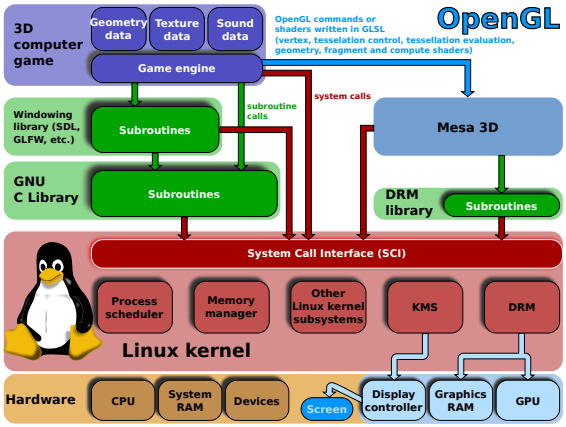
OpenGL

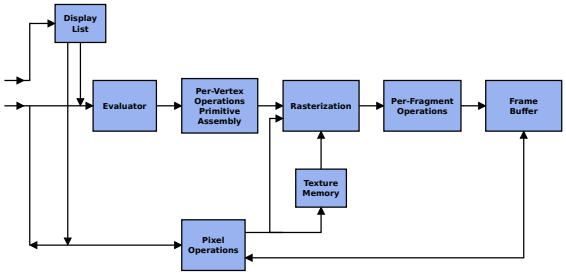


Video games outsource real-time rendering calculations to the GPU over OpenGL. The rendered results are not sent back, and are instead stored in a framebuffer whose content is sent to the display controller.

OpenGL (**Open Graphics Library**)[3][4][5] is a cross-language, multi-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) started developing OpenGL in 1991 and released it in January 1992;[6] applications use it extensively in the fields of CAD, virtual reality, scientific visualization, information visualization, flight simulation, and video games. OpenGL is managed by the non-profit technology consortium Khronos Group.

1 Design



An illustration of the graphics pipeline process

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 number of functions which may be called by the client program, alongside a number 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 C programming language, 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 platform-independent. 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.

1.1 Development

OpenGL is an evolving API. New versions of the OpenGL specifications are regularly 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.[7]

In addition to the features required by the core API, 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.[8]

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().[9] 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.[10]

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.

2 Documentation

OpenGL's popularity is partially due to the quality of its official 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 almost universally known by the colors of their covers:

The Red Book OpenGL Programming Guide, 8th Edition. ISBN 0-321-77303-9

A tutorial and reference book.

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 GLUT.

The Blue Book OpenGL Reference manual, 4th edition. ISBN 0-321-17383-X

Essentially a hard-copy printout of the 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.

3 Associated libraries

The earliest versions of OpenGL were released with a companion library called GLU, the OpenGL Utility Library. It provided simple, useful features which were unlikely to be supported in contemporary hardware, such as mipmap generation, tessellation, and generation of primitive shapes. The GLU specification was last updated in 1998, and the latest version depends on features which were deprecated with the release of OpenGL 3.1 in 2009.[8]

3.1 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 GLUT (later superseded by freeglut). GLFW is a newer alternative.[11]

- These toolkits are designed specifically around creating and managing OpenGL windows. They also manage input, but little beyond that.[12]
  - GLFW — A crossplatform windowing and keyboard/mouse/joystick handler. Is more aimed for creating games.
  - freeglut — A crossplatform 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.
  - 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.
  - SDL — A cross-platform multimedia library with a C API.
  - SFML — A cross-platform multimedia library with a C++ API.
- Widget toolkits
  - FLTK — A small cross-platform C++ widget library.

- Qt — A cross-platform C++ widget toolkit. It provides a number of OpenGL helper objects, which even abstract away the difference between desktop GL and OpenGL ES.
- wxWidgets — A cross-platform C++ widget toolkit.

3.2 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 GLEE, GLEW and glbinding. Extensions are also loaded automatically by most language bindings, such as JOGL and PyOpenGL.

3.3 Implementations

Mesa 3D is an open source implementation of OpenGL. It can do pure software rendering, and it may also use hardware acceleration on the Linux platform by taking advantage of the Direct Rendering Infrastructure. As of version 10.0, it implements version 3.3 of the OpenGL standard.[13]

4 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, SGI was a leader in 3D graphics for workstations. Their IRIS GL API[14] was considered state-of-the-art and became the de facto industry standard, overshadowing 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 terms of 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. This in turn caused SGI market share to weaken as more 3D graphics hardware suppliers entered the market. In an effort to influence the market, SGI decided to turn the IrisGL API into an open standard – **OpenGL**.

However, SGI had a large number of software customers for whom the change from IrisGL to OpenGL would require significant investment. Moreover, IrisGL had API functions that were not relevant 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 systems were developed. And, IrisGL 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 IrisGL was that it only provided access to features supported by the underlying hardware. If the graphics hardware did not support a feature, then the application could not use it. OpenGL overcame this problem by providing support in software for 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 (sometimes called 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,[15] 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.[16]

Microsoft released Direct3D in 1995, which eventually became the main competitor of OpenGL. On December 17, 1997,[17] 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.[18] 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 general lack of industry support, it was abandoned in 1999.[19]

In July 2006 the OpenGL Architecture Review Board voted to transfer control of the OpenGL API standard to the Khronos Group.[20][21]

5 Version history

The first version of OpenGL, version 1.0, was released in January 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 a number of extensions which have widespread support among graphics-card vendors, although the details of those extensions may be changed.

5.1 OpenGL 1.1

Release Date: March 4, 1997

5.2 OpenGL 1.2

Release Date: March 16, 1998

One notable feature of OpenGL 1.2 was the introduction of the *imaging subset*. This is a set of features which are very useful to image-processing applications, but which have limited usefulness elsewhere. Implementation of this subset has always been optional; support is indicated by advertising the extension string ARB_imaging.

5.2.1 OpenGL 1.2.1

Release Date: October 14, 1998

OpenGL 1.2.1 was a minor release, appearing only seven months after the release of version 1.2. It introduced the concept of ARB extensions, and defined the extension ARB_multitexture, without yet incorporating it into the OpenGL core specification.

5.3 OpenGL 1.3

Release Date: August 14, 2001

5.4 OpenGL 1.4

Release date: July 24, 2002

5.5 OpenGL 1.5

Release Date: July 29, 2003

Alongside the release of OpenGL 1.5, the ARB released the OpenGL Shading Language specification, and the extensions ARB_shader_objects, ARB_vertex_shader, and ARB_fragment_shader. However, these would not be incorporated into the core specification until the next release.

5.6 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.[22] 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 GLSL (the OpenGL Shading Language or GLslang). Like the assembly-like shading languages that it was replacing, it allowed the programmer to replace 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 limitations of the hardware then available; this hearkened 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[23] includes support for GLSL.

5.7 OpenGL 2.1

Release Date: July 2, 2006

OpenGL 2.1 required implementations to support version 1.20 of the OpenGL Shading Language.

5.8 Longs Peak and OpenGL 3.0

Prior to the release of OpenGL 3.0, the new revision was known as 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, in order 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 a single 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.[24] 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,[25] 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 in order 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,[26] with many vendors showing support for the update.[27][28]

5.9 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.

5.10 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 extension, but this is not guaranteed.

5.11 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.

5.12 OpenGL 3.3

Release Date: March 11, 2010

OpenGL 3.3 was released alongside version 4.0. It was designed to target hardware capable of supporting Direct3D 10.

5.13 OpenGL 4.0

Release Date: March 11, 2010

OpenGL 4.0 was released alongside version 3.3. It was designed to target hardware capable of supporting Direct3D 11.

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

Hardware support: Nvidia GeForce 400 series and newer, ATI Radeon HD 5000 Series and newer, Intel HD Graphics in Intel Ivy Bridge processors and newer.[29]

5.14 OpenGL 4.1

Release Date: July 26, 2010

Hardware support: Nvidia GeForce 400 series and newer, ATI Radeon HD 5000 Series and newer, Intel HD Graphics in Intel Ivy Bridge processors and newer.[29]

5.15 OpenGL 4.2

Release Date: August 8, 2011[30]

- Support for shaders with atomic counters and load/store/atomic read-modify-write operations to a single 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, ATI Radeon HD 5000 Series and newer, Intel HD Graphics in Intel Haswell processors and newer.[29]

5.16 OpenGL 4.3

Release Date: August 6, 2012[31]

- 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 capabilities to receive debugging messages during application development
- Texture views for interpreting textures in different ways without data replication
- Increased memory security and multi-application robustness

Hardware support: Nvidia GeForce 400 series and newer, ATI Radeon HD 5000 Series and newer, Intel HD Graphics in Intel Haswell processors and newer.[29]

5.17 OpenGL 4.4

Release Date: July 22, 2013[32]

- 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: Nvidia GeForce 400 series and newer, AMD Radeon HD 5000 Series and newer, Intel HD Graphics in Intel Skylake processors and newer,[33] Tegra K1.

5.18 OpenGL 4.5

Release Date: August 11, 2014[34][8]

- 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.[35]
- 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: Nvidia GeForce 400 series and newer, as well as the Tegra K1 and Tegra X1.[36][37]

6 Successor

Main article: Vulkan (API)

Vulkan, previously known as glNext or the "Next Generation OpenGL Initiative",[38][39] 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.[40][41][42] AMD offered its Mantle API without any conditions as the foundation of Vulkan.[43][44]

A 2015 GDC conference session by Valve Corporation, with participation from Electronic Arts, Epic Games and Unity Technologies, unveiled Vulkan.[45]

On 3 March 2015, Valve announced Source 2 which will be compatible with Vulkan.[46]

7 See also

- ARB assembly language – OpenGL's legacy low-level shading language
- Comparison of OpenGL and Direct3D
- Glide API – a graphics API once used on 3dfx Voodoo cards
- List of OpenGL programs
- 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-level, cross-platform 2D and 3D graphics API, the "next generation OpenGL initiative"

8 Further reading

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

9 References

[1] Lextrait, Vincent (January 2010). "The Programming Languages Beacon, v10.0". Retrieved 14 March 2010.

[2] "Products: Software: OpenGL: Licensing and Logos". SGI. Retrieved November 7, 2012.

[3] https://www.opengl.org/wiki/Main_Page

[4] "OpenGL 4.0 Specification" (PDF). Retrieved 2013-05-02.

[5] http://faculty.ycp.edu/~{}dbabcock/PastCourses/cs370/labs/index.html

[6] "SGI – OpenGL Overview".

[7] "Khronos Membership Overview and FAQ". Khronos.org. Retrieved November 7, 2012.

[8] "The OpenGL Registry". Opengl.org. Retrieved 2013-05-02.

[9] http://www.opengl.org/registry/specs/NV/half_float.txt

[10] "How to Create Khronos API Extensions". Opengl.org. August 13, 2006. Retrieved November 7, 2012.

[11] "A list of GLUT alternatives, maintained by". Opengl.org. Retrieved 2013-05-02.

[12] "Related toolkits and APIs". *www.opengl.org*. OpenGL. Retrieved 8 October 2014.

[13] "Mesa 10.0 Release Notes". mesa3d.org. 2013-11-30. Retrieved 2013-12-05.

[14] "IRIS GL, SGI's property".

[15] "Creation of the OpenGL ARB".

[16] "End of OpenGL++". opengl.org.

[17] "Announcement of Fahrenheit".

[18] "Members of Fahrenheit. 1998.". *Computergram International*. 1998.

[19] "End of Fahrenheit".

[20] OpenGL ARB to pass control of OpenGL specification to Khronos Group, Khronos press release

[21] OpenGL ARB to Pass Control of OpenGL Specification to Khronos Group, AccessMyLibrary Archive

[22] Fedy Abi-Chahla (September 16, 2008). "OpenGL 3 (3DLabs And The Evolution Of OpenGL)". Tom's Hardware. Retrieved October 24, 2010.

[23] http://www.opengl.org/documentation/specs/version2.0/glspec20.pdf

[24] "OpenGL ARB announces an update on OpenGL 3.0". October 30, 2007. Retrieved October 31, 2007.

[25] "OpenGL 3.0 Released, Developers Furious – Slashdot". Tech.slashdot.org. Retrieved November 7, 2012.

[26] "OpenGL BOF went over well, no pitch forks seen".

[27] "The Industry Standard for High Performance Graphics". OpenGL. August 18, 2008. Retrieved November 7, 2012.

[28] "NVIDIA provides early OpenGL 3.0 driver now".

[29] "Intel Iris and HD Graphics Driver for Windows 7/8/8.1 64bit". *Intel® Download Center*.

[30] "Khronos Enriches Cross-Platform 3D Graphics with Release of OpenGL 4.2 Specification".

[31] "Khronos Releases OpenGL 4.3 Specification with Major Enhancements".

[32] "Khronos Releases OpenGL 4.4 Specification".

[33] "Intel Skylake-S CPUs and 100-series Chipsets Detailed in Apparent Leak". *NDTV Gadgets*. 17 April 2015.

[34] "Khronos Group Announces Key Advances in OpenGL Ecosystem – Khronos Group Press Release". Retrieved April 17, 2015.

[35] "OpenGL 4.5 released—with one of Direct3D's best features". *Ars Technica*. Retrieved April 17, 2015.

[36] "SG4121: OpenGL Update for NVIDIA GPUs". *Ustream*. Retrieved April 17, 2015.

[37] Mark Kilgard. "OpenGL 4.5 Update for NVIDIA GPUs". Retrieved April 17, 2015.

[38] Dingman, Hayden (3 March 2015). "Meet Vulkan, the powerful, platform-agnostic gaming tech taking aim at DirectX 12". *PC World*. Retrieved 3 March 2015.

[39] Bright, Peter (3 March 2015). "Khronos unveils Vulkan: OpenGL built for modern systems". *Ars Technica*. Retrieved 3 March 2015.

[40] "Khronos Announces Next Generation OpenGL Initiative". AnandTech. Retrieved 20 August 2014.

[41] "OpenGL 4.5 released, next-gen OpenGL unveiled: Cross-platform Mantle killer, DX12 competitor". Retrieved 20 August 2014.

[42] "Khronos Publishes Its Slides About OpenGL-Next". Phoronix. Retrieved 22 August 2014.

[43] "AMD hopes to put a little Mantle in OpenGL Next". techreport.

[44] Koduri, Raja (4 March 2015). "On APIs and the future of Mantle". AMD. Retrieved 19 May 2015. ...(T)he Khronos Group has selected Mantle to serve as the foundation for Vulkan....

[45] "Valve to present glNext: High Performance Graphics at GDC". Hexus. 2014-02-05. Retrieved 2014-02-07.

[46] "Valve Announces Source 2, Steam Link, and Steam VR". *Valve Announces Source 2, Steam Link, and Steam VR – GameSpot*. GameSpot. March 3, 2015.

10 External links

- Official website
- OpenGL.org's Wiki with more information on OpenGL Language bindings
- SGI's OpenGL website
- OpenGL at DMOZ
- Khronos Group, Inc.
- Edward Angel and Dave Shreiner (2013) SIGGRAPH University: "An Introduction to OpenGL Programming"—free course
- Tutorial for modern OpenGL (3.3+)
- OpenGL 3.x/4.x examples
- OpenGL (version 3 and higher) examples

11 Text and image sources, contributors, and licenses

11.1 Text

- **OpenGL** *Source:* https://en.wikipedia.org/wiki/OpenGL?oldid=680731447 *Contributors:* Peter Winnberg, The Anome, Youssefsan, Roadrunner, Hephaestos, Cwitty, Mrwojo, Spiff~enwiki, Frecklefoot, Edward, Lezek, DopefishJustin, Blueshade, Nixdorf, Pnm, Komap, Miciah, Flamurai, Loisel, Minesweeper, CesarB, Haakon, Elano, Miri~enwiki, Glenn, Nikai, Cadr, Jordi Burguet Castell, BAxelrod, Tarka, Victor Engel, Dmsar, Dysprosia, Jharrell, Marco Neves, Wik, Haukurth, Tpbradbury, Furrykef, Stormie, Robbot, Chealer, Razi~enwiki, Dittaeva, AndreasB, Elie De Brauwer, Pengo, Dusik, David Gerard, Enochlau, Michael2, Connelly, Mintleaf~enwiki, Haeleth, Waxmop, Dawidl, AlistairMcMillan, Elmindreda, Bobblewik, Edcolins, Dnas, Yggdræsil, Sam Hocevar, Funvill, DmitryKo, Mormegil, Imroy, Neckelmann, Jcm, EugeneZelenko, RossPatterson, Rich Farmbrough, Alexkon, Sesse, ArnoldReinhold, Dfan, Robertbowerman, Khalid, Andrejj, Evice, Bcat, RJHall, CanisRufus, Jonon, Southen, DarkArctic, Thunderbrand, BrokenSegue, StoatBringer, Cwolfsheep, Csl77, Giraffedata, Colonel Cow, Daf, BlueNovember, JohnyDog, Eile, DanielLC, Fadookie, Derbeth, Lerdsuwa, MIT Trekkie, Mikenolte, Forderud, Mahanga, Marasmusine, Linas, Macaddct1984, SDC, Marudubshinki, RichardWeiss, Ilya, Qwertyus, KramarDanIkabu, Sjö, Rjwilmsi, Tangotango, MZMcBride, The wub, Amelio Vázquez, Reedbeta, FlaBot, Moskvax, Chadstarr, Gurch, T-tus, Fosnez, Fresheneesz, Sketch-The-Fox, SteveBaker, Ahunt, Samkass, Chobot, Nehalem, Dresdnhope, YurikBot, Wormholio, Todd Vierling, Retodon8, Jengelh, Lord-Bleen, Van der Hoorn, טרול רפאים, LesmanaZimmer, Bovineone, 0waldo, Mipadi, Shreddy, Keka, Irishguy, Off!, Bota47, Marcelo-Silva, MaxDZ8, Sandstein, Malchivus, Tdangkhoa, BSTRhino, NotInventedHere, GraemeL, JLaTondre, Btarunr, Samuel Blanning, Klassobanieras, Matzon, Ankurdave, Yakudza, SmackBot, Adam majewski, Renegadeviking, Praetor alpha, Bomac, Darklock, Kragnerac, Alsandro, Xaosflux, Richmeister, Bluebot, Snori, Gil mo, MalafayaBot, ERobson, Octahedron80, Shalom Yechiel, Frap, Racklever, Rrburke, Mark kilgard, UU, Seasage, DrWorm477, Chargh, Rajrajmarley, Martijn Hoekstra, Warren, Iridescence, Salamurai, MadCow257, SashatoBot, Doug Bell, Guyjohnston, Zearin, Khazar, Jinnai, Korval, Indutiomarus, Soumyasch, Physis, BlindWanderer, Mets501, Saxbryn, Jpelcis, JoeBot, AuDioFreaK39, Aeons, Courcelles, Alanedwardes, Tga.D, Spdegabrielle, Chrumps, The ed17, TheoA, Mika1h, Jesse Viviano, Pgr94, Andkore, Pph~enwiki, Skrapion, Skybon, Cydebot, Gremagor, Anthonyhcole, Alanbly, SimenH, SeanMon, Mr. XYZ, Torc2, Chrislk02, Bryan Seecrets, PianoSpleen, Onekopaka, Ebrahim, Amit Moscovich, Thijs!bot, Hervegirod, Jdm64, Mikedep333, NocNokNeo, Escarbot, Gioto, Widefox, Qaridarium, Robzz, KKong, Ellenaz, Carewolf, BahamutZERO, Canadian-Bacon, LiraNuna, Markthemac, IanOsgood, Jephir, .anacondabot, Magioladitis, JamesBWatson, TARBOT, Tedickey, Jancikotuc, Wonx2150, DerHexer, Nicolasherry, 0612, Gwern, Dragentsheets, Quanticles, Speck-Made, R'n'B, Ccs4ever, Juventas, CfW, AgentJ38, Hootsmon, Guilherme Paula, Aldur42, 525252a, Smitty, Sigmundur, Michael Angelkovich, Rfdparker, Plumenator, Vranak, VolkovBot, Sykopomp, Zorexx, Grafman, GrahamAsher, PGSONIC, Moogwrench, Red Act, Comrade Graham, GOD ACRONYM, Bedwyr, Mgsegal, Dlae, Buffs, Haseo9999, Triesault, EmxBot, Trumpetpunk42, Kuøn, Pramsey1234, Wing gundam, J-p krelli, Jerryobject, Flyer22, Elibarzilay, Lightmouse, OKBot, Svick, Pithree, Surakus, Leushenko, Martarius, Sfan00 IMG, ClueBot, Annoyed about usernames, Robenel, EoGuy, Rilak, Aaa3-other, YassineMrabet~enwiki, Spandrawn, UKoch, Apple1976, TMV943, Qalnor, ThinPetr, Mewtu, SF007, Expertjohn, DumZiBoT, Jpirie23, DzzD, Eik Corell, XLinkBot, John Bahrain, JeGX, Dsimic, Addbot, Ghettoblaster, YarkzyBear, Freakmighty, Magus732, MrOllie, Download, AnnaFrance, LinkFA-Bot, MattiPaavola, LarryJeff, Sergioledesma, Abaraba, Jarble, Rchoetzlein, El3fth3ro, Luckas-bot, Yobot, 4th-otaku, AnomieBOT, DemocraticLuntz, Efa, Jim1138, DiscipleRayne, DavidGL, Plattapuss, LilHelpa, Analphabot, CoolingGibbon, Dobz116, Martin Kraus, GrouchoBot, RibotBOT, Hymek, Toyotabedzrock, Nameless23, FrescoBot, Horserice, Lonaowna, Tajmiester, NuclearWizard, M.J. Moore-McGonigal PhD, P.Eng, Winterst, SiPlus, Patrhoue, Egallois, Jandalhandler, Herakleitoszefesu, Rc4, Thelennonorth, Michael9422, Rbarris, Fossalta, Hellomisterman123, ChronoKinetic, Oliver H, PleaseStand, Arnold.Y, RjwilmsiBot, Offnfopt, Phlegat, LPGhatguy, Bdanchilla, J36miles, EmausBot, Eekerz, Super48paul, Dewritech, Ibbn, Zvonce, Kkm010, GoldRenet, Fintelia, A930913, Isrufelvalis, Leledumbo, Signsamongafter, TyA, Tomy9510, Gmt2001, ClueBot NG, Santiagoapplerocks, Catlemur, Satisf, Strcat, Bandit0s, Widr, Ysu cs prof, Boston-cplusplus-guy, Helpful Pixie Bot, BG19bot, Kendall-K1, Yowanvista, Ianteraf, Happyuk, Chmarkine, Gprathour, Afree10, Jonny2BeGood, Vijesh51, Jimw338, Abledsoe78, ChrisGualtieri, Dexbot, Lone boatman, Lugia2453, Matus chochlik, Antoantonov, Kissofdeath.pwn, JakeWi, Simsciloki, R00stare, Georgij Michaliutin, Comp.arch, NottNott, Oranjelo100, ScotXW, BlitzGreg, RealZeratul, Nguillemot, Amortias, DSCrowned, Gregory.d.weber, JMP EAX, Flare loc, Maths314, Pishcal, Sizeofint, Christiand135, Knife-in-the-drawer, Yes mun, Communal t, SpeedDemon520 and Anonymous: 580

11.2 Images

- **File:Commons-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/4/4a/Commons-logo.svg *License:* ? *Contributors:* ? *Original artist:* ?
- **File:Linux_kernel_and_OpenGL_video_games.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/9/99/Linux_kernel_and_OpenGL_video_games.svg *License:* CC BY-SA 3.0 *Contributors:* This image includes elements that have been taken or adapted from this: Tux-shaded.svg. *Original artist:* ScotXW
- **File:OpenGL_logo_(Nov14).svg** *Source:* https://upload.wikimedia.org/wikipedia/en/f/fb/OpenGL_logo_%28Nov14%29.svg *License:* Fair use *Contributors:* https://www.khronos.org/news/logos *Original artist:* ?
- **File:Pipeline_OpenGL.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/d/da/Pipeline_OpenGL.svg *License:* Public domain *Contributors:* Own work created using File:Pipeline OpenGL (en).png and File:Pipeline OpenGL (es).png as a reference. *Original artist:* Offnfopt
- **File:Wikibooks-logo-en-noslogan.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/d/df/Wikibooks-logo-en-noslogan.svg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* User:Bastique, User:Ramac et al.

11.3 Content license

- Creative Commons Attribution-Share Alike 3.0