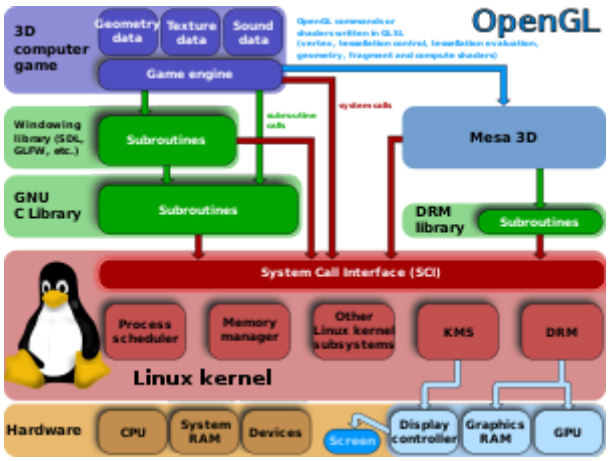
# OpenGL Shading Language

**OpenGL Shading Language** (**GLSL**) is a high-level shading language with a syntax based on the C programming language. It was created by the OpenGL ARB (OpenGL Architecture Review Board) to give developers more direct control of the graphics pipeline without having to use ARB assembly language or hardware-specific languages.




Video games outsource rendering calculations to the GPU over OpenGL in real-time. Shaders are written in **OpenGL Shading Language** and compiled. The compiled programs are executed on the GPU.

## Contents



## Background

With advances in graphics cards, new features have been added to allow for increased flexibility in the rendering pipeline at the vertex and fragment level. Programmability at this level is achieved with the use of fragment and vertex shaders.

Originally, this functionality was achieved by writing shaders in ARB assembly language – a complex and unintuitive task. The OpenGL ARB created the OpenGL Shading Language to provide a more intuitive method for programming the graphics processing unit while maintaining the open standards advantage that has driven OpenGL throughout its history.

Originally introduced as an extension to OpenGL 1.4, GLSL was formally included into the OpenGL 2.0 core in 2004 by the OpenGL ARB. It was the first major revision to OpenGL since the creation of OpenGL 1.0 in 1992.

Some benefits of using GLSL are:

- Cross-platform compatibility on multiple operating systems, including Linux, macOS and Windows.

- The ability to write shaders that can be used on any hardware vendor's graphics card that supports the OpenGL Shading Language.
- Each hardware vendor includes the GLSL compiler in their driver, thus allowing each vendor to create code optimized for their particular graphics card's architecture.

# Versions

GLSL versions have evolved alongside specific versions of the OpenGL API. It is only with OpenGL versions 3.3 and above that the GLSL and OpenGL major and minor version numbers match. These versions for GLSL and OpenGL are related in the following table:

| GLSL Version | OpenGL Version | Date | Shader Preprocessor |
|---|---|---|---|
| 1.10.59[1] | 2.0 | 30 April 2004 | #version 110 |
| 1.20.8[2] | 2.1 | 07 September 2006 | #version 120 |
| 1.30.10[3] | 3.0 | 22 November 2009 | #version 130 |
| 1.40.08[4] | 3.1 | 22 November 2009 | #version 140 |
| 1.50.11[5] | 3.2 | 04 December 2009 | #version 150 |
| 3.30.6[6] | 3.3 | 11 March 2010 | #version 330 |
| 4.00.9[7] | 4.0 | 24 July 2010 | #version 400 |
| 4.10.6[8] | 4.1 | 24 July 2010 | #version 410 |
| 4.20.11[9] | 4.2 | 12 December 2011 | #version 420 |
| 4.30.8[10] | 4.3 | 7 February 2013 | #version 430 |
| 4.40.9[11] | 4.4 | 16 June 2014 | #version 440 |
| 4.50.7[12] | 4.5 | 09 May 2017 | #version 450 |
| 4.60.5[13] | 4.6 | 14 June 2018 | #version 460 |

OpenGL ES and WebGL use **OpenGL ES Shading Language** (abbreviated: **GLSL ES** or **ESSL**).

| GLSL ES version | OpenGL ES version | WebGL version | Based on GLSL version | Date | Shader Preprocessor |
|---|---|---|---|---|---|
| 1.00.17[14] | 2.0 | 1.0 | 1.20 | 12 May 2009 | #version 100 |
| 3.00.6[15] | 3.0 | 2.0 | 3.30 | 29 January 2016 | #version 300 es |

The two languages are related but not directly compatible. They can be interconverted through SPIR-Cross.[16]

# Language

## Operators

GLSL contains the same operators as the operators in C and C++, with the exception of pointers. Bitwise operators were added in version 1.30.

### Functions and control structures

Similar to the C programming language, GLSL supports loops and branching, for instance: if-else, for, switch, etc. Recursion is forbidden and checked for during compilation.

User-defined functions are supported and built-in functions are provided. The graphics card manufacturer may optimize built-in functions at the hardware level. Many of these functions are similar to those in the math library of the C programming language while others are specific to graphics programming. Most of the built-in functions and operators, can operate both on scalars and vectors (up to 4 elements), for one or both operands. Common built-in functions that are provided and are commonly used for graphics purposes are: `mix`, `smoothstep`, `normalize`, `inversesqrt`, `clamp`, `length`, `distance`, `dot`, `cross`, `reflect`, `refract` and vector `min` and `max`. Other functions like `abs`, `sin`, `pow`, etc, are provided but they can also all operate on vector quantities, i.e. `pow(vec3(1.5, 2.0, 2.5), abs(vec3(0.1, -0.2, 0.3)))`. GLSL supports function overloading (for both built-in functions and operators, and user-defined functions), so there might be multiple function definitions with the same name, having different number of parameters or parameter types. Each of them can have own independent return type.

### Preprocessor

GLSL defines a subset of the C preprocessor (CPP), combined with its own special directives for specifying versions and OpenGL extensions. The parts removed from CPP are those relating to file names such as `#include` and `__FILE__`.[17]

The `GL_ARB_shading_language_include` extension[18] (implemented for example in Nvidia drivers[19] on Windows and Linux, and all Mesa 20.0.0[20] drivers on Linux, FreeBSD and Android) implements ability to use `#include` in source code, allowing easier sharing of code and definitions between many shaders without extra manual pre-processing. Similar extension `GL_GOOGLE_include_directive` and `GL_GOOGLE_cpp_style_line_directive` exist for using GLSL with Vulkan, and are supported in reference SPIR-V compiler (`glslang` aka glslangValidator).[21][22][23]

## Compilation and execution

GLSL shaders are not stand-alone applications; they require an application that utilizes the OpenGL API, which is available on many different platforms (e.g., Linux, macOS, Windows). There are language bindings for C, C++, C#, JavaScript, Delphi, Java, and many more.

GLSL shaders themselves are simply a set of strings that are passed to the hardware vendor's driver for compilation from within an application using the OpenGL API's entry points. Shaders can be created on the fly from within an application, or read-in as text files, but must be sent to the driver in the form of a string.

The set of APIs used to compile, link, and pass parameters to GLSL programs are specified in three OpenGL extensions, and became part of core OpenGL as of OpenGL Version 2.0. The API was expanded with geometry shaders in OpenGL 3.2, tessellation shaders in OpenGL 4.0 and compute shaders in OpenGL 4.3. These OpenGL APIs are found in the extensions:

- ARB vertex shader
- ARB fragment shader
- ARB shader objects
- ARB geometry shader 4
- ARB tessellation shader
- ARB compute shader

GLSL shaders can also be used with Vulkan, and are a common way of using shaders in Vulkan. GLSL shaders are precompiled before use, or at runtime, into a binary bytecode format called SPIR-V, usually using offline compiler.

# See also

- Standard Portable Intermediate Representation, an intermediate shader language by Khronos Group
- 3D computer graphics
- Khronos Group
- WebGL, an OpenGL-ES dialect for web browsers, which uses GLSL for shaders
- Shadertoy
- LWJGL, a library that includes Java bindings for OpenGL.

## Other shading languages

- ARB assembly language, a low-level shading language
- Cg, a high-level shading language for programming vertex and pixel shaders
- HLSL, a high-level shading language for use with Direct3D
- TGSI, a low-level intermediate language introduced by Gallium3D
- AMDIL, a low-level intermediate language used internally at AMD
- RenderMan Shading Language

# References

### Citations


1. "GLSL Language Specification, Version 1.10.59" (https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.1.10.pdf) (PDF).
2. "GLSL Language Specification, Version 1.20.8" (https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.1.20.pdf) (PDF).
3. "GLSL Language Specification, Version 1.30.10" (https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.1.30.pdf) (PDF).
4. "GLSL Language Specification, Version 1.40.08" (https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.1.40.pdf) (PDF).
5. "GLSL Language Specification, Version 1.50.11" (https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.1.50.pdf) (PDF).
6. "GLSL Language Specification, Version 3.30.6" (https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.3.30.pdf) (PDF).
7. "GLSL Language Specification, Version 4.00.9" (https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.4.00.pdf) (PDF).

8. "GLSL Language Specification, Version 4.10.6" (https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.4.10.pdf) (PDF).
9. "GLSL Language Specification, Version 4.20.11" (https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.4.20.pdf) (PDF).
10. "GLSL Language Specification, Version 4.30.8" (https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.4.30.pdf) (PDF).
11. "GLSL Language Specification, Version 4.40.9" (https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.4.40.pdf) (PDF).
12. "GLSL Language Specification, Version 4.50.7" (https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.4.50.pdf) (PDF).
13. "GLSL Language Specification, Version 4.60.5" (https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.4.60.pdf) (PDF).
14. "GLSL ES Language Specification, Version 1.00, revision 17" (https://www.khronos.org/registry/OpenGL/specs/es/2.0/GLSL_ES_Specification_1.00.pdf) (PDF).
15. "GLSL ES Language Specification, Version 3.00, revision 6" (https://www.khronos.org/registry/OpenGL/specs/es/3.0/GLSL_ES_Specification_3.00.pdf) (PDF).
16. *KhronosGroup/SPIRV-Cross* (https://github.com/KhronosGroup/SPIRV-Cross), The Khronos Group, 2019-09-06, retrieved 2019-09-08
17. "Shader Preprocessor". *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3, Eighth Edition* (https://www.oreilly.com/library/view/opengl-programming-guide/9780132748445/ch02lev2sec5.html).
18. "ARB_shading_language_include" (https://www.khronos.org/registry/OpenGL/extensions/ARB/ARB_shading_language_include.txt). *Khronos.org*. Retrieved 2020-05-31.
19. "NVIDIA driver 265.90 WHQL Quadro" (https://forums.laptopvideo2go.com/topic/27639-nvidia-driver-26590-whql-quadro/?tab=comments#comment-134520). *LaptopVideo2Go Forums*.
20. "Mesa 20.0.0 Release Notes / 2020-02-19" (https://www.mesa3d.org/relnotes/20.0.0.html). *www.mesa3d.org*. Retrieved 2020-05-31.
21. "#include directive support by antiagainst · Pull Request #46 · KhronosGroup/glslang" (https://github.com/KhronosGroup/glslang/pull/46). *GitHub*. Retrieved 2020-05-31.
22. "Preprocessing line number handling by antiagainst · Pull Request #38 · KhronosGroup/glslang" (https://github.com/KhronosGroup/glslang/pull/38). *GitHub*.
23. "Extend the syntax of #line and __FILE__ to support filename strings by antiagainst · Pull Request #43 · KhronosGroup/glslang" (https://github.com/KhronosGroup/glslang/pull/43). *GitHub*.

# Further reading

### Books

- Rost, Randi J. (30 July 2009). *OpenGL Shading Language* (3rd ed.). Addison-Wesley. ISBN 978-0-321-63763-5.
- Kessenich, John; Baldwin, David; Rost, Randi. *The OpenGL Shading Language*. Version 1.10.59. 3Dlabs, Inc. Ltd.
- Bailey, Mike; Cunningham, Steve (22 April 2009). *Graphics Shaders: Theory and Practice* (2nd ed.). CRC Press. ISBN 978-1-56881-434-6.

# External links

- The official OpenGL website (http://www.opengl.org/)