define XGetScreenSaver _glfw.x11.xlib.GetScreenSaver
239 #define XGetSelectionOwner _glfw.x11.xlib.GetSelectionOwner
240 #define XGetVisualInfo _glfw.x11.xlib.GetVisualInfo
241 #define XGetWMNormalHints _glfw.x11.xlib.GetWMNormalHints
242 #define XGetWindowAttributes _glfw.x11.xlib.GetWindowAttributes
243 #define XGetWindowProperty _glfw.x11.xlib.GetWindowProperty
244 #define XGrabPointer _glfw.x11.xlib.GrabPointer
245 #define XIconifyWindow _glfw.x11.xlib.IconifyWindow
246 #define XInternAtom _glfw.x11.xlib.InternAtom
247 #define XLookupString _glfw.x11.xlib.LookupString
248 #define XMapRaised _glfw.x11.xlib.MapRaised
249 #define XMapWindow _glfw.x11.xlib.MapWindow
250 #define XMoveResizeWindow _glfw.x11.xlib.MoveResizeWindow
251 #define XMoveWindow _glfw.x11.xlib.MoveWindow
252 #define XNextEvent _glfw.x11.xlib.NextEvent
253 #define XOpenIM _glfw.x11.xlib.OpenIM
254 #define XPeekEvent _glfw.x11.xlib.PeekEvent
255 #define XPending _glfw.x11.xlib.Pending
256 #define XQueryExtension _glfw.x11.xlib.QueryExtension
257 #define XQueryPointer _glfw.x11.xlib.QueryPointer
258 #define XRaiseWindow _glfw.x11.xlib.RaiseWindow
259 #define XRegisterIMInstantiateCallback _glfw.x11.xlib.RegisterIMInstantiateCallback
260 #define XResizeWindow _glfw.x11.xlib.ResizeWindow
261 #define XResourceManagerString _glfw.x11.xlib.ResourceManagerString
262 #define XSaveContext _glfw.x11.xlib.SaveContext
263 #define XSelectInput _glfw.x11.xlib.SelectInput
264 #define XSendEvent _glfw.x11.xlib.SendEvent
265 #define XSetClassHint _glfw.x11.xlib.SetClassHint
266 #define XSetErrorHandler _glfw.x11.xlib.SetErrorHandler
267 #define XSetICFocus _glfw.x11.xlib.SetICFocus
268 #define XSetIMValues _glfw.x11.xlib.SetIMValues
269 #define XSetInputFocus _glfw.x11.xlib.SetInputFocus
270 #define XSetLocaleModifiers _glfw.x11.xlib.SetLocaleModifiers
271 #define XSetScreenSaver _glfw.x11.xlib.SetScreenSaver
272 #define XSetSelectionOwner _glfw.x11.xlib.SetSelectionOwner
273 #define XSetWMHints _glfw.x11.xlib.SetWMHints
274 #define XSetWMNormalHints _glfw.x11.xlib.SetWMNormalHints
275 #define XSetWMProtocols _glfw.x11.xlib.SetWMProtocols
276 #define XSupportsLocale _glfw.x11.xlib.SupportsLocale
277 #define XSync _glfw.x11.xlib.Sync
278 #define XTranslateCoordinates _glfw.x11.xlib.TranslateCoordinates
279 #define XUndefineCursor _glfw.x11.xlib.UndefineCursor
280 #define XUngrabPointer _glfw.x11.xlib.UngrabPointer
281 #define XUnmapWindow _glfw.x11.xlib.UnmapWindow
282 #define XUnsetICFocus _glfw.x11.xlib.UnsetICFocus
283 #define XVisualIDFromVisual _glfw.x11.xlib.VisualIDFromVisual
284 #define XWarpPointer _glfw.x11.xlib.WarpPointer
285 #define XkbFreeKeyboard _glfw.x11.xkb.FreeKeyboard
286 #define XkbFreeNames _glfw.x11.xkb.FreeNames
287 #define XkbGetMap _glfw.x11.xkb.GetMap
288 #define XkbGetNames _glfw.x11.xkb.GetNames
289 #define XkbGetState _glfw.x11.xkb.GetState
290 #define XkbKeycodeToKeysym _glfw.x11.xkb.KeycodeToKeysym
291 #define XkbQueryExtension _glfw.x11.xkb.QueryExtension
292 #define XkbSelectEventDetails _glfw.x11.xkb.SelectEventDetails
293 #define XkbSetDetectableAutoRepeat _glfw.x11.xkb.SetDetectableAutoRepeat
294 #define XrmDestroyDatabase _glfw.x11.xrm.DestroyDatabase
295 #define XrmGetResource _glfw.x11.xrm.GetResource
296 #define XrmGetStringDatabase _glfw.x11.xrm.GetStringDatabase
297 #define XrmUniqueQuark _glfw.x11.xrm.UniqueQuark
298 #define XUnregisterIMInstantiateCallback _glfw.x11.xlib.UnregisterIMInstantiateCallback
299 #define Xutf8LookupString _glfw.x11.xlib.utf8LookupString
300 #define Xutf8SetWMProperties _glfw.x11.xlib.utf8SetWMProperties
301
302 typedef XRRCrtcGamma* (* PFN_XRRAllocGamma)(int);
303 typedef void (* PFN_XRRFreeCrtcInfo)(XRRCrtcInfo*);
304 typedef void (* PFN_XRRFreeGamma)(XRRCrtcGamma*);
305 typedef void (* PFN_XRRFreeOutputInfo)(XRROutputInfo*);
306 typedef void (* PFN_XRRFreeScreenResources)(XRRScreenResources*);
307 typedef XRRCrtcGamma* (* PFN_XRRGetCrtcGamma)(Display*,RRCrtc);
308 typedef int (* PFN_XRRGetCrtcGammaSize)(Display*,RRCrtc);
```

```
309 typedef XRRCrtcInfo* (* PFN_XRRGetCrtcInfo) (Display*,XRRScreenResources*,RRCrtc);
310 typedef XRROutputInfo* (* PFN_XRRGetOutputInfo)(Display*,XRRScreenResources*,RROutput);
311 typedef RROutput (* PFN_XRRGetOutputPrimary)(Display*,Window);
312 typedef XRRScreenResources* (* PFN_XRRGetScreenResourcesCurrent)(Display*,Window);
313 typedef Bool (* PFN_XRRQueryExtension)(Display*,int*,int*);
314 typedef Status (* PFN_XRRQueryVersion)(Display*,int*,int*);
315 typedef void (* PFN_XRRSelectInput)(Display*,Window,int);
316 typedef Status (*
       PFN_XRRSetCrtcConfig)(Display*,XRRScreenResources*,RRCrtc,Time,int,int,RRMode,Rotation,RROutput*,int);
317 typedef void (* PFN_XRRSetCrtcGamma)(Display*,RRCrtc,XRRCrtcGamma*);
318 typedef int (* PFN_XRRUpdateConfiguration)(XEvent*);
319 #define XRRAllocGamma _glfw.x11.randr.AllocGamma
320 #define XRRFreeCrtcInfo _glfw.x11.randr.FreeCrtcInfo
321 #define XRRFreeGamma _glfw.x11.randr.FreeGamma
322 #define XRRFreeOutputInfo _glfw.x11.randr.FreeOutputInfo
323 #define XRRFreeScreenResources _glfw.x11.randr.FreeScreenResources
324 #define XRRGetCrtcGamma _glfw.x11.randr.GetCrtcGamma
325 #define XRRGetCrtcGammaSize _glfw.x11.randr.GetCrtcGammaSize
326 #define XRRGetCrtcInfo _glfw.x11.randr.GetCrtcInfo
327 #define XRRGetOutputInfo _glfw.x11.randr.GetOutputInfo
328 #define XRRGetOutputPrimary _glfw.x11.randr.GetOutputPrimary
329 #define XRRGetScreenResourcesCurrent _glfw.x11.randr.GetScreenResourcesCurrent
330 #define XRRQueryExtension _glfw.x11.randr.QueryExtension
331 #define XRRQueryVersion _glfw.x11.randr.QueryVersion
332 #define XRRSelectInput _glfw.x11.randr.SelectInput
333 #define XRRSetCrtcConfig _glfw.x11.randr.SetCrtcConfig
334 #define XRRSetCrtcGamma _glfw.x11.randr.SetCrtcGamma
335 #define XRRUpdateConfiguration _glfw.x11.randr.UpdateConfiguration
336
337 typedef XcursorImage* (* PFN_XcursorImageCreate)(int,int);
338 typedef void (* PFN_XcursorImageDestroy)(XcursorImage*);
339 typedef Cursor (* PFN_XcursorImageLoadCursor)(Display*,const XcursorImage*);
340 typedef char* (* PFN_XcursorGetTheme)(Display*);
341 typedef int (* PFN_XcursorGetDefaultSize)(Display*);
342 typedef XcursorImage* (* PFN_XcursorLibraryLoadImage)(const char*,const char*,int);
343 #define XcursorImageCreate _glfw.x11.xcursor.ImageCreate
344 #define XcursorImageDestroy _glfw.x11.xcursor.ImageDestroy
345 #define XcursorImageLoadCursor _glfw.x11.xcursor.ImageLoadCursor
346 #define XcursorGetTheme _glfw.x11.xcursor.GetTheme
347 #define XcursorGetDefaultSize _glfw.x11.xcursor.GetDefaultSize
348 #define XcursorLibraryLoadImage _glfw.x11.xcursor.LibraryLoadImage
349
350 typedef Bool (* PFN_XineramaIsActive)(Display*);
351 typedef Bool (* PFN_XineramaQueryExtension)(Display*,int*,int*);
352 typedef XineramaScreenInfo* (* PFN_XineramaQueryScreens)(Display*,int*);
353 #define XineramaIsActive _glfw.x11.xinerama.IsActive
354 #define XineramaQueryExtension _glfw.x11.xinerama.QueryExtension
355 #define XineramaQueryScreens _glfw.x11.xinerama.QueryScreens
356
357 typedef XID xcb_window_t;
358 typedef XID xcb_visualid_t;
359 typedef struct xcb_connection_t xcb_connection_t;
360 typedef xcb_connection_t* (* PFN_XGetXCBConnection)(Display*);
361 #define XGetXCBConnection _glfw.x11.x11xcb.GetXCBConnection
362
363 typedef Bool (* PFN_XF86VidModeQueryExtension)(Display*,int*,int*);
364 typedef Bool (* PFN_XF86VidModeGetGammaRamp)(Display*,int,int,unsigned short*,unsigned short*,unsigned
       short*);
365 typedef Bool (* PFN_XF86VidModeSetGammaRamp)(Display*,int,int,unsigned short*,unsigned short*,unsigned
       short*);
366 typedef Bool (* PFN_XF86VidModeGetGammaRampSize)(Display*,int,int*);
367 #define XF86VidModeQueryExtension _glfw.x11.vidmode.QueryExtension
368 #define XF86VidModeGetGammaRamp _glfw.x11.vidmode.GetGammaRamp
369 #define XF86VidModeSetGammaRamp _glfw.x11.vidmode.SetGammaRamp
370 #define XF86VidModeGetGammaRampSize _glfw.x11.vidmode.GetGammaRampSize
371
372 typedef Status (* PFN_XIQueryVersion)(Display*,int*,int*);
373 typedef int (* PFN_XISelectEvents)(Display*,Window,XIEventMask*,int);
374 #define XIQueryVersion _glfw.x11.xi.QueryVersion
375 #define XISelectEvents _glfw.x11.xi.SelectEvents
376
377 typedef Bool (* PFN_XRenderQueryExtension)(Display*,int*,int*);
378 typedef Status (* PFN_XRenderQueryVersion)(Display*dpy,int*,int*);
379 typedef XRenderPictFormat* (* PFN_XRenderFindVisualFormat)(Display*,Visual const*);
380 #define XRenderQueryExtension _glfw.x11.xrender.QueryExtension
381 #define XRenderQueryVersion _glfw.x11.xrender.QueryVersion
382 #define XRenderFindVisualFormat _glfw.x11.xrender.FindVisualFormat
383
384 typedef Bool (* PFN_XShapeQueryExtension)(Display*,int*,int*);
385 typedef Status (* PFN_XShapeQueryVersion)(Display*dpy,int*,int*);
386 typedef void (* PFN_XShapeCombineRegion)(Display*,Window,int,int,int,Region,int);
387 typedef void (* PFN_XShapeCombineMask)(Display*,Window,int,int,int,Pixmap,int);
388
389 #define XShapeQueryExtension _glfw.x11.xshape.QueryExtension
390 #define XShapeQueryVersion _glfw.x11.xshape.QueryVersion
391 #define XShapeCombineRegion _glfw.x11.xshape.ShapeCombineRegion
392 #define XShapeCombineMask _glfw.x11.xshape.ShapeCombineMask
```

```
393
394 typedef int (*PFNGLXGETFBCONFIGATTRIBPROC)(Display*,GLXFBConfig,int,int*);
395 typedef const char* (*PFNGLXGETCLIENTSTRINGPROC)(Display*,int);
396 typedef Bool (*PFNGLXQUERYEXTENSIONPROC)(Display*,int*,int*);
397 typedef Bool (*PFNGLXQUERYVERSIONPROC)(Display*,int*,int*);
398 typedef void (*PFNGLXDESTROYCONTEXTPROC)(Display*,GLXContext);
399 typedef Bool (*PFNGLXMAKECURRENTPROC)(Display*,GLXDrawable,GLXContext);
400 typedef void (*PFNGLXSWAPBUFFERSPROC)(Display*,GLXDrawable);
401 typedef const char* (*PFNGLXQUERYEXTENSIONSSTRINGPROC)(Display*,int);
402 typedef GLXFBConfig* (*PFNGLXGETFBCONFIGSPROC)(Display*,int,int*);
403 typedef GLXContext (*PFNGLXCREATENEWCONTEXTPROC)(Display*,GLXFBConfig,int,GLXContext,Bool);
404 typedef __GLXextproc (* PFNGLXGETPROCADDRESSPROC)(const GLubyte *procName);
405 typedef void (*PFNGLXSWAPINTERVALEXTPROC)(Display*,GLXDrawable,int);
406 typedef XVisualInfo* (*PFNGLXGETVISUALFROMFBCONFIGPROC)(Display*,GLXFBConfig);
407 typedef GLXWindow (*PFNGLXCREATEWINDOWPROC)(Display*,GLXFBConfig,Window,const int*);
408 typedef void (*PFNGLXDESTROYWINDOWPROC)(Display*,GLXWindow);
409
410 typedef int (*PFNGLXSWAPINTERVALMESAPROC)(int);
411 typedef int (*PFNGLXSWAPINTERVALSGIPROC)(int);
412 typedef GLXContext (*PFNGLXCREATECONTEXTATTRIBSARBPROC)(Display*,GLXFBConfig,GLXContext,Bool,const
       int*);
413
414 // libGL.so function pointer typedefs
415 #define glXGetFBConfigs _glfw.glx.GetFBConfigs
416 #define glXGetFBConfigAttrib _glfw.glx.GetFBConfigAttrib
417 #define glXGetClientString _glfw.glx.GetClientString
418 #define glXQueryExtension _glfw.glx.QueryExtension
419 #define glXQueryVersion _glfw.glx.QueryVersion
420 #define glXDestroyContext _glfw.glx.DestroyContext
421 #define glXMakeCurrent _glfw.glx.MakeCurrent
422 #define glXSwapBuffers _glfw.glx.SwapBuffers
423 #define glXQueryExtensionsString _glfw.glx.QueryExtensionsString
424 #define glXCreateNewContext _glfw.glx.CreateNewContext
425 #define glXGetVisualFromFBConfig _glfw.glx.GetVisualFromFBConfig
426 #define glXCreateWindow _glfw.glx.CreateWindow
427 #define glXDestroyWindow _glfw.glx.DestroyWindow
428
429 typedef VkFlags VkXlibSurfaceCreateFlagsKHR;
430 typedef VkFlags VkXcbSurfaceCreateFlagsKHR;
431
432 typedef struct VkXlibSurfaceCreateInfoKHR
433 {
434     VkStructureType             sType;
435     const void*                 pNext;
436     VkXlibSurfaceCreateFlagsKHR flags;
437     Display*                    dpy;
438     Window                      window;
439 } VkXlibSurfaceCreateInfoKHR;
440
441 typedef struct VkXcbSurfaceCreateInfoKHR
442 {
443     VkStructureType             sType;
444     const void*                 pNext;
445     VkXcbSurfaceCreateFlagsKHR  flags;
446     xcb_connection_t*           connection;
447     xcb_window_t                window;
448 } VkXcbSurfaceCreateInfoKHR;
449
450 typedef VkResult (APIENTRY *PFN_vkCreateXlibSurfaceKHR)(VkInstance,const
       VkXlibSurfaceCreateInfoKHR*,const VkAllocationCallbacks*,VkSurfaceKHR*);
451 typedef VkBool32 (APIENTRY
       *PFN_vkGetPhysicalDeviceXlibPresentationSupportKHR)(VkPhysicalDevice,uint32_t,Display*,VisualID);
452 typedef VkResult (APIENTRY *PFN_vkCreateXcbSurfaceKHR)(VkInstance,const VkXcbSurfaceCreateInfoKHR*,const
       VkAllocationCallbacks*,VkSurfaceKHR*);
453 typedef VkBool32 (APIENTRY
       *PFN_vkGetPhysicalDeviceXcbPresentationSupportKHR)(VkPhysicalDevice,uint32_t,xcb_connection_t*,xcb_visualid_t);
454
455 #include "xkb_unicode.h"
456 #include "posix_poll.h"
457
458 #define GLFW_X11_WINDOW_STATE           _GLFWwindowX11 x11;
459 #define GLFW_X11_LIBRARY_WINDOW_STATE   _GLFWlibraryX11 x11;
460 #define GLFW_X11_MONITOR_STATE          _GLFWmonitorX11 x11;
461 #define GLFW_X11_CURSOR_STATE           _GLFWcursorX11 x11;
462
463 #define GLFW_GLX_CONTEXT_STATE          _GLFWcontextGLX glx;
464 #define GLFW_GLX_LIBRARY_CONTEXT_STATE  _GLFWlibraryGLX glx;
465
466
467 // GLX-specific per-context data
468 //
469 typedef struct _GLFWcontextGLX
470 {
471     GLXContext      handle;
472     GLXWindow       window;
473 } _GLFWcontextGLX;
474
```

```
475 // GLX-specific global data
476 //
477 typedef struct _GLFWlibraryGLX
478 {
479     int             major, minor;
480     int             eventBase;
481     int             errorBase;
482
483     // dlopen handle for libGL.so.1
484     void*           handle;
485
486     // GLX 1.3 functions
487     PFNGLXGETFBCONFIGSPROC              GetFBConfigs;
488     PFNGLXGETFBCONFIGATTRIBPROC         GetFBConfigAttrib;
489     PFNGLXGETCLIENTSTRINGPROC           GetClientString;
490     PFNGLXQUERYEXTENSIONPROC            QueryExtension;
491     PFNGLXQUERYVERSIONPROC              QueryVersion;
492     PFNGLXDESTROYCONTEXTPROC            DestroyContext;
493     PFNGLXMAKECURRENTPROC               MakeCurrent;
494     PFNGLXSWAPBUFFERSPROC               SwapBuffers;
495     PFNGLXQUERYEXTENSIONSSTRINGPROC     QueryExtensionsString;
496     PFNGLXCREATENEWCONTEXTPROC          CreateNewContext;
497     PFNGLXGETVISUALFROMFBCONFIGPROC     GetVisualFromFBConfig;
498     PFNGLXCREATEWINDOWPROC              CreateWindow;
499     PFNGLXDESTROYWINDOWPROC             DestroyWindow;
500
501     // GLX 1.4 and extension functions
502     PFNGLXGETPROCADDRESSPROC            GetProcAddress;
503     PFNGLXGETPROCADDRESSPROC            GetProcAddressARB;
504     PFNGLXSWAPINTERVALSGIPROC           SwapIntervalSGI;
505     PFNGLXSWAPINTERVALEXTPROC           SwapIntervalEXT;
506     PFNGLXSWAPINTERVALMESAPROC          SwapIntervalMESA;
507     PFNGLXCREATECONTEXTATTRIBSARBPROC   CreateContextAttribsARB;
508     GLFWbool        SGI_swap_control;
509     GLFWbool        EXT_swap_control;
510     GLFWbool        MESA_swap_control;
511     GLFWbool        ARB_multisample;
512     GLFWbool        ARB_framebuffer_sRGB;
513     GLFWbool        EXT_framebuffer_sRGB;
514     GLFWbool        ARB_create_context;
515     GLFWbool        ARB_create_context_profile;
516     GLFWbool        ARB_create_context_robustness;
517     GLFWbool        EXT_create_context_es2_profile;
518     GLFWbool        ARB_create_context_no_error;
519     GLFWbool        ARB_context_flush_control;
520 } _GLFWlibraryGLX;
521
522 // X11-specific per-window data
523 //
524 typedef struct _GLFWwindowX11
525 {
526     Colormap        colormap;
527     Window          handle;
528     Window          parent;
529     XIC             ic;
530
531     GLFWbool        overrideRedirect;
532     GLFWbool        iconified;
533     GLFWbool        maximized;
534
535     // Whether the visual supports framebuffer transparency
536     GLFWbool        transparent;
537
538     // Cached position and size used to filter out duplicate events
539     int             width, height;
540     int             xpos, ypos;
541
542     // The last received cursor position, regardless of source
543     int             lastCursorPosX, lastCursorPosY;
544     // The last position the cursor was warped to by GLFW
545     int             warpCursorPosX, warpCursorPosY;
546
547     // The time of the last KeyPress event per keycode, for discarding
548     // duplicate key events generated for some keys by ibus
549     Time            keyPressTimes[256];
550 } _GLFWwindowX11;
551
552 // X11-specific global data
553 //
554 typedef struct _GLFWlibraryX11
555 {
556     Display*        display;
557     int             screen;
558     Window          root;
559
560     // System content scale
561     float           contentScaleX, contentScaleY;
```

```
562     // Helper window for IPC
563     Window          helperWindowHandle;
564     // Invisible cursor for hidden cursor mode
565     Cursor          hiddenCursorHandle;
566     // Context for mapping window XIDs to _GLFWwindow pointers
567     XContext        context;
568     // XIM input method
569     XIM             im;
570     // Most recent error code received by X error handler
571     int             errorCode;
572     // Primary selection string (while the primary selection is owned)
573     char*           primarySelectionString;
574     // Clipboard string (while the selection is owned)
575     char*           clipboardString;
576     // Key name string
577     char            keynames[GLFW_KEY_LAST + 1][5];
578     // X11 keycode to GLFW key LUT
579     short int       keycodes[256];
580     // GLFW key to X11 keycode LUT
581     short int       scancodes[GLFW_KEY_LAST + 1];
582     // Where to place the cursor when re-enabled
583     double          restoreCursorPosX, restoreCursorPosY;
584     // The window whose disabled cursor mode is active
585     _GLFWwindow*    disabledCursorWindow;
586     int             emptyEventPipe[2];
587
588     // Window manager atoms
589     Atom            NET_SUPPORTED;
590     Atom            NET_SUPPORTING_WM_CHECK;
591     Atom            WM_PROTOCOLS;
592     Atom            WM_STATE;
593     Atom            WM_DELETE_WINDOW;
594     Atom            NET_WM_NAME;
595     Atom            NET_WM_ICON_NAME;
596     Atom            NET_WM_ICON;
597     Atom            NET_WM_PID;
598     Atom            NET_WM_PING;
599     Atom            NET_WM_WINDOW_TYPE;
600     Atom            NET_WM_WINDOW_TYPE_NORMAL;
601     Atom            NET_WM_STATE;
602     Atom            NET_WM_STATE_ABOVE;
603     Atom            NET_WM_STATE_FULLSCREEN;
604     Atom            NET_WM_STATE_MAXIMIZED_VERT;
605     Atom            NET_WM_STATE_MAXIMIZED_HORZ;
606     Atom            NET_WM_STATE_DEMANDS_ATTENTION;
607     Atom            NET_WM_BYPASS_COMPOSITOR;
608     Atom            NET_WM_FULLSCREEN_MONITORS;
609     Atom            NET_WM_WINDOW_OPACITY;
610     Atom            NET_WM_CM_Sx;
611     Atom            NET_WORKAREA;
612     Atom            NET_CURRENT_DESKTOP;
613     Atom            NET_ACTIVE_WINDOW;
614     Atom            NET_FRAME_EXTENTS;
615     Atom            NET_REQUEST_FRAME_EXTENTS;
616     Atom            MOTIF_WM_HINTS;
617
618     // Xdnd (drag and drop) atoms
619     Atom            XdndAware;
620     Atom            XdndEnter;
621     Atom            XdndPosition;
622     Atom            XdndStatus;
623     Atom            XdndActionCopy;
624     Atom            XdndDrop;
625     Atom            XdndFinished;
626     Atom            XdndSelection;
627     Atom            XdndTypeList;
628     Atom            text_uri_list;
629
630     // Selection (clipboard) atoms
631     Atom            TARGETS;
632     Atom            MULTIPLE;
633     Atom            INCR;
634     Atom            CLIPBOARD;
635     Atom            PRIMARY;
636     Atom            CLIPBOARD_MANAGER;
637     Atom            SAVE_TARGETS;
638     Atom            NULL_;
639     Atom            UTF8_STRING;
640     Atom            COMPOUND_STRING;
641     Atom            ATOM_PAIR;
642     Atom            GLFW_SELECTION;
643
644     struct {
645         void*       handle;
646         GLFWbool    utf8;
647         PFN_XAllocClassHint AllocClassHint;
648         PFN_XAllocSizeHints AllocSizeHints;
```

```
649         PFN_XAllocWMHints AllocWMHints;
650         PFN_XChangeProperty ChangeProperty;
651         PFN_XChangeWindowAttributes ChangeWindowAttributes;
652         PFN_XCheckIfEvent CheckIfEvent;
653         PFN_XCheckTypedWindowEvent CheckTypedWindowEvent;
654         PFN_XCloseDisplay CloseDisplay;
655         PFN_XCloseIM CloseIM;
656         PFN_XConvertSelection ConvertSelection;
657         PFN_XCreateColormap CreateColormap;
658         PFN_XCreateFontCursor CreateFontCursor;
659         PFN_XCreateIC CreateIC;
660         PFN_XCreateRegion CreateRegion;
661         PFN_XCreateWindow CreateWindow;
662         PFN_XDefineCursor DefineCursor;
663         PFN_XDeleteContext DeleteContext;
664         PFN_XDeleteProperty DeleteProperty;
665         PFN_XDestroyIC DestroyIC;
666         PFN_XDestroyRegion DestroyRegion;
667         PFN_XDestroyWindow DestroyWindow;
668         PFN_XDisplayKeycodes DisplayKeycodes;
669         PFN_XEventsQueued EventsQueued;
670         PFN_XFilterEvent FilterEvent;
671         PFN_XFindContext FindContext;
672         PFN_XFlush Flush;
673         PFN_XFree Free;
674         PFN_XFreeColormap FreeColormap;
675         PFN_XFreeCursor FreeCursor;
676         PFN_XFreeEventData FreeEventData;
677         PFN_XGetErrorText GetErrorText;
678         PFN_XGetEventData GetEventData;
679         PFN_XGetICValues GetICValues;
680         PFN_XGetIMValues GetIMValues;
681         PFN_XGetInputFocus GetInputFocus;
682         PFN_XGetKeyboardMapping GetKeyboardMapping;
683         PFN_XGetScreenSaver GetScreenSaver;
684         PFN_XGetSelectionOwner GetSelectionOwner;
685         PFN_XGetVisualInfo GetVisualInfo;
686         PFN_XGetWMNormalHints GetWMNormalHints;
687         PFN_XGetWindowAttributes GetWindowAttributes;
688         PFN_XGetWindowProperty GetWindowProperty;
689         PFN_XGrabPointer GrabPointer;
690         PFN_XIconifyWindow IconifyWindow;
691         PFN_XInternAtom InternAtom;
692         PFN_XLookupString LookupString;
693         PFN_XMapRaised MapRaised;
694         PFN_XMapWindow MapWindow;
695         PFN_XMoveResizeWindow MoveResizeWindow;
696         PFN_XMoveWindow MoveWindow;
697         PFN_XNextEvent NextEvent;
698         PFN_XOpenIM OpenIM;
699         PFN_XPeekEvent PeekEvent;
700         PFN_XPending Pending;
701         PFN_XQueryExtension QueryExtension;
702         PFN_XQueryPointer QueryPointer;
703         PFN_XRaiseWindow RaiseWindow;
704         PFN_XRegisterIMInstantiateCallback RegisterIMInstantiateCallback;
705         PFN_XResizeWindow ResizeWindow;
706         PFN_XResourceManagerString ResourceManagerString;
707         PFN_XSaveContext SaveContext;
708         PFN_XSelectInput SelectInput;
709         PFN_XSendEvent SendEvent;
710         PFN_XSetClassHint SetClassHint;
711         PFN_XSetErrorHandler SetErrorHandler;
712         PFN_XSetICFocus SetICFocus;
713         PFN_XSetIMValues SetIMValues;
714         PFN_XSetInputFocus SetInputFocus;
715         PFN_XSetLocaleModifiers SetLocaleModifiers;
716         PFN_XSetScreenSaver SetScreenSaver;
717         PFN_XSetSelectionOwner SetSelectionOwner;
718         PFN_XSetWMHints SetWMHints;
719         PFN_XSetWMNormalHints SetWMNormalHints;
720         PFN_XSetWMProtocols SetWMProtocols;
721         PFN_XSupportsLocale SupportsLocale;
722         PFN_XSync Sync;
723         PFN_XTranslateCoordinates TranslateCoordinates;
724         PFN_XUndefineCursor UndefineCursor;
725         PFN_XUngrabPointer UngrabPointer;
726         PFN_XUnmapWindow UnmapWindow;
727         PFN_XUnsetICFocus UnsetICFocus;
728         PFN_XVisualIDFromVisual VisualIDFromVisual;
729         PFN_XWarpPointer WarpPointer;
730         PFN_XUnregisterIMInstantiateCallback UnregisterIMInstantiateCallback;
731         PFN_Xutf8LookupString utf8LookupString;
732         PFN_Xutf8SetWMProperties utf8SetWMProperties;
733     } xlib;
734
735     struct {
```

```
736         PFN_XrmDestroyDatabase DestroyDatabase;
737         PFN_XrmGetResource GetResource;
738         PFN_XrmGetStringDatabase GetStringDatabase;
739         PFN_XrmUniqueQuark UniqueQuark;
740     } xrm;
741
742     struct {
743         GLFWbool    available;
744         void*       handle;
745         int         eventBase;
746         int         errorBase;
747         int         major;
748         int         minor;
749         GLFWbool    gammaBroken;
750         GLFWbool    monitorBroken;
751         PFN_XRRAllocGamma AllocGamma;
752         PFN_XRRFreeCrtcInfo FreeCrtcInfo;
753         PFN_XRRFreeGamma FreeGamma;
754         PFN_XRRFreeOutputInfo FreeOutputInfo;
755         PFN_XRRFreeScreenResources FreeScreenResources;
756         PFN_XRRGetCrtcGamma GetCrtcGamma;
757         PFN_XRRGetCrtcGammaSize GetCrtcGammaSize;
758         PFN_XRRGetCrtcInfo GetCrtcInfo;
759         PFN_XRRGetOutputInfo GetOutputInfo;
760         PFN_XRRGetOutputPrimary GetOutputPrimary;
761         PFN_XRRGetScreenResourcesCurrent GetScreenResourcesCurrent;
762         PFN_XRRQueryExtension QueryExtension;
763         PFN_XRRQueryVersion QueryVersion;
764         PFN_XRRSelectInput SelectInput;
765         PFN_XRRSetCrtcConfig SetCrtcConfig;
766         PFN_XRRSetCrtcGamma SetCrtcGamma;
767         PFN_XRRUpdateConfiguration UpdateConfiguration;
768     } randr;
769
770     struct {
771         GLFWbool     available;
772         GLFWbool     detectable;
773         int          majorOpcode;
774         int          eventBase;
775         int          errorBase;
776         int          major;
777         int          minor;
778         unsigned int group;
779         PFN_XkbFreeKeyboard FreeKeyboard;
780         PFN_XkbFreeNames FreeNames;
781         PFN_XkbGetMap GetMap;
782         PFN_XkbGetNames GetNames;
783         PFN_XkbGetState GetState;
784         PFN_XkbKeycodeToKeysym KeycodeToKeysym;
785         PFN_XkbQueryExtension QueryExtension;
786         PFN_XkbSelectEventDetails SelectEventDetails;
787         PFN_XkbSetDetectableAutoRepeat SetDetectableAutoRepeat;
788     } xkb;
789
790     struct {
791         int         count;
792         int         timeout;
793         int         interval;
794         int         blanking;
795         int         exposure;
796     } saver;
797
798     struct {
799         int         version;
800         Window      source;
801         Atom        format;
802     } xdnd;
803
804     struct {
805         void*       handle;
806         PFN_XcursorImageCreate ImageCreate;
807         PFN_XcursorImageDestroy ImageDestroy;
808         PFN_XcursorImageLoadCursor ImageLoadCursor;
809         PFN_XcursorGetTheme GetTheme;
810         PFN_XcursorGetDefaultSize GetDefaultSize;
811         PFN_XcursorLibraryLoadImage LibraryLoadImage;
812     } xcursor;
813
814     struct {
815         GLFWbool    available;
816         void*       handle;
817         int         major;
818         int         minor;
819         PFN_XineramaIsActive IsActive;
820         PFN_XineramaQueryExtension QueryExtension;
821         PFN_XineramaQueryScreens QueryScreens;
822     } xinerama;
```

```
823
824     struct {
825         void*       handle;
826         PFN_XGetXCBConnection GetXCBConnection;
827     } x11xcb;
828
829     struct {
830         GLFWbool    available;
831         void*       handle;
832         int         eventBase;
833         int         errorBase;
834         PFN_XF86VidModeQueryExtension QueryExtension;
835         PFN_XF86VidModeGetGammaRamp GetGammaRamp;
836         PFN_XF86VidModeSetGammaRamp SetGammaRamp;
837         PFN_XF86VidModeGetGammaRampSize GetGammaRampSize;
838     } vidmode;
839
840     struct {
841         GLFWbool    available;
842         void*       handle;
843         int         majorOpcode;
844         int         eventBase;
845         int         errorBase;
846         int         major;
847         int         minor;
848         PFN_XIQueryVersion QueryVersion;
849         PFN_XISelectEvents SelectEvents;
850     } xi;
851
852     struct {
853         GLFWbool    available;
854         void*       handle;
855         int         major;
856         int         minor;
857         int         eventBase;
858         int         errorBase;
859         PFN_XRenderQueryExtension QueryExtension;
860         PFN_XRenderQueryVersion QueryVersion;
861         PFN_XRenderFindVisualFormat FindVisualFormat;
862     } xrender;
863
864     struct {
865         GLFWbool    available;
866         void*       handle;
867         int         major;
868         int         minor;
869         int         eventBase;
870         int         errorBase;
871         PFN_XShapeQueryExtension QueryExtension;
872         PFN_XShapeCombineRegion ShapeCombineRegion;
873         PFN_XShapeQueryVersion QueryVersion;
874         PFN_XShapeCombineMask ShapeCombineMask;
875     } xshape;
876 } _GLFWlibraryX11;
877
878 // X11-specific per-monitor data
879 //
880 typedef struct _GLFWmonitorX11
881 {
882     RROutput        output;
883     RRCrtc          crtc;
884     RRMode          oldMode;
885
886     // Index of corresponding Xinerama screen,
887     // for EWMH full screen window placement
888     int             index;
889 } _GLFWmonitorX11;
890
891 // X11-specific per-cursor data
892 //
893 typedef struct _GLFWcursorX11
894 {
895     Cursor handle;
896 } _GLFWcursorX11;
897
898
899 GLFWbool _glfwConnectX11(int platformID, _GLFWplatform* platform);
900 int _glfwInitX11(void);
901 void _glfwTerminateX11(void);
902
903 int _glfwCreateWindowX11(_GLFWwindow* window, const _GLFWwndconfig* wndconfig, const _GLFWctxconfig*
    ctxconfig, const _GLFWfbconfig* fbconfig);
904 void _glfwDestroyWindowX11(_GLFWwindow* window);
905 void _glfwSetWindowTitleX11(_GLFWwindow* window, const char* title);
906 void _glfwSetWindowIconX11(_GLFWwindow* window, int count, const GLFWimage* images);
907 void _glfwGetWindowPosX11(_GLFWwindow* window, int* xpos, int* ypos);
908 void _glfwSetWindowPosX11(_GLFWwindow* window, int xpos, int ypos);
```

```
909 void _glfwGetWindowSizeX11(_GLFWwindow* window, int* width, int* height);
910 void _glfwSetWindowSizeX11(_GLFWwindow* window, int width, int height);
911 void _glfwSetWindowSizeLimitsX11(_GLFWwindow* window, int minwidth, int minheight, int maxwidth, int
       maxheight);
912 void _glfwSetWindowAspectRatioX11(_GLFWwindow* window, int numer, int denom);
913 void _glfwGetFramebufferSizeX11(_GLFWwindow* window, int* width, int* height);
914 void _glfwGetWindowFrameSizeX11(_GLFWwindow* window, int* left, int* top, int* right, int* bottom);
915 void _glfwGetWindowContentScaleX11(_GLFWwindow* window, float* xscale, float* yscale);
916 void _glfwIconifyWindowX11(_GLFWwindow* window);
917 void _glfwRestoreWindowX11(_GLFWwindow* window);
918 void _glfwMaximizeWindowX11(_GLFWwindow* window);
919 void _glfwShowWindowX11(_GLFWwindow* window);
920 void _glfwHideWindowX11(_GLFWwindow* window);
921 void _glfwRequestWindowAttentionX11(_GLFWwindow* window);
922 void _glfwFocusWindowX11(_GLFWwindow* window);
923 void _glfwSetWindowMonitorX11(_GLFWwindow* window, _GLFWmonitor* monitor, int xpos, int ypos, int width,
       int height, int refreshRate);
924 int _glfwWindowFocusedX11(_GLFWwindow* window);
925 int _glfwWindowIconifiedX11(_GLFWwindow* window);
926 int _glfwWindowVisibleX11(_GLFWwindow* window);
927 int _glfwWindowMaximizedX11(_GLFWwindow* window);
928 int _glfwWindowHoveredX11(_GLFWwindow* window);
929 int _glfwFramebufferTransparentX11(_GLFWwindow* window);
930 void _glfwSetWindowResizableX11(_GLFWwindow* window, GLFWbool enabled);
931 void _glfwSetWindowDecoratedX11(_GLFWwindow* window, GLFWbool enabled);
932 void _glfwSetWindowFloatingX11(_GLFWwindow* window, GLFWbool enabled);
933 float _glfwGetWindowOpacityX11(_GLFWwindow* window);
934 void _glfwSetWindowOpacityX11(_GLFWwindow* window, float opacity);
935 void _glfwSetWindowMousePassthroughX11(_GLFWwindow* window, GLFWbool enabled);
936
937 void _glfwSetRawMouseMotionX11(_GLFWwindow *window, GLFWbool enabled);
938 GLFWbool _glfwRawMouseMotionSupportedX11(void);
939
940 void _glfwPollEventsX11(void);
941 void _glfwWaitEventsX11(void);
942 void _glfwWaitEventsTimeoutX11(double timeout);
943 void _glfwPostEmptyEventX11(void);
944
945 void _glfwGetCursorPosX11(_GLFWwindow* window, double* xpos, double* ypos);
946 void _glfwSetCursorPosX11(_GLFWwindow* window, double xpos, double ypos);
947 void _glfwSetCursorModeX11(_GLFWwindow* window, int mode);
948 const char* _glfwGetScancodeNameX11(int scancode);
949 int _glfwGetKeyScancodeX11(int key);
950 int _glfwCreateCursorX11(_GLFWcursor* cursor, const GLFWimage* image, int xhot, int yhot);
951 int _glfwCreateStandardCursorX11(_GLFWcursor* cursor, int shape);
952 void _glfwDestroyCursorX11(_GLFWcursor* cursor);
953 void _glfwSetCursorX11(_GLFWwindow* window, _GLFWcursor* cursor);
954 void _glfwSetClipboardStringX11(const char* string);
955 const char* _glfwGetClipboardStringX11(void);
956
957 EGLenum _glfwGetEGLPlatformX11(EGLint** attribs);
958 EGLNativeDisplayType _glfwGetEGLNativeDisplayX11(void);
959 EGLNativeWindowType _glfwGetEGLNativeWindowX11(_GLFWwindow* window);
960
961 void _glfwGetRequiredInstanceExtensionsX11(char** extensions);
962 int _glfwGetPhysicalDevicePresentationSupportX11(VkInstance instance, VkPhysicalDevice device, uint32_t
       queuefamily);
963 VkResult _glfwCreateWindowSurfaceX11(VkInstance instance, _GLFWwindow* window, const
       VkAllocationCallbacks* allocator, VkSurfaceKHR* surface);
964
965 void _glfwFreeMonitorX11(_GLFWmonitor* monitor);
966 void _glfwGetMonitorPosX11(_GLFWmonitor* monitor, int* xpos, int* ypos);
967 void _glfwGetMonitorContentScaleX11(_GLFWmonitor* monitor, float* xscale, float* yscale);
968 void _glfwGetMonitorWorkareaX11(_GLFWmonitor* monitor, int* xpos, int* ypos, int* width, int* height);
969 GLFWvidmode* _glfwGetVideoModesX11(_GLFWmonitor* monitor, int* count);
970 void _glfwGetVideoModeX11(_GLFWmonitor* monitor, GLFWvidmode* mode);
971 GLFWbool _glfwGetGammaRampX11(_GLFWmonitor* monitor, GLFWgammaramp* ramp);
972 void _glfwSetGammaRampX11(_GLFWmonitor* monitor, const GLFWgammaramp* ramp);
973
974 void _glfwPollMonitorsX11(void);
975 void _glfwSetVideoModeX11(_GLFWmonitor* monitor, const GLFWvidmode* desired);
976 void _glfwRestoreVideoModeX11(_GLFWmonitor* monitor);
977
978 Cursor _glfwCreateNativeCursorX11(const GLFWimage* image, int xhot, int yhot);
979
980 unsigned long _glfwGetWindowPropertyX11(Window window,
981                                         Atom property,
982                                         Atom type,
983                                         unsigned char** value);
984 GLFWbool _glfwIsVisualTransparentX11(Visual* visual);
985
986 void _glfwGrabErrorHandlerX11(void);
987 void _glfwReleaseErrorHandlerX11(void);
988 void _glfwInputErrorX11(int error, const char* message);
989
990 void _glfwPushSelectionToManagerX11(void);
991 void _glfwCreateInputContextX11(_GLFWwindow* window);
```

```
992
993 GLFWbool _glfwInitGLX(void);
994 void _glfwTerminateGLX(void);
995 GLFWbool _glfwCreateContextGLX(_GLFWwindow* window,
996                                const _GLFWctxconfig* ctxconfig,
997                                const _GLFWfbconfig* fbconfig);
998 void _glfwDestroyContextGLX(_GLFWwindow* window);
999 GLFWbool _glfwChooseVisualGLX(const _GLFWwndconfig* wndconfig,
1000                               const _GLFWctxconfig* ctxconfig,
1001                               const _GLFWfbconfig* fbconfig,
1002                               Visual** visual, int* depth);
1003
```

27.37 xkb_unicode.h

```
1 //========================================================================
2 // GLFW 3.4 Linux - www.glfw.org
3 //------------------------------------------------------------------------
4 // Copyright (c) 2014 Jonas Ådahl <jadahl@gmail.com>
5 //
6 // This software is provided 'as-is', without any express or implied
7 // warranty. In no event will the authors be held liable for any damages
8 // arising from the use of this software.
9 //
10 // Permission is granted to anyone to use this software for any purpose,
11 // including commercial applications, and to alter it and redistribute it
12 // freely, subject to the following restrictions:
13 //
14 // 1. The origin of this software must not be misrepresented; you must not
15 //    claim that you wrote the original software. If you use this software
16 //    in a product, an acknowledgment in the product documentation would
17 //    be appreciated but is not required.
18 //
19 // 2. Altered source versions must be plainly marked as such, and must not
20 //    be misrepresented as being the original software.
21 //
22 // 3. This notice may not be removed or altered from any source
23 //    distribution.
24 //
25 //========================================================================
26
27 #define GLFW_INVALID_CODEPOINT 0xffffffffu
28
29 uint32_t _glfwKeySym2Unicode(unsigned int keysym);
30
```