An OpenGL Extension Manager for VTK

Kenneth Moreland

Last Updated: November 16, 2004.

1. Introduction

Over the past several years, the capabilities of graphics hardware have increased quickly. The OpenGL rendering system, however, has not changed as rapidly. Instead, OpenGL implements a means of providing extensions to the base capability. These extensions can be implemented in the underlying graphics libraries without changing the overall API.

Unfortunately, OpenGL extensions are difficult to use. Using OpenGL extensions means retrieving functions from the underlying library, and the mode to do that changes from system to system. Also, code that uses OpenGL extensions must include headers that define necessary identifiers for constants and prototypes for functions and function pointers. Porting these header files is notoriously difficult. It is therefore worthwhile to consolidate the loading and using of extensions into a single module that can be ported and debug independently of all the units using extensions.

There are actually some existing libraries designed to make porting OpenGL extensions easy. For a while we were using gluX.[1] It was easy to use, but had portability problems and was slow to compile. Eventually we determined that the value added was not worth the extra overhead of maintaining or of linking to a constantly changing external library.

Our current solution is basically our own implementation of an OpenGL extension wrapping library. The solution consists of two parts. The first is a small program, vtkParseOGLExt, that parses header files that define OpenGL extensions. During build of vtkSNL, this program is built and run on versions of glext.h, glxext.h, and wglext.h, which are distributed with vtkSNL. vtkParseOGLExt generates a file called vtkgl.h that contains all the constants, prototypes, and pointers needed to use the extensions defined in the three header files. Unlike the original header files, vtkgl.h uses namespaces to avoid naming conflicts or multiple definitions.

The second part of our solution is a VTK object called vtkOpenGLExtensionManager that manages the querying and loading of OpenGL extensions. vtkOpenGLExtension-Manager works in conjunction with the declarations in vtkgl.h.

2. Using OpenGL Extensions

Generally speaking, when using OpenGL extensions, you will need a vtkOpenGL-ExtensionManager and the prototypes defined in vtkgl.h.

---

[1] http://w3imagis.imag.fr/Membres/Sylvain.Lefebvre/glux/

```
#include "vtkOpenGLExtensionManager.h"
#include "vtkgl.h"
```

The vtkgl.h include file contains all the constants and function pointers required for using OpenGL extensions in a portable and namespace safe way.

Before using vtkOpenGLExtensionManager, an OpenGL context must be created. This is generally done with a vtkRenderWindow. This window is given to the vtkOpenGL-ExtensionManager.

```
vtkOpenGLExtensionManager *extensions = vtkOpenGLExtensionManager::New();
extensions->SetRenderWindow(renwin);
```

If it is not given a render window, the vtkOpenGLExtensionManger attempts to load OpenGL extensions from the currently active context. Failing that, it will create its own render window. Note that simply creating the vtkRenderWindow is not sufficient. Usually you have to call Render before the actual OpenGL context is created. vtkOpen-GLExtensionManager will call Render on the render window if necessary.

Once you have established a vtkOpenGLExtensionManager, you can query it to see if an extension is supported with the ExtensionSupported method. Valid names for extensions are given in the OpenGL extension registry.[2] You can also grep vtkgl.h for appropriate names. There are also special extensions GL_VERSION_X_X (where X_X is replaced with a major and minor version), which encapsulate any added functionality for OpenGL releases past 1.1. This is necessary because it is possible (even likely) that the underlying driver will support a newer version of OpenGL than is represented in the gl.h file.

```
if (   !extensions->ExtensionSupported("GL_VERSION_1_2")
    || !extensions->ExtensionSupported("GL_ARB_multitexture") ) {
  {
  vtkErrorMacro("Required extensions not supported!");
  }
```

Once you have verified that the extensions you want exist, you have to load them with LoadExtension before you can use them.

```
extensions->LoadExtension("GL_VERSION_1_2");
extensions->LoadExtension("GL_ARB_multitexture");
```

Finally, the extensions are ready to use. You no longer need the extension manager and can therefore delete it. To use a constant declared in an extension, simply replace the "GL_" prefix with "vtkgl::".

---

[2] http://oss.sgi.com/projects/ogl-sample/registry/

```
extensions->Delete();
…
vtkgl::ActiveTexture(vtkgl::TEXTURE0_ARB);
```

For wgl extensions, replace the “WGL_” and “wgl” prefixes with “vtkwgl::”. For glX extensions, replace the “GLX_” and “glx” prefixes with “vtkglX::”.

3. Acknowledgements

This work was done at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000.