Debug Output

- Obsolete
  - supported, do not use (fallback option)

```
#define GLcall(x) x; …                          // macros: avoid in general, difficult to debug, error prone

GLenum last_error =  glGetError();
if (last_error != GL_NO_ERROR ) {               // only enum, no details …

Glcall(glEnable(GL_CULL_FACE));                 // must manually wrap EVERY call: slow, error prone, unreadable, ...
GLcall(glCullFace(GL_FRONT));
GLcall(glDrawElements(GL_TRIANGLES, count, GL_UNSIGNED_INT, nullptr));
```

- Modern, using asynchronous callback
  - callback function declaration

```
void callback_func(GLenum src, GLenum type, GLuint id, GLenum severity, GLsizei length,
                        GLchar const* msg, void const* user_param);
```

since OpenGL 4.3+
Core in version 4.6
Core since version 4.3
Core ARB extension:
GL_KHR_debug
ARB extension:
GL_ARB_debug_output
Vendor extension
GL_AMD_debug_output

  - at the end of init() function, after all GLFW & GLEW init

```
if (GLEW_ARB_debug_output)
{
      glDebugMessageCallback(MessageCallback, 0);
      glEnable(GL_DEBUG_OUTPUT);
      std::cout << "GL_DEBUG enabled." << std::endl;
}
```

GL Error Callback Example (decoding to string)

```
void GLAPIENTRY MessageCallback(GLenum source, GLenum type, GLuint id, GLenum severity, GLsizei length, const GLchar* message, const void*
userParam)
{
        auto const src_str = [source]() {
                switch (source)
                {
                case GL_DEBUG_SOURCE_API: return "API";
                case GL_DEBUG_SOURCE_WINDOW_SYSTEM: return "WINDOW SYSTEM";
                case GL_DEBUG_SOURCE_SHADER_COMPILER: return "SHADER COMPILER";
                case GL_DEBUG_SOURCE_THIRD_PARTY: return "THIRD PARTY";
                case GL_DEBUG_SOURCE_APPLICATION: return "APPLICATION";
                case GL_DEBUG_SOURCE_OTHER: return "OTHER";
                default: return "Unknown";
                }
        }();

        auto const type_str = [type]() {
                switch (type)
                {
                case GL_DEBUG_TYPE_ERROR: return "ERROR";
                case GL_DEBUG_TYPE_DEPRECATED_BEHAVIOR: return "DEPRECATED_BEHAVIOR";
                case GL_DEBUG_TYPE_UNDEFINED_BEHAVIOR: return "UNDEFINED_BEHAVIOR";
                case GL_DEBUG_TYPE_PORTABILITY: return "PORTABILITY";
                case GL_DEBUG_TYPE_PERFORMANCE: return "PERFORMANCE";
                case GL_DEBUG_TYPE_MARKER: return "MARKER";
                case GL_DEBUG_TYPE_OTHER: return "OTHER";
                default: return "Unknown";
                }
        }();

        auto const severity_str = [severity]() {
                switch (severity) {
                case GL_DEBUG_SEVERITY_NOTIFICATION: return "NOTIFICATION";
                case GL_DEBUG_SEVERITY_LOW: return "LOW";
                case GL_DEBUG_SEVERITY_MEDIUM: return "MEDIUM";
                case GL_DEBUG_SEVERITY_HIGH: return "HIGH";
                default: return "Unknown";
                }
        }();

        std::cout << "[GL CALLBACK]: " <<
                "source = " << src_str <<
                ", type = " << type_str <<
                ", severity = " << severity_str <<
                ", ID = '" << id << '\'' <<
                ", message = '" << message << '\'' << std::endl;
}
```

Detailed Debug Output

- synchronous output – slower, but immediate
  - for glGetError() replacement

```
glEnable(GL_DEBUG_OUTPUT);
glEnable(GL_DEBUG_OUTPUT_SYNCHRONOUS);
glDebugMessageCallback(MessageCallback, nullptr);
```

- usually too noisy, use filter control

```
void glDebugMessageControl(GLenum source, GLenum type, GLenum severity, GLsizei count,
                           const GLuint* ids, GLboolean enabled)
```

  - e.g. disable notification

```
glDebugMessageControl(GL_DONT_CARE, GL_DONT_CARE, GL_DEBUG_SEVERITY_NOTIFICATION, 0, nullptr, GL_FALSE);
```