Introduction to OpenGL Extensions

OpenGL extensions introduce new features and enhance performance. Some extensions provide completely new functionality; for example, the convolution extension allows you to blur or sharpen images using a filter kernel. Other extensions enhance existing functionality; for example, the subtexture extension lets you replace a portion of a texture without reloading the entire image.

Several extensions provide IRIS GL functionality that is not available in OpenGL 1.0. If you are porting a program from IRIS GL to OpenGL, you may therefore find some extensions particularly helpful. See Appendix A, "OpenGL and IRIS GL," for a list of IRIS GL commands and corresponding OpenGL functionality.

This chapter provides basic information about OpenGL extensions. You learn about:

- "Determining Extension Availability" on page 81
- "Finding Information About Extensions" on page 84

Determining Extension Availability

Function names and tokens for OpenGL extensions have EXT or a vendor-specific acronym as a suffix, for example **glConvolutionFilter2DEXT()** or **glColorTableSGI()**. The names of the extensions themselves (the extension strings) use prefixes, for example, SGI_color_table. Here's a detailed list of all suffixes and prefixes:

- EXT is used for extensions that have been reviewed and approved by more than one OpenGL vendor. (Typically these are supported by at least two OpenGL vendors.)
- SGI is used for extensions that may be found across the Silicon Graphics product line, although the support for all products may not appear in the same release.
- SGIS is used for extensions that can be found only on a subset of Silicon Graphics platforms.
- SGIX is used for extensions that are experimental: in future releases, the API for these extensions may change, or they may not be supported at all.

How to Check for OpenGL Extension Availability

All supported extensions have a corresponding definition in *gl.h* and a token in the extensions string returned by **glGetString()**. For example, if the ABGR extension (EXT_abgr) is supported, it is defined in *gl.h* as follows:

```
#define GL_EXT_abgr 1
```

GL_EXT_abgr appears in the extensions string returned by **glGetString()**. Use the definitions in *gl.h* at compile time to determine if procedure calls corresponding to an extension exist in the library.

Applications should do compile-time checking—for example, making sure GL_EXT_abgr is defined; and run-time checking—for example, making sure GL_EXT_abgr is in the extension string returned by **glGetString()**.

- Compile-time checking ensures that entry points such as new functions or new enums are supported. You can't compile or link a program that uses a certain extension if the client side development environment does not support it.
- Run-time checking ensures that the extension is supported for the OpenGL server and run-time library you are using.

  Note that availability depends not only on the operating system but also on the particular hardware you are using: even though the 5.3 OpenGL library supports GL_CONVOLUTION_2D_EXT, you get an GL_INVALID_OPERATION error if you call **glConvolutionFilter2DEXT()** on an Indy system.

Example Program: Checking for Extension Availability

In Example 5-1, the function **QueryExtension()** checks whether an extension is available.

Example 5-1 Checking for Extensions

```
main(int argc, char* argv[]) {
...
    if (!QueryExtension("GL_EXT_texture_object")) {
        fprintf(stderr, "texture_object extension not supported.\n");
        exit(1);
    }
...
}
```

```
static GLboolean QueryExtension(char *extName)
{
    /*
    ** Search for extName in the extensions string. Use of strstr()
    ** is not sufficient because extension names can be prefixes of
    ** other extension names. Could use strtok() but the constant
    ** string returned by glGetString might be in read-only memory.
    */
    char *p;
    char *end;
    int extNameLen;

    extNameLen = strlen(extName);

    p = (char *)glGetString(GL_EXTENSIONS);
    if (NULL == p) {
        return GL_FALSE;
    }

    end = p + strlen(p);

    while (p < end) {
        int n = strcspn(p, " ");
        if ((extNameLen == n) && (strncmp(extName, p, n) == 0)) {
            return GL_TRUE;
        }
        p += (n + 1);
    }
    return GL_FALSE;
}
```

As an alternative to checking for each extension explicitly, you can make the following calls to determine the system and IRIX release on which your program is running:

```
glGetString(GL_RENDERER)
...
glGetString(GL_VERSION)
```

Given a list of extensions supported on that system for that release, you can determine whether the particular extension you need is available. For this to work on all systems, a table of different systems and the extensions supported has to be available.

When an extension is incomplete, it is not advertised in the extensions string. Some of the RealityEngine extensions supported in IRIX 5.3 (for example, the subtexture, sharpen texture, convolution, and histogram extensions) fall in that category.

Checking for GLX Extension Availability

If you use any of the extensions to GLX, described in Chapter 9, “Extensions to GLX,” you also need to check for GLX extension availability.

Querying for GLX extension support is similar to querying for OpenGL extension support with the following exceptions:

- Compile time defines are found in *glx.h.*
- To get the list of supported GLX extensions, call **glXQueryExtensionsString().**
- GLX versions must be 1.1 or greater (no extensions to GLX 1.0 exist).

Adapt the process described in “How to Check for OpenGL Extension Availability” on page 82 taking these exceptions into account.

Finding Information About Extensions

While this book provides rather detailed coverage of several extensions, you may find it helpful to look at the *glintro* reference page for information about the current state of extensions on your system. See the *glXintro* reference page for more information about GLX extensions on your system.

Note that reference pages for extensions are included where the extension is available. Other reference pages that have new options available because of extensions have been updated accordingly.

All complete example programs (though not the short code fragments) are available in */usr/share/src/OpenGL* if you have the *ogl_dev.sw.samples* subsystem installed.