



OpenGL 3.2 Overview

SIGGRAPH 2009

OpenGL 3.2 Release

- **OpenGL 3.2 Core profile**
 - adds new set of Core features
 - Supportable by all OpenGL 3.x platforms
- **OpenGL 3.2 Compatibility profile**
 - integrates new Core features with legacy functionality
- **GLSL 1.50**
 - support for OpenGL 3.2 features
- **New ARB extensions for OpenGL 3.2**
 - forward looking functionality
 - may require new hardware to support
- **New ARB extensions for OpenGL 2.x**
 - Brings some features of OpenGL 3.x back to 2.x implementations

OpenGL 3.2 Goals

- Target market-specific needs through introduction of profiles
- Increase efficiency for single and multi-threaded applications
- Increase ease of application portability to OpenGL
- **Additional pipeline programmability**
 - new programmable stage - geometry shaders
- **Improve rendering quality**
 - Better multi-sampling control
 - Increased vertex array rendering performance
 - More flexible shadow volume rendering

OpenGL 3.2 - New Features

Feature	Based on Extension
Geometry shaders	GL_ARB_geometry_shader4
Fence sync objects	GL_ARB_sync
OpenGL Shading Language 1.50	n/a
Multisampled textures and sample location control	GL_ARB_texture_multisample
Provoking vertex control	GL_ARB_provoking_vertex
Seamless cube map filtering	GL_ARB_seamless_cube_map
Drawing commands with base vertex index control	GL_ARB_draw_elements_base_vertex
Fragment depth clamping	GL_ARB_depth_clamp
Fragment coordinate convention control	GL_ARB_fragment_coord_conventions
BGRA vertex component ordering	GL_ARB_vertex_array_bgra
Introduction of Core and Compatibility profiles	GL_ARB_compatibility

OpenGL 3.2 – Geometry Shaders

- New programmable stage between vertex and fragment processing
- Accepts incoming primitives from vertex shader
- Can generate new vertices and primitives for rasterization
- Can also output to memory via transform feedback
- Useful for efficient and flexible point sprites
- Integrated from ARB_geometry_shader4 extension

OpenGL 3.2 – Sync Objects

- **New synchronization primitives**
 - adds new sync objects and command stream fences
 - allows one set of commands to wait on another set of commands
- **Support synchronization across OpenGL contexts**
 - sync objects are shared
- **Can synchronize commands on client or server**
- **Foundation for synchronizing OpenGL and OpenCL commands**
- **Also available as ARB_sync extension**

OpenGL 3.2 – Multisample textures

- Defines new texture formats that contain multi-sample data
- Enables fetching specific samples from a texture in a shader
- Allows rendering into multisample textures via FBOs
- Also available as `ARB_texture_multisample` extension
- Additional multisampling support available in optional `ARB_sample_shading` extension

OpenGL 3.2 – Profiles

- **OpenGL 3.0**
 - deprecated (but did not remove) legacy features from core OpenGL
 - fixed-function, immediate mode, etc.
- **OpenGL 3.1**
 - moved deprecated features from core OpenGL to `ARB_compatibility` extension
- **OpenGL 3.2 - formally defines two profiles:**
 - **Core Profile**
 - new features for OpenGL 3.2
 - includes only streamlined, modern functionality
 - **Compatibility Profile**
 - includes all features of “Core”
 - integrates legacy features from `ARB_compatibility` extension
 - defines interaction between legacy features and new 3.2 features

New ARB extensions

- **These five extensions may require new HW support**
 - Not available in all OpenGL 3.X capable currently shipping hardware
- **Improved blending control**
 - Set blending parameters separately for each fragment shader output
 - `GL_ARB_draw_buffers_blend`
- **Added support for storing cubemaps in a texture array**
 - `GL_ARB_texture_cube_map_array`
- **Increased multi-sample rendering quality**
 - Enabling a fragment shader to run per sample, not per pixel, with unique inputs
 - `ARB_sample_shading`
- **Increased texturing quality in a shader**
 - Write custom texture filter kernels for improved rendering quality using `ARB_texture_gather`
 - Across mip-map levels by using the LOD value returned in `ARB_texture_query_lod`

Extensions for OpenGL 3.2

Feature	Extension for OpenGL 3.2
Individual blend equations and functions	ARB_draw_buffers_blend
Cube map array textures	ARB_texture_cube_map_array
Run a fragment shader per sample instead of per pixel for increased rendering quality	ARB_sample_shading
GLSL texture function to return the LOD value	ARB_texture_query_lod
GLSL texture function to return 2x2 footprint used for linear filtering	ARB_texture_gather
Platform extension for creating OpenGL 3 contexts	{WGL GLX}_ARB_create_context

Note: *may require additional hardware support above OpenGL 3.0*

Extensions for OpenGL 2.x

Feature from OpenGL 3.2	Extension for OpenGL 2.x
Add a base value to all indices used by Drawelements	ARB_draw_elements_base_vertex
Use OpenGL or D3D XY location convention in fragment shaders	ARB_fragment_coord_conventions
Use OpenGL or D3D rules for determining the provoking vertex	ARB_provoking_vertex
Support for BGRA vertex format	ARB_vertex_array_bgra
Improved quality when filtering across cube-map faces	ARB_seamless_cube_map
Support for “multi-sample textures”	ARB_texture_multisample
Synchronization primitives, includes cross-context support	ARB_sync