Slide 1

Vulkan®

GLFW

Oregon State University

Mike Bailey

mjb@cs.oregonstate.edu

Oregon State University
Computer Graphics

GLFW.pptx   mjb – September 17, 2018

Slide 2

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 windows, 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 licensed under the zlib/libpng 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, gamepad, time and window event input, via polling or callbacks

Comes with guides, a tutorial, 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, then there are alternatives.

Oregon State University
Computer Graphics   mjb – September 17, 2018

Slide 3

Setting Up GLFW

```
void
InitGLFW( )
{
        glfwInit( );
        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, &Surface );

        glfwSetErrorCallback( GLFWErrorCallback );
        glfwSetKeyCallback( MainWindow, GLFWKeyboard );
        glfwSetCursorPosCallback( MainWindow, GLFWMouseMotion );
        glfwSetMouseButtonCallback( MainWindow, GLFWMouseButton );
}
```

Oregon State University
Computer Graphics   mjb – September 17, 2018

Slide 4

GLFW Keyboard Callback

```
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'\n", key, key );
                  fflush(FpDebug);
          }
      }
}
```

Oregon State University
Computer Graphics   mjb – September 17, 2018

GLFW Mouse Button Callback

```
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\n", button );
        }

        // 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
        }
}
```

Oregon State
University
Computer Graphics

GLFW Mouse Motion Callback

```
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 < MINSCALE )
                        Scale = MINSCALE;
        }

        Xmouse = (int)xpos;             // new current position
        Ymouse = (int)ypos;
}
```

Oregon State
University
Computer Graphics

Looping and Closing GLFW

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

        vkQueueWaitIdle( Queue );
        vkDeviceWaitIdle( LogicalDevice );
        DestroyAllVulkan( );
        glfwDestroyWindow( MainWindow );
        glfwTerminate( );
```

Loop

Oregon State
University
Computer Graphics