GLFW

Mike Bailey
mjb@cs.oregonstate.edu

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License

http://www.glfw.org/

GLFW is an Open Source, multi-platform library for OpenGL, OpenGL ES and Vulkan development on the desktop. It provides a simple API for creating window contexts and surfaces, receiving input and events.

GLFW is written in C and has native support for Windows, macOS and many Unix-like systems using the X Window System, such as Linux and FreeBSD.

GLFW is released under the zlib/png/license.

Gives you a window and OpenGL context with just two function calls

Support for OpenGL, OpenGL ES, Vulkan and related options, flags and extensions

Support for multiple windows, multiple monitors, high-DPI and gamma ramps

Support for keyboard, mouse, gestures, line and window event input, via polling or callbacks

Comes with guides, tutorials, reference documentation, examples and test programs

Open Source with an OSI-certified license allowing commercial use

Access to native objects and compile-time options for platform-specific features

Community-maintained bindings for many different languages

No library can be perfect for everyone. If GLFW isn't what you're looking for, there are alternatives.
Setting Up GLFW

```c
#define GLFW_INCLUDE_VULKAN
#include "glfw3.h"

uint32_t Width, Height;
VkSurfaceKHR Surface;

void InitGLFW()
{
    glfwInit();
    if( !glfwVulkanSupported() )
    {
        fprintf( stderr, "Vulkan is not supported on this system!\n" );
        exit( 1 );
    }
    glfwWindowHint( GLFW_CLIENT_API, GLFW_NO_API );
    glfwWindowHint( GLFW_RESIZABLE, GLFW_FALSE );
    MainWindow = glfwCreateWindow( Width, Height, "Vulkan Sample", NULL, NULL );
    VkResult result = glfwCreateWindowSurface( Instance, MainWindow, NULL, OUT &Surface );
    glfwSetErrorCallback( GLFWErrorCallback );
    glfwSetKeyCallback( MainWindow, GLFWKeyboard);
    glfwSetCursorPosCallback( MainWindow, GLFWMouseMotion);
    glfwSetMouseButtonCallback( MainWindow, GLFWMouseButton);
}
```

You Can Also Query What Vulkan Extensions GLFW Requires

```c
uint32_t count;
const char ** extensions = glfwGetRequiredInstanceExtensions (&count);

fprintf( FpDebug, "Found %d GLFW Required Instance Extensions:\n", count );
for( uint32_t i = 0; i < count; i++ )
{
    fprintf( FpDebug, "\t%s\n", extensions[ i ] );
}
```

```
Found 2 GLFW Required Instance Extensions:
VK_KHR_surface
VK_KHR_win32_surface
```
GLFW Keyboard Callback

```c
void GLFWKeyboard(GLFWwindow * window, int key, int scancode, int action, int mods )
{
    if( action == GLFW_PRESS )
    {
        switch( key )
        {
            case GLFW_KEY_M:
            case 'm':
            case 'M':
            Mode++;
            if( Mode >= 2 )
                Mode = 0;
            break;
            default:
                fprintf( FpDebug, "Unknow key hit: 0x%04x = '%c', key, key );
                fflush(FpDebug);
        }
    }
}
```

GLFW Mouse Button Callback

```c
void GLFWMouseButton(GLFWwindow *window, int button, int action, int mods )
{
    int b = 0;              // LEFT, MIDDLE, or RIGHT
    // get the proper button bit mask:
    switch( button )
    {
        case GLFW_MOUSE_BUTTON_LEFT:
            b = LEFT;               break;
        case GLFW_MOUSE_BUTTON_MIDDLE:
            b = MIDDLE;             break;
        case GLFW_MOUSE_BUTTON_RIGHT:
            b = RIGHT;              break;
        default:
            b = 0;
            fprintf( FpDebug, "Unknown mouse button: %d", button );
            break;
    }
    // button down sets the bit, up clears the bit:
    if( action == GLFW_PRESS )
    {
        double xpos, ypos;
        glfwGetCursorPos( window, &xpos, &ypos);
        Xmouse = (int)xpos;
        Ymouse = (int)ypos;
        ActiveButton |= b;              // set the proper bit
    }
    else
    {
        ActiveButton &= ~b;             // clear the proper bit
    }
}
```
void GLFWMouseMotion( GLFWwindow *window, double xpos, double ypos )
{
    int dx = (int)xpos - Xmouse;            // change in mouse coords
    int dy = (int)ypos - Ymouse;
    if( ( ActiveButton & LEFT ) != 0 )
    {
        Xrot += ( ANGFACT*dy );
        Yrot += ( ANGFACT*dx );
    }
    if( ( ActiveButton & MIDDLE ) != 0 )
    {
        Scale += SCLFACT * (float) ( dx - dy );
        // keep object from turning inside-out or disappearing:
        if( Scale < MINSIZE )
            Scale = MINSIZE;
    }
    Xmouse = (int)xpos;                     // new current position
    Ymouse = (int)ypos;
}

while( glfwWindowShouldClose( MainWindow ) == 0 )
{
    glfwPollEvents();
    Time = glfwGetTime();          // elapsed time, in double-precision seconds
    UpdateScene();
    RenderScene();
}

vkQueueWaitIdle( Queue );
vkDeviceWaitIdle( LogicalDevice );
DestroyAllVulkan();
glfwDestroyWindow( MainWindow );
glfwTerminate();
Loopying and Cloasing GLFW

If you would like to block waiting for events, use:

\[ \text{glfwWaitEvents();} \]

You can have the blocking wake up after a timeout period with:

\[ \text{glfwWaitEventsTimeout(double secs);} \]

You can wake up one of these blocks from another thread with:

\[ \text{glfwPostEmptyEvent();} \]