OpenGL EXT
(OpenGL EXTensions)

OpenGL extensions

- Naming convention: prefix1_prefix2_functionname
  - Prefix1
    - GL_ : all platforms; GLX_ : Linux & Mac (X11); WGL_ : Windows
- New functionality added to OpenGL
  - At first vendor specific (more EXTs for same func possible)

    prefix2: HP_, NV_, ATI_, SGI_, INTEL_ …
  - Generic extension (vendors agree on common implementation)

    prefix2: EXT_
  - EXT promoted to ARB
    - If there is a lot of demand of function, after approval from OpenGL Architecture Review Board

      prefix2: ARB_

- Test for presence of "GL_EXT_bgra" extension
  - `if (GLEW_EXT_bgra) { … }`
  - `bool glewIsSupported("GL_EXT_bgra")`
  - `int glfwExtensionSupported("GL_EXT_bgra")`

Enabling extensions

- In Windows you can directly access only OpenGL up to 1.2 incl.
- Newer functions are present in drivers, but you need to enable it (register entry point)
- Manually: complicated, error prone… (thousands of definitions)
- Example:

```
hasPointParams = isExtensionSupported("GL_EXT_point_parameters");
if (hasPointParams) {
    glPointParameterfEXT = (PFNGLPOINTPARAMETERFEXTPROC);
    wglGetProcAddress("glPointParameterfEXT");
}
```

- Later you can use standard function name

```
if (hasPointParams) {
    static GLfloat quadratic[3] = { 0.25, 0.0, 1/60.0 };
    glPointParameterfvEXT(GL_DISTANCE_ATTENUATION_EXT, quadratic);
}
```

GLEW

- OpenGL Extension Wrangler
  - Simple library for extensions enabling
  - Register all available extensions, constants, …

- Usage (w/o error check):

```
#include „glew.h“
#include „wglew.h“
main ()
{
    … create GL context, e.g. by glfwCreateWindow() …

    glewInit(); //now we can register all usable functions
    wglewInit();
}
```

http://glew.sourceforge.net

Usage (with error check)

⚠ Do not forget: set Visual Studio project directories!

```cpp
// OpenGL Extension Wrangler
// Platform independent part
#include <GL/glew.h>
// Platform specific part. WGLEW = Windows GL Extension Wrangler.
// Change to glxew.h & glxewInit(), or eglew.h & eglewInit(...) if necessary.)
#include <GL/wglew.h>

void init_glew(void) {
    //
    // Initialize all valid generic GL extensions with GLEW.
    // Usable AFTER creating GL context! (create with glfwInit(), glfwCreateWindow(), glfwMakeContextCurrent()
    //
    {
        GLenum glew_ret;
        glew_ret = glewInit();
        if (glew_ret != GLEW_OK) {
            throw std::runtime_error(std::string("GLEW failed with error: ")
                                        + reinterpret_cast<const char *>(glewGetErrorString(glew_ret)) );
        }
        else {
            std::cout << "GLEW successfully initialized to version: " << glewGetString(GLEW_VERSION) << std::endl;
        }

        // Platform specific init.
        glew_ret = wglewInit();
        if (glew_ret != GLEW_OK) {
            throw std::runtime_error(std::string("WGLEW failed with error: ")
                                        + reinterpret_cast<const char *>(glewGetErrorString(glew_ret)) );
        }
        else {
            std::cout << "WGLEW successfully initialized platform specific functions." << std::endl;
        }
    }
    { // if you want, you can get extension list...
        GLint n = 0;
        glGetIntegerv(GL_NUM_EXTENSIONS, &n);
        for (GLint i = 0; i < n; i++) {
            const char* extension_name = reinterpret_cast<const char *>(glGetStringi(GL_EXTENSIONS, i));
            std::cout << extension_name << '\n';
        }
    }
}
```