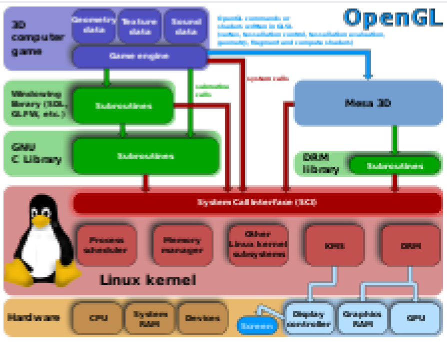
WIKIPEDIA
The Free Encyclopedia

OpenGL

OpenGL (**Open Graphics Library**[3]) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering.

Silicon Graphics, Inc. (SGI) began developing OpenGL in 1991 and released it on June 30, 1992;[4][5] applications use it extensively in the fields of computer-aided design (CAD), virtual reality, scientific visualization, information visualization, flight simulation, and video games. Since 2006, OpenGL has been managed by the non-profit technology consortium Khronos Group.[6]

OpenGL



Video games outsource real-time rendering calculations to the GPU over OpenGL. The rendered results are not sent back to main memory, but to the framebuffer of video memory instead. The display controller will then send this data to the display device.

Original author(s)	Silicon Graphics
Developer(s)	Khronos Group (formerly ARB)
Initial release	June 30, 1992
Stable release	4.6 / July 31, 2017
Written in	C[1]
Successor	Vulkan
Type	3D graphics API
License	▪ Open source license for use of the S.I.: This is a Free Software License B closely modeled on BSD, X, and

Design

The OpenGL specification describes an abstract API for drawing 2D and 3D graphics. Although it is possible for the API to be implemented entirely in software, it is designed to be implemented mostly or entirely in hardware.

The API is defined as a set of functions which may be called by the client program, alongside a set of named integer constants (for example, the constant GL_TEXTURE_2D, which corresponds to the decimal number 3553). Although the function definitions are superficially similar to those of the programming language C, they are language-independent. As such, OpenGL has many language bindings, some of the most noteworthy being the JavaScript binding WebGL (API, based on OpenGL ES 2.0, for 3D rendering from within a web browser); the C bindings WGL, GLX and CGL; the C binding provided by iOS; and the Java and C bindings provided by Android.

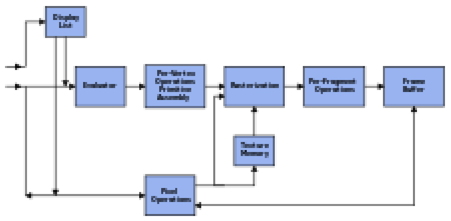
In addition to being language-independent, OpenGL is also cross-platform. The specification says nothing on the subject of obtaining and managing an OpenGL context, leaving this as a detail of the underlying windowing system. For the same reason, OpenGL is purely concerned with rendering, providing no APIs related to input, audio, or windowing.

Development

	Mozilla licenses. ▪ Trademark license for new licensees who want to use the OpenGL trademark and logo and claim conformance.[2]
Website	opengl.org (https://www.opengl.org/)

OpenGL is no longer in active development: whereas between 2001 and 2014 OpenGL specification was updated mostly on a yearly basis, with two releases (3.1 and 3.2) taking place in 2009 and three (3.3, 4.0 and 4.1) in 2010, the latest OpenGL specification 4.6 was released in 2017, after a three-year break, and was limited to inclusion of eleven existing ARB and EXT extensions into the core profile.[7]

Active development of OpenGL was dropped in favor of Vulkan API released in 2016 and codenamed glNext during initial development. In 2017, Khronos Group announced that OpenGL ES will not have new versions[8] and has since concentrated on development of Vulkan and other technologies.[9][10] As a result, certain capabilities offered by modern GPUs, e.g. ray tracing, are not supported by OpenGL.



An illustration of the graphics pipeline process

New versions of the OpenGL specifications are released by the Khronos Group, each of which extends the API to support various new features. The details of each version are decided by consensus between the Group's members, including graphics card manufacturers, operating system designers, and general technology companies such as Mozilla and Google.[11]

In addition to the features required by the core API, graphics processing unit (GPU) vendors may provide additional functionality in the form of *extensions*. Extensions may introduce new functions and new constants, and may relax or remove restrictions on existing OpenGL functions. Vendors can use extensions to expose custom APIs without needing support from other vendors or the Khronos Group as a whole, which greatly increases the flexibility of OpenGL. All extensions are collected in, and defined by, the OpenGL Registry.[12]

Each extension is associated with a short identifier, based on the name of the company which developed it. For example, Nvidia's identifier is NV, which is part of the extension name `GL_NV_half_float`, the constant `GL_HALF_FLOAT_NV`, and the function `glVertex2hNV()`.[13] If multiple vendors agree to implement the same functionality using the same API, a shared extension may be released, using the identifier EXT. In such cases, it could also happen that the Khronos Group's Architecture Review Board gives the extension their explicit approval, in which case the identifier ARB is used.[14]

The features introduced by each new version of OpenGL are typically formed from the combined features of several widely implemented extensions, especially extensions of type ARB or EXT.

Documentation

The OpenGL Architecture Review Board released a series of manuals along with the specification which have been updated to track changes in the API. These are commonly referred to by the colors of their covers:

The Red Book
: OpenGL Programming Guide, 9th Edition. ISBN 978-0-134-49549-1
: The Official Guide to Learning OpenGL, Version 4.5 with SPIR-V

The Orange Book
: OpenGL Shading Language, 3rd edition. ISBN 0-321-63763-1
: A tutorial and reference book for GLSL.

Historic books (pre-OpenGL 2.0):

The Green Book
: OpenGL Programming for the X Window System. ISBN 978-0-201-48359-8
: A book about X11 interfacing and OpenGL Utility Toolkit (GLUT).

The Blue Book
: OpenGL Reference manual, 4th edition. ISBN 0-321-17383-X
: Essentially a hard-copy printout of the Unix manual (man) pages for OpenGL.
: Includes a poster-sized fold-out diagram showing the structure of an idealised OpenGL implementation.

The Alpha Book (white cover)
: OpenGL Programming for Windows 95 and Windows NT. ISBN 0-201-40709-4
: A book about interfacing OpenGL with Microsoft Windows.

OpenGL's documentation is also accessible via its official webpage.[15]

Associated libraries

The earliest versions of OpenGL were released with a companion library called the OpenGL Utility Library (GLU). It provided simple, useful features which were unlikely to be supported in contemporary hardware, such as tessellating, and generating mipmaps and primitive shapes. The GLU specification was last updated in 1998 and depends on OpenGL features which are now deprecated.

Context and window toolkits

Given that creating an OpenGL context is quite a complex process, and given that it varies between operating systems, automatic OpenGL context creation has become a common feature of several game-development and user-interface libraries, including SDL, Allegro, SFML, FLTK, and Qt. A few libraries have been designed solely to produce an OpenGL-capable window. The first such library was OpenGL Utility Toolkit (GLUT), later superseded by freeglut. GLFW is a newer alternative.[16]

- These toolkits are designed to create and manage OpenGL windows, and manage input, but little beyond that.[17]
  - GLFW – A cross-platform windowing and keyboard-mouse-joystick handler; is more game-oriented
  - freeglut – A cross-platform windowing and keyboard-mouse handler; its API is a superset of the GLUT API, and it is more stable and up to date than GLUT
  - OpenGL Utility Toolkit (GLUT) – An old windowing handler, no longer maintained.
- Several "multimedia libraries" can create OpenGL windows, in addition to input, sound and other tasks useful for game-like applications
  - Allegro 5 – A cross-platform multimedia library with a C API focused on game development
  - Simple DirectMedia Layer (SDL) – A cross-platform multimedia library with a C API
  - SFML – A cross-platform multimedia library with a C++ API and multiple other bindings to languages such as C#, Java, Haskell, and Go
- Widget toolkits

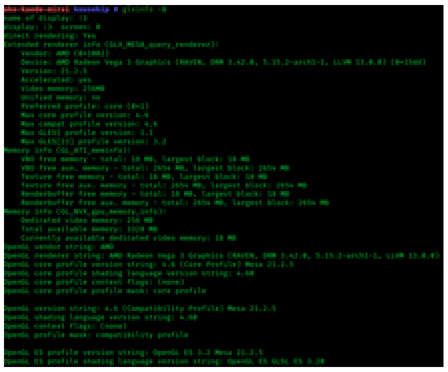
- FLTK – A small cross-platform C++ widget library
- Qt – A cross-platform C++ widget toolkit. It provides many OpenGL helper objects, which even abstract away the difference between desktop GL and OpenGL ES
- wxWidgets – A cross-platform C++ widget toolkit

Extension loading libraries

Given the high workload involved in identifying and loading OpenGL extensions, a few libraries have been designed which load all available extensions and functions automatically. Examples include OpenGL Easy Extension library (GLEE), OpenGL Extension Wrangler Library (GLEW) and glbinding. Extensions are also loaded automatically by most language bindings, such as JOGL and PyOpenGL.

Implementations

Mesa 3D is an open-source implementation of OpenGL. It can do pure software rendering, and it may also use hardware acceleration on BSD, Linux, and other platforms by taking advantage of the Direct Rendering Infrastructure. As of version 20.0, it implements version 4.6 of the OpenGL standard.


Screenshot of `glxinfo`, showing information of Mesa implementation of OpenGL on a system

History

In the 1980s, developing software that could function with a wide range of graphics hardware was a real challenge. Software developers wrote custom interfaces and drivers for each piece of hardware. This was expensive and resulted in multiplication of effort.

By the early 1990s, Silicon Graphics (SGI) was a leader in 3D graphics for workstations. Their IRIS GL API[18][19] became the industry standard, used more widely than the open standards-based PHIGS. This was because IRIS GL was considered easier to use, and because it supported immediate mode rendering. By contrast, PHIGS was considered difficult to use and outdated in functionality.

SGI's competitors (including Sun Microsystems, Hewlett-Packard and IBM) were also able to bring to market 3D hardware supported by extensions made to the PHIGS standard, which pressured SGI to open source a version of IRIS GL as a public standard called **OpenGL**.

However, SGI had many customers for whom the change from IRIS GL to OpenGL would demand significant investment. Moreover, IRIS GL had API functions that were irrelevant to 3D graphics. For example, it included a windowing, keyboard and mouse API, in part because it was developed before the X Window System and Sun's NeWS. And, IRIS GL libraries were unsuitable for opening due to licensing and patent issues. These factors required SGI to continue to support the advanced and proprietary Iris Inventor and Iris Performer programming APIs while market support for OpenGL matured.

One of the restrictions of IRIS GL was that it only provided access to features supported by the underlying hardware. If the graphics hardware did not support a feature natively, then the application could not use it. OpenGL overcame this problem by providing software implementations of features unsupported by hardware, allowing applications to use advanced graphics on relatively low-powered systems. OpenGL standardized access to hardware, pushed the development responsibility of hardware interface programs (device drivers) to hardware manufacturers, and delegated windowing functions to the underlying operating system. With so many different kinds of graphics hardware, getting them all to speak the same language in this way had a remarkable impact by giving software developers a higher-level platform for 3D-software development.

In 1992,[20] SGI led the creation of the OpenGL Architecture Review Board (OpenGL ARB), the group of companies that would maintain and expand the OpenGL specification in the future.

In 1994, SGI played with the idea of releasing something called "OpenGL++" which included elements such as a scene-graph API (presumably based on their Performer technology). The specification was circulated among a few interested parties – but never turned into a product.[21]

In 1996, Microsoft released Direct3D, which eventually became the main competitor of OpenGL. Over 50 game developers signed an open letter to Microsoft, released on June 12, 1997, calling on the company to actively support OpenGL.[22] On December 17, 1997,[23] Microsoft and SGI initiated the Fahrenheit project, which was a joint effort with the goal of unifying the OpenGL and Direct3D interfaces (and adding a scene-graph API too). In 1998, Hewlett-Packard joined the project.[24] It initially showed some promise of bringing order to the world of interactive 3D computer graphics APIs, but on account of financial constraints at SGI, strategic reasons at Microsoft, and a general lack of industry support, it was abandoned in 1999.[25]

In July 2006, the OpenGL Architecture Review Board voted to transfer control of the OpenGL API standard to the Khronos Group.[26][27]

Industry support

In June 2018, Apple deprecated OpenGL APIs on all of their platforms (iOS, macOS and tvOS), strongly encouraging developers to use their proprietary Metal API, which was introduced in 2014.[28]

id Software supported OpenGL in subsequent versions of id Tech game engines starting with the 1996 Quake II engine.[29] In 2016, they released an update for the id Tech 6 that added support for Vulkan, a successor to OpenGL. ID Tech 7 eliminated support for OpenGL.[30]

Version history

The first version of OpenGL, version 1.0, was released on June 30, 1992, by Mark Segal and Kurt Akeley. Since then, OpenGL has occasionally been extended by releasing a new version of the specification. Such releases define a baseline set of features which all conforming graphics cards must support, and against which new extensions can more easily be written. Each new version of OpenGL tends to incorporate several extensions which have widespread support among graphics-card vendors, although the details of those extensions may be changed.

OpenGL version history

Version	Release Date	Features
1.1	1995	Texture objects, Vertex Arrays
1.2	March 16, 1998	3D textures, BGRA and packed pixel formats,[31] introduction of the *imaging subset* useful to image-processing applications
1.2.1	October 14, 1998	A concept of ARB extensions
1.3	August 14, 2001	Multitexturing, multisampling, texture compression
1.4	July 24, 2002	Depth textures, GLSlang[32]
1.5	July 29, 2003	Vertex Buffer Object (VBO), Occlusion Queries[33]
2.0	September 7, 2004	GLSL 1.1, MRT, Non Power of Two textures, Point Sprites,[34] Two-sided stencil[33]
2.1	July 2, 2006	GLSL 1.2, Pixel Buffer Object (PBO), sRGB Textures[33]
3.0	August 11, 2008	GLSL 1.3, Texture Arrays, Conditional rendering, Frame Buffer Object (FBO)[35]
3.1	March 24, 2009	GLSL 1.4, Instancing, Texture Buffer Object, Uniform Buffer Object, Primitive restart[36]
3.2	August 3, 2009	GLSL 1.5, Geometry Shader, Multi-sampled textures[37]
3.3	March 11, 2010	GLSL 3.30, Backports as much function as possible from the OpenGL 4.0 specification
4.0	March 11, 2010	GLSL 4.00, Tessellation on GPU, shaders with 64-bit precision[38]
4.1	July 26, 2010	GLSL 4.10, Developer-friendly debug outputs, compatibility with OpenGL ES 2.0[39]
4.2	August 8, 2011[40]	GLSL 4.20, Shaders with atomic counters, draw transform feedback instanced, shader packing, performance improvements
4.3	August 6, 2012[41]	GLSL 4.30, Compute shaders leveraging GPU parallelism, shader storage buffer objects, high-quality ETC2/EAC texture compression, increased memory security, a multi-application robustness extension, compatibility with OpenGL ES 3.0[42]
4.4	July 22, 2013[43]	GLSL 4.40, Buffer Placement Control, Efficient Asynchronous Queries, Shader Variable Layout, Efficient Multiple Object Binding, Streamlined Porting of Direct3D applications, Bindless Texture Extension, Sparse Texture Extension[43]
4.5	August 11, 2014[12][44]	GLSL 4.50, Direct State Access (DSA), Flush Control, Robustness, OpenGL ES 3.1 API and shader compatibility, DX11 emulation features
4.6	July 31, 2017[7][45]	GLSL 4.60, More efficient geometry processing and shader execution, more information, no error context, polygon offset clamp, SPIR-V, anisotropic filtering

OpenGL 2.0

Release date: September 7, 2004

OpenGL 2.0 was originally conceived by 3Dlabs to address concerns that OpenGL was stagnating and lacked a strong direction.[46] 3Dlabs proposed a number of major additions to the standard. Most of these were, at the time, rejected by the ARB or otherwise never came to fruition in the form that 3Dlabs proposed. However, their proposal for a C-style shading language was eventually completed, resulting in the current formulation of the OpenGL Shading Language (GLSL or GLslang). Like the assembly-like shading languages it was replacing, it allowed replacing the fixed-function vertex and fragment pipe with shaders, though this time written in a C-like high-level language.

The design of GLSL was notable for making relatively few concessions to the limits of the hardware then available. This harked back to the earlier tradition of OpenGL setting an ambitious, forward-looking target for 3D accelerators rather than merely tracking the state of currently available hardware. The final OpenGL 2.0 specification[47] includes support for GLSL.

Longs Peak and OpenGL 3.0

Before the release of OpenGL 3.0, the new revision had the codename Longs Peak. At the time of its original announcement, Longs Peak was presented as the first major API revision in OpenGL's lifetime. It consisted of an overhaul to the way that OpenGL works, calling for fundamental changes to the API.

The draft introduced a change to object management. The GL 2.1 object model was built upon the state-based design of OpenGL. That is, to modify an object or to use it, one needs to bind the object to the state system, then make modifications to the state or perform function calls that use the bound object.

Because of OpenGL's use of a state system, objects must be mutable. That is, the basic structure of an object can change at any time, even if the rendering pipeline is asynchronously using that object. A texture object can be redefined from 2D to 3D. This requires any OpenGL implementations to add a degree of complexity to internal object management.

Under the Longs Peak API, object creation would become atomic, using templates to define the properties of an object which would be created with one function call. The object could then be used immediately across multiple threads. Objects would also be immutable; however, they could have their contents changed and updated. For example, a texture could change its image, but its size and format could not be changed.

To support backwards compatibility, the old state based API would still be available, but no new functionality would be exposed via the old API in later versions of OpenGL. This would have allowed legacy code bases, such as the majority of CAD products, to continue to run while other software could be written against or ported to the new API.

Longs Peak was initially due to be finalized in September 2007 under the name OpenGL 3.0, but the Khronos Group announced on October 30 that it had run into several issues that it wished to address before releasing the specification.[48] As a result, the spec was delayed, and the Khronos Group went into a media blackout until the release of the final OpenGL 3.0 spec.

The final specification proved far less revolutionary than the Longs Peak proposal. Instead of removing all immediate mode and fixed functionality (non-shader mode), the spec included them as deprecated features. The proposed object model was not included, and no plans have been announced to include it in any future revisions. As a result, the API remained largely the same with a few existing extensions being promoted to core functionality.

Among some developer groups this decision caused something of an uproar,[49] with many developers professing that they would switch to DirectX in protest. Most complaints revolved around the lack of communication by Khronos to the development community and multiple features being discarded that were viewed favorably by many. Other frustrations included the requirement of DirectX 10 level hardware to use OpenGL 3.0 and the absence of geometry shaders and instanced rendering as core features.

Other sources reported that the community reaction was not quite as severe as originally presented,[50] with many vendors showing support for the update.[51][52]

OpenGL 3.0

Release date: August 11, 2008

OpenGL 3.0 introduced a deprecation mechanism to simplify future revisions of the API. Certain features, marked as deprecated, could be completely disabled by requesting a *forward-compatible context* from the windowing system. OpenGL 3.0 features could still be accessed alongside these deprecated features, however, by requesting a *full context*.

Deprecated features include:

- All fixed-function vertex and fragment processing
- Direct-mode rendering, using glBegin and glEnd
- Display lists
- Indexed-color rendering targets
- OpenGL Shading Language versions 1.10 and 1.20

OpenGL 3.1

Release date: March 24, 2009

OpenGL 3.1 fully removed all of the features which were deprecated in version 3.0, with the exception of wide lines. From this version onwards, it's not possible to access new features using a *full context*, or to access deprecated features using a *forward-compatible context*. An exception to the former rule is made if the implementation supports the ARB_compatibility (https://www.khronos.org/registry/OpenGL/extensions/ARB/ARB_compatibility.txt) extension, but this is not guaranteed.

Hardware: Mesa supports ARM Panfrost with Version 21.0.

OpenGL 3.2

Release date: August 3, 2009

OpenGL 3.2 further built on the deprecation mechanisms introduced by OpenGL 3.0, by dividing the specification into a *core profile* and *compatibility profile*. Compatibility contexts include the previously removed fixed-function APIs, equivalent to the ARB_compatibility extension released alongside OpenGL 3.1, while core contexts do not. OpenGL 3.2 also included an upgrade to GLSL version 1.50.

OpenGL 3.3

Release date: March 11, 2010

Mesa supports software Driver SWR, softpipe and for older Nvidia cards with NV50.

OpenGL 4.0

Release date: March 11, 2010

OpenGL 4.0 was released alongside version 3.3. It was designed for hardware able to support Direct3D 11.

As in OpenGL 3.0, this version of OpenGL contains a high number of fairly inconsequential extensions, designed to thoroughly expose the abilities of Direct3D 11-class hardware. Only the most influential extensions are listed below.

Hardware support: Nvidia GeForce 400 series and newer, AMD Radeon HD 5000 Series and newer (FP64 shaders implemented by emulation on some TeraScale GPUs), Intel HD Graphics in Intel Ivy Bridge processors and newer.[53]

OpenGL 4.1

Release date: July 26, 2010

Hardware support: Nvidia GeForce 400 series and newer, AMD Radeon HD 5000 Series and newer (FP64 shaders implemented by emulation on some TeraScale GPUs), Intel HD Graphics in Intel Ivy Bridge processors and newer.[53]

- Minimum "maximum texture size" is 16,384 × 16,384 for GPU's implementing this specification.[54]

OpenGL 4.2

Release date: August 8, 2011[40]

- Support for shaders with atomic counters and load-store-atomic read-modify-write operations to one level of a texture
- Drawing multiple instances of data captured from GPU vertex processing (including tessellation), to enable complex objects to be efficiently repositioned and replicated
- Support for modifying an arbitrary subset of a compressed texture, without having to re-download the whole texture to the GPU for significant performance improvements

Hardware support: Nvidia GeForce 400 series and newer, AMD Radeon HD 5000 Series and newer (FP64 shaders implemented by emulation on some TeraScale GPUs), and Intel HD Graphics in Intel Haswell processors and newer.[53] (Linux Mesa: Ivy Bridge and newer)

OpenGL 4.3

Release date: August 6, 2012[41]

- Compute shaders leveraging GPU parallelism within the context of the graphics pipeline
- Shader storage buffer objects, allowing shaders to read and write buffer objects like image load/store from 4.2, but through the language rather than function calls.

- Image format parameter queries
- ETC2/EAC texture compression as a standard feature
- Full compatibility with OpenGL ES 3.0 APIs
- Debug abilities to receive debugging messages during application development
- Texture views to interpret textures in different ways without data replication
- Increased memory security and multi-application robustness

Hardware support: AMD Radeon HD 5000 Series and newer (FP64 shaders implemented by emulation on some TeraScale GPUs), Intel HD Graphics in Intel Haswell processors and newer.[53] (Linux Mesa: Ivy Bridge without stencil texturing, Haswell and newer), Nvidia GeForce 400 series and newer. VIRGL Emulation for virtual machines supports 4.3+ with Mesa 20.

OpenGL 4.4

Release date: July 22, 2013[43]

- Enforced buffer object usage controls
- Asynchronous queries into buffer objects
- Expression of more layout controls of interface variables in shaders
- Efficient binding of multiple objects simultaneously

Hardware support: AMD Radeon HD 5000 Series and newer (FP64 shaders implemented by emulation on some TeraScale GPUs), Intel HD Graphics in Intel Broadwell processors and newer (Linux Mesa: Haswell and newer),[55] Nvidia GeForce 400 series and newer,[56] Tegra K1.

OpenGL 4.5

Release date: August 11, 2014[12][44]

- Direct State Access (DSA) – object accessors enable state to be queried and modified without binding objects to contexts, for increased application and middleware efficiency and flexibility.[57]
- Flush Control – applications can control flushing of pending commands before context switching – enabling high-performance multithreaded applications;
- Robustness – providing a secure platform for applications such as WebGL browsers, including preventing a GPU reset affecting any other running applications;
- OpenGL ES 3.1 API and shader compatibility – to enable the easy development and execution of the latest OpenGL ES applications on desktop systems.

Hardware support: AMD Radeon HD 5000 Series and newer (FP64 shaders implemented by emulation on some TeraScale GPUs), Intel HD Graphics in Intel Broadwell processors and newer (Linux Mesa: Haswell and newer), Nvidia GeForce 400 series and newer,[56] Tegra K1, and Tegra X1.[58][59]

OpenGL 4.6

Release date: July 31, 2017[12][7][45]

- more efficient, GPU-sided, geometry processing

- more efficient shader execution (AZDO)
- more information through statistics, overflow query and counters
- higher performance through no error handling contexts
- clamping of polygon offset function, solves a shadow rendering problem
- SPIR-V shaders
- Improved anisotropic filtering

Hardware support: AMD Radeon HD 7000 Series and newer (FP64 shaders implemented by emulation on some TeraScale GPUs), Intel Haswell and newer, Nvidia GeForce 400 series and newer.[56]

Driver support:

- Mesa 19.2 on Linux supports OpenGL 4.6 for Intel Broadwell and newer.[60] Mesa 20.0 supports AMD Radeon GPUs,[61] while support for Nvidia Kepler+ is in progress. Zink as Emulation Driver with 21.1 and software driver LLVMpipe also support with Mesa 21.0.
- AMD Adrenalin 18.4.1 Graphics Driver on Windows 7 SP1, 10 version 1803 (April 2018 update) for AMD Radeon™ HD 7700+, HD 8500+ and newer. Released April 2018.[62][63]
- Intel 26.20.100.6861 graphics driver on Windows 10. Released May 2019.[64][65]
- NVIDIA GeForce 397.31 Graphics Driver on Windows 7, 8, 10 x86-64 bit only, no 32-bit support. Released April 2018[66]

Alternative implementations

Apple deprecated OpenGL in iOS 12 and macOS 10.14 Mojave in favor of Metal, but it is still available as of macOS 13 Ventura (including Apple silicon devices).[67] The latest version supported for OpenGL is 4.1 from 2011.[68][69] A proprietary library from Molten – authors of MoltenVK – called MoltenGL, can translate OpenGL calls to Metal.[70]

There are several projects which attempt to implement OpenGL on top of Vulkan. The Vulkan backend for Google's ANGLE achieved OpenGL ES 3.1 conformance in July 2020.[71] The Mesa3D project also includes such a driver, called *Zink*.[72]

Vulkan

Vulkan, formerly named the "Next Generation OpenGL Initiative" (glNext),[73][74] is a grounds-up redesign effort to unify OpenGL and OpenGL ES into one common API that will not be backwards compatible with existing OpenGL versions.[75][76][77]

The initial version of Vulkan API was released on February 16, 2016.

See also

Free and open-source software portal

- ARB assembly language – OpenGL's legacy low-level shading language
- Comparison of OpenGL and Direct3D
- Direct3D – main competitor of OpenGL

- Glide (API) – a graphics API once used on 3dfx Voodoo cards
- List of OpenGL applications
- Metal (API) – a graphics API for iOS, macOS, tvOS, watchOS
- OpenAL – cross-platform audio library, designed to resemble OpenGL
- OpenGL ES – OpenGL for embedded systems
- OpenSL ES – API for audio on embedded systems, developed by the Khronos Group
- OpenVG – API for accelerated 2D graphics, developed by the Khronos Group
- RenderMan Interface Specification (RISpec) – Pixar's open API for photorealistic off-line rendering
- VOGL – a debugger for OpenGL
- Vulkan – low-overhead, cross-platform 2D and 3D graphics API, the "next generation OpenGL initiative"
- Graphics pipeline
- WebGPU

References

1. Lextrait, Vincent (January 2010). "The Programming Languages Beacon, v10.0" (https://archive.today/20120530/http://www.lextrait.com/Vincent/implementations.html). Archived from the original (http://www.lextrait.com/Vincent/implementations.html) on May 30, 2012. Retrieved March 14, 2010.
2. "Products: Software: OpenGL: Licensing and Logos" (https://web.archive.org/web/20121101073722/http://www.sgi.com/products/software/opengl/license.html). SGI. Archived from the original (http://www.sgi.com/products/software/opengl/license.html) on November 1, 2012. Retrieved November 7, 2012.
3. "The OpenGL Graphics System: A Specification" (https://www.khronos.org/registry/OpenGL/specs/gl/glspec40.core.pdf) (PDF). 4.0 (Core Profile). March 11, 2010.
4. "SGI – OpenGL Overview" (http://www.sgi.com/products/software/opengl/overview.html). Archived (https://web.archive.org/web/20041031094901/http://www.sgi.com/products/software/opengl/overview.html) from the original on October 31, 2004. Retrieved February 16, 2007.
5. Peddie, Jon (July 2012). "Who's the Fairest of Them All?" (http://www.cgw.com/Publications/CGW/2012/Volume-35-Issue-4-June-July-2012/Who-s-the-Fairest-of-Them-All-.aspx). Computer Graphics World. Retrieved May 30, 2018.
6. "OpenGL ARB to Pass Control of OpenGL Specification to Khronos Group" (https://www.khronos.org/news/press/opengl_arb_to_pass_control_of_opengl_specification_to_khronos_group). *The Khronos Group*. July 31, 2006. Retrieved March 18, 2021.
7. "Khronos Releases OpenGL 4.6 with SPIR-V Support" (https://www.khronos.org/news/press/khronos-releases-opengl-4.6-with-spir-v-support). The Khronos Group Inc. July 31, 2017. Retrieved July 31, 2017.
8. "The Future of OpenGL (forum discussion)" (https://community.khronos.org/t/the-future-of-opengl/106317). *Khronos Group*. 2020.
9. "Khronos News Archives" (https://www.khronos.org/news/archives). *Khronos Group*. November 28, 2022.
10. "Khronos Blog" (https://www.khronos.org/blog/). *Khronos Group*. November 28, 2022.
11. "Khronos Membership Overview and FAQ" (https://www.khronos.org/members/). Khronos.org. Retrieved November 7, 2012.
12. "Khronos OpenGL Registry" (https://khronos.org/registry/OpenGL/index_gl.php). Khronos Group. Retrieved July 31, 2017.

13. "NV_half_float" (https://www.khronos.org/registry/OpenGL/extensions/NV/NV_half_float.txt). *OpenGL Registry*. Khronos Group.
14. "How to Create Khronos API Extensions" (https://khronos.org/registry/OpenGL/docs/rules.html). Khronos Group. Retrieved July 31, 2017.
15. "OpenGL - The Industry's Foundation for High Performance Graphics" (https://www.khronos.org/opengl/). *The Khronos Group*. July 19, 2011. Retrieved March 18, 2021.
16. "A list of GLUT alternatives, maintained by" (https://www.opengl.org/resources/libraries/windowtoolkits/). Khronos Group. Retrieved May 2, 2013.
17. "Related toolkits and APIs" (https://www.opengl.org/wiki/Related_toolkits_and_APIs#Context.2FWindow_Toolkits). *www.opengl.org*. OpenGL. Retrieved October 8, 2014.
18. "IRIS GL, SGI's property" (http://www.cg.tuwien.ac.at/~wimmer/apis/API_Summary.html).
19. Kilgard, Mark (2008). "OpenGL Prehistory: IRIS GL (slide)" (https://www.slideshare.net/Mark_Kilgard/sigraph-asia-2008-modern-opengl-presentation/13-13OpenGLs_PrehistoryIRIS_GL_1Window_system). *www.slideshare.net*.
20. "Creation of the OpenGL ARB" (https://web.archive.org/web/20070222123208/http://www.sgi.com/company_info/newsroom/press_releases/2004/august/opengl.html). Archived from the original (http://www.sgi.com/company_info/newsroom/press_releases/2004/august/opengl.html) on February 22, 2007. Retrieved February 16, 2007.
21. "End of OpenGL++" (https://www.opengl.org/archives/about/arb/meeting_notes/notes/Meeting1.2/meeting_note_10-03-98.html). Khronos Group.
22. "Top Game Developers Call on Microsoft to Actively Support OpenGL" (https://archive.org/details/NEXT_Generation_32/page/n18). *Next Generation*. No. 32. Imagine Media. August 1997. p. 17.
23. "Announcement of Fahrenheit" (https://web.archive.org/web/20070927212603/http://www.windowsitpro.com/Article/ArticleID/17533/17533.html). Archived from the original (http://www.windowsitpro.com/Article/ArticleID/17533/17533.html) on September 27, 2007.
24. "Members of Fahrenheit. 1998" (https://web.archive.org/web/20071005013207/http://findarticles.com/p/articles/mi_m0CGN/is_n3341/ai_20211297). *Computergram International*. 1998. Archived from the original (http://www.findarticles.com/p/articles/mi_m0CGN/is_n3341/ai_20211297) on October 5, 2007.
25. "End of Fahrenheit" (https://www.theregister.co.uk/1999/11/29/ms_quietly_dumps_windows_opengl/). *The Register*.
26. "OpenGL ARB to pass control of OpenGL specification to Khronos Group" (https://www.khronos.org/news/press/opengl_arb_to_pass_control_of_opengl_specification_to_khronos_group). Khronos press release. July 31, 2006.
27. "OpenGL ARB to Pass Control of OpenGL Specification to Khronos Group" (http://www.accessmylibrary.com/coms2/summary_0286-16157838_ITM). AccessMyLibrary Archive.
28. Smith, Ryan (June 5, 2018). "Apple Deprecates OpenGL Across All OSes; Urges Developers to use Metal" (https://www.anandtech.com/show/12894/apple-deprecates-opengl-across-all-oses). *www.anandtech.com*. Purch. Retrieved June 5, 2018.
29. "Technology Licensing: id Tech 2" (https://web.archive.org/web/20091108191715/http://www.idsoftware.com/business/idtech2/). Archived from the original (http://www.idsoftware.com/business/idtech2/) on November 8, 2009. Retrieved September 17, 2008.
30. "Doom Wiki: id Tech 7" (https://doomwiki.org/wiki/Id_Tech_7). Retrieved October 26, 2021.
31. Astle, Dave (April 1, 2003). "Moving Beyond OpenGL 1.1 for Windows" (https://www.gamedev.net/articles/programming/graphics/moving-beyond-opengl-11-for-windows-r1929/). *gamedev.net*. Retrieved November 15, 2007.

32. Isorna, J.M. (2015). *Simulación visual de materiales : teoría, técnicas, análisis de casos* (https://books.google.com/books?id=npmdCwAAQBAJ&pg=PA191). UPC Grau. Arquitectura, urbanisme i edificació (in Spanish). Universitat Politècnica de Catalunya. p. 191. ISBN 978-84-9880-564-2. Retrieved August 21, 2019.
33. "The OpenGL Graphics System: A Specification" (https://www.khronos.org/registry/OpenGL/specs/gl/glspec21.pdf) (PDF). 2.1. December 1, 2006.
34. "Point Primitive" (https://www.khronos.org/opengl/wiki/Primitive#Point_primitives).
35. "The OpenGL Graphics System: A Specification" (https://www.khronos.org/registry/OpenGL/specs/gl/glspec30.pdf) (PDF). 3.0. September 23, 2008.
36. "The OpenGL Graphics System: A Specification" (https://www.khronos.org/registry/OpenGL/specs/gl/glspec31.pdf) (PDF). 3.1. May 28, 2009.
37. "The OpenGL Graphics System: A Specification" (https://www.khronos.org/registry/OpenGL/specs/gl/glspec32.core.pdf) (PDF). 3.2 (Core Profile). December 7, 2009.
38. "Khronos Unleashes Cutting-Edge, Cross-Platform Graphics Acceleration with OpenGL 4.0" (https://www.khronos.org/news/press/khronos-unleashes-cutting-edge-cross-platform-graphics-acceleration-opengl4). March 11, 2010.
39. "Khronos Drives Evolution of Cross-Platform 3D Graphics with Release of OpenGL 4.1 Specification" (https://www.khronos.org/news/press/opengl-4-1-released). July 26, 2010.
40. "Khronos Enriches Cross-Platform 3D Graphics with Release of OpenGL 4.2 Specification" (https://www.khronos.org/news/press/khronos-enriches-cross-platform-3d-graphics-with-release-of-opengl-4.2-spec). August 8, 2011.
41. "Khronos Releases OpenGL 4.3 Specification with Major Enhancements" (https://www.khronos.org/news/press/khronos-releases-opengl-4.3-specification-with-major-enhancements). August 6, 2012.
42. "Khronos Releases OpenGL 4.3 Specification with Major Enhancements" (https://www.khronos.org/news/press/khronos-releases-opengl-4.3-specification-with-major-enhancements). August 6, 2012.
43. "Khronos Releases OpenGL 4.4 Specification" (https://www.khronos.org/news/press/khronos-releases-opengl-4.4-specification). July 22, 2013.
44. "Khronos Group Announces Key Advances in OpenGL Ecosystem – Khronos Group Press Release" (https://www.khronos.org/news/press/khronos-group-announces-key-advances-in-opengl-ecosystem). The Khronos Group Inc. August 10, 2014. Retrieved April 17, 2015.
45. Kessenich, John; Baldwin, Dave. "The OpenGL® Shading Language, Version 4.60.7" (https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.4.60.html). *The Khronos Group Inc*. Retrieved August 21, 2019.
46. Abi-Chahla, Fedy (September 16, 2008). "OpenGL 3 (3DLabs And The Evolution Of OpenGL)" (http://www.tomshardware.com/reviews/opengl-directx,2019-2.html). Tom's Hardware. Retrieved October 24, 2010.
47. "The OpenGL Graphics System: A Specification" (https://www.khronos.org/registry/OpenGL/specs/gl/glspec20.pdf) (PDF). 2.0. October 22, 2004.
48. "OpenGL ARB announces an update on OpenGL 3.0" (http://www.opengl.org/discussion_boards/ubbthreads.php?ubb=showflat&Number=229374#Post229374). October 30, 2007. Retrieved October 31, 2007.
49. "OpenGL 3.0 Released, Developers Furious – Slashdot" (http://tech.slashdot.org/article.pl?sid=08/08/11/2135259). Tech.slashdot.org. Retrieved November 7, 2012.
50. "OpenGL BOF went over well, no pitch forks seen" (https://www.opengl.org/news/opengl_bof_went_over_well_no_pitch_forks_seen).

51. "The Industry Standard for High Performance Graphics" (https://www.opengl.org/news/nick_haemel_amd_blog_post_opengl_30_a_big_step_in_the_right_direction/). OpenGL. August 18, 2008. Retrieved July 31, 2017.
52. "NVIDIA provides early OpenGL 3.0 driver now" (https://www.opengl.org/news/nvidia_provides_early_opengl_30_driver_now).
53. "Intel Iris and HD Graphics Driver for Windows 7/8/8.1 64bit" (https://web.archive.org/web/20150402105758/https://downloadcenter.intel.com/download/24785). *Intel Download Center*. Archived from the original (https://downloadcenter.intel.com/download/24785) on April 2, 2015.
54. "Expected maximum texture size - Graphics and GPU Programming" (http://www.gamedev.net/topic/646362-expected-maximum-texture-size/). *GameDev.net*.
55. "Intel Skylake-S CPUs and 100-series Chipsets Detailed in Apparent Leak" (http://gadgets.ndtv.com/laptops/news/intel-skylake-s-cpus-and-100-series-chipsets-detailed-in-apparent-leak-682437). *NDTV Gadgets*. April 17, 2015.
56. Larabel, Michael (July 31, 2017). "NVIDIA Releases 381.26.11 Linux Driver With OpenGL 4.6 Support" (https://www.phoronix.com/scan.php?page=news_item&px=NVIDIA-OpenGL-4.6-Driver). *Phoronix*.
57. "OpenGL 4.5 released—with one of Direct3D's best features" (https://arstechnica.com/information-technology/2014/08/opengl-4-5-released-with-one-of-direct3ds-best-features/). *Ars Technica*. August 11, 2014. Retrieved April 17, 2015.
58. "SG4121: OpenGL Update for NVIDIA GPUs" (https://web.archive.org/web/20150517205154/http://www.ustream.tv/recorded/51255959). *Ustream*. Archived from the original (http://www.ustream.tv/recorded/51255959) on May 17, 2015. Retrieved April 17, 2015.
59. Kilgard, Mark (August 12, 2014). "OpenGL 4.5 Update for NVIDIA GPUs" (https://www.slideshare.net/Mark_Kilgard/opengl-45-update-for-nvidia-gpus). Retrieved April 17, 2015.
60. Larabel, Michael (August 21, 2019). "Intel's OpenGL Linux Driver Now Has OpenGL 4.6 Support For Mesa 19.2" (https://www.phoronix.com/scan.php?page=news_item&px=OpenGL-4.6-Mesa-19.2-Intel). *Phoronix*.
61. Larabel, Michael (November 27, 2019). "AMD's RadeonSI Driver Finally Enables OpenGL 4.6" (https://www.phoronix.com/scan.php?page=news_item&px=RadeonSI-GL-4.6-NIR-Lands). *Phoronix*.
62. "AMD Adrenalin 18.4.1 Graphics Driver Released (OpenGL 4.6, Vulkan 1.1.70) – Geeks3D" (http://www.geeks3d.com/20180501/amd-adrenalin-18-4-1-graphics-driver-released-opengl-4-6-vulkan-1-1-70/). *www.geeks3d.com*. Retrieved May 10, 2018.
63. "Radeon™ Software Adrenalin Edition 18.4.1 Release Notes" (https://support.amd.com/en-us/kb-articles/Pages/Radeon-Software-Adrenalin-Edition-18.4.1-Release-Notes.aspx). *support.amd.com*. Retrieved May 10, 2018.
64. "Intel Graphics Driver 25.20.100.6861 Released (OpenGL 4.6 + Vulkan 1.1.103) | Geeks3D" (https://www.geeks3d.com/20190516/intel-graphics-driver-25-20-100-6861-released-opengl-4-6-vulkan-1-1-103/). Retrieved May 16, 2019.
65. "Windows® 10 DCH Drivers" (https://downloadcenter.intel.com/download/28783/Intel-Graphics-Windows-10-DCH-Drivers). *Intel DownloadCenter*. Retrieved August 21, 2019.
66. "NVIDIA GeForce 397.31 Graphics Driver Released (OpenGL 4.6, Vulkan 1.1, RTX, CUDA 9.2) – Geeks3D" (http://www.geeks3d.com/20180425/nvidia-geforce-397-31-graphics-driver-released-opengl-4-6-vulkan-1-1-rtx-cuda-9-2/). *www.geeks3d.com*. Retrieved May 10, 2018.
67. "Apple Developer Documentation" (https://developer.apple.com/documentation/apple-silicon/porting-your-macos-apps-to-apple-silicon). *developer.apple.com*.
68. Cunningham, Andrew (October 7, 2019). "macOS 10.15 Catalina: The Ars Technica review" (https://arstechnica.com/gadgets/2019/10/macos-10-15-catalina-the-ars-technica-review/3/#h2). *Ars Technica*.

69. Axon, Samuel (June 6, 2018). "The end of OpenGL support, plus other updates Apple didn't share at the keynote" (https://arstechnica.com/gadgets/2018/06/the-end-of-opengl-support-other-updates-apple-didnt-share-at-the-keynote/). *Ars Technica*. Retrieved October 19, 2020.
70. "Vulkan, and faster OpenGL ES, on iOS and macOS" (https://moltengl.com/). *Molten*. Retrieved October 19, 2020.
71. The ANGLE Project Authors (October 14, 2020). "google/angle: A conformant OpenGL ES implementation for Windows, Mac, Linux, iOS and Android" (https://github.com/google/angle). *GitHub*. Retrieved December 17, 2020.
72. "Zink" (https://docs.mesa3d.org/gallium/drivers/zink.html). *The Mesa 3D Graphics Library latest documentation*.
73. Dingman, Hayden (March 3, 2015). "Meet Vulkan, the powerful, platform-agnostic gaming tech taking aim at DirectX 12" (http://www.pcworld.com/article/2891613/meet-vulkan-the-powerful-platform-agnostic-gaming-tech-taking-aim-at-directx-12.html). *PC World*. Retrieved March 3, 2015.
74. Bright, Peter (March 3, 2015). "Khronos unveils Vulkan: OpenGL built for modern systems" (https://arstechnica.com/gadgets/2015/03/khronos-unveils-vulkan-opengl-built-for-modern-systems/). *Ars Technica*. Retrieved March 3, 2015.
75. "Khronos Announces Next Generation OpenGL Initiative" (http://www.anandtech.com/show/8363/khronos-announces-next-generation-opengl-initiative). AnandTech. Retrieved August 20, 2014.
76. "OpenGL 4.5 released, next-gen OpenGL unveiled: Cross-platform Mantle killer, DX12 competitor" (https://www.extremetech.com/gaming/187796-opengl-4-5-released-next-gen-opengl-unveiled-cross-platform-mantle-killer-dx12-competitor). Retrieved August 20, 2014.
77. "Khronos Publishes Its Slides About OpenGL-Next" (https://www.phoronix.com/scan.php?page=news_item&px=MTc2ODQ). Phoronix. Retrieved August 22, 2014.

Further reading

- Shreiner, Dave; Sellers, Graham; et al. (March 30, 2013). *OpenGL Programming Guide: The Official Guide to Learning OpenGL*. Version 4.3 (8th ed.). Addison-Wesley. ISBN 978-0-321-77303-6.
- Sellers, Graham; Wright, Richard S.; Haemel, Nicholas (July 31, 2013). *OpenGL SuperBible: Comprehensive Tutorial and Reference* (6th ed.). Addison-Wesley. ISBN 978-0-321-90294-8.
- Rost, Randi J. (July 30, 2009). *OpenGL Shading Language* (3rd ed.). Addison-Wesley. ISBN 978-0-321-63763-5.
- Lengyel, Eric (2003). *The OpenGL Extensions Guide* (https://archive.org/details/openglextensions0000leng). Charles River Media. ISBN 1-58450-294-0.
- OpenGL Architecture Review Board; Shreiner, Dave (2004). *OpenGL Reference Manual: The Official Reference Document to OpenGL*. Version 1.4. Addison-Wesley. ISBN 0-321-17383-X.
- OpenGL Architecture Review Board; Shreiner, Dave; et al. (2006). *OpenGL Programming Guide: The Official Guide to Learning OpenGL* (https://archive.org/details/openglprogrammin0000unse). Version 2 (5th ed.). Addison-Wesley. ISBN 0-321-33573-2.

External links

- Official website (https://www.opengl.org/)
- OpenGL Overview (https://www.khronos.org/opengl/) and OpenGL.org's Wiki (https://www.khronos.org/opengl/wiki/) with more information on OpenGL Language bindings

- SGI's OpenGL website (https://web.archive.org/web/20161112121902/http://www.sgi.com/tech/opengl/)
- OpenGL (https://curlie.org/Computers/Programming/Graphics/Libraries/OpenGL) at Curlie
- Khronos Group, Inc. (https://www.khronos.org/)
