Appendix D

The OpenGL Extension to the X Window System

This appendix briefly discusses the routines defined as part of the OpenGL Extension to the X Window System (GLX). These routines are discussed in more detail in the *OpenGL Reference Manual*. You need to have some knowledge of X to fully understand the following and to use GLX successfully. This appendix has the following major sections:

- **"Initialization"**
- **"Controlling Rendering"**
- **"GLX Prototypes"**

In the X Window System, OpenGL rendering is made available as an extension to X in the formal X sense: Connection and authentication are accomplished with the normal X mechanisms. As with other X extensions, there is a defined network protocol for OpenGL's rendering commands encapsulated within the X byte stream. Since performance is critical in three−dimensional rendering, the OpenGL extension to X allows OpenGL to bypass the X server's involvement in data encoding, copying, and interpretation and instead render directly to the graphics pipeline.

Initialization

Use *glXQueryExtension()* and *glXQueryVersion()* to determine whether the GLX extension is defined for an X server, and if so, which version is present. The *glXChooseVisual()* routine returns a pointer to an XVisualInfo structure describing the visual that best meets the client's specified attributes. You can query a visual about its support of a particular OpenGL attribute with *glXGetConfig()*.

Controlling Rendering

Several GLX routines are provided for creating and managing an OpenGL rendering context. You can use such a context to render off−screen if you want. Routines are also provided for such tasks as synchronizing execution between the X and OpenGL streams, swapping front and back buffers, and using an X font.

Managing an OpenGL Rendering Context

An OpenGL rendering context is created with *glXCreateContext()*. One of the arguments to this routine allows you to request a direct rendering context that bypasses the X server as described previously. (Note that to do direct rendering, the X server connection must be local, and the OpenGL implementation needs to support direct rendering.) You can determine whether a GLX context is direct with *glXIsDirect()*.

To make a rendering context current, use *glXMakeCurrent()*; *glXGetCurrentContext()* returns the current context. You can also obtain the current drawable with *glXGetCurrentDrawable()*. Remember that only one context can be current for any thread at any one time. If you have multiple contexts, you can copy selected groups of OpenGL state variables from one context to another with *glXCopyContext()*. When you're finished with a particular context, destroy it with *glXDestroyContext()*.

Off−Screen Rendering

To render off−screen, first create an X Pixmap and then pass this as an argument to *glXCreateGLXPixmap()*. Once rendering is completed, you can destroy the association between the X and GLX Pixmaps with *glXDestroyGLXPixmap()*. (Off−screen rendering isn't guaranteed to be supported for direct renderers.)

Synchronizing Execution

To prevent X requests from executing until any outstanding OpenGL rendering is completed, call *glXWaitGL()*. Then, any previously issued OpenGL commands are guaranteed to be executed before any X rendering calls made after *glXWaitGL()*. Although the same result can be achieved with *glFinish()*, *glXWaitGL()* doesn't require a round trip to the server and thus is more efficient in cases where the client and server are on separate machines.

To prevent an OpenGL command sequence from executing until any outstanding X requests are completed, use *glXWaitX()*. This routine guarantees that previously issued X rendering calls are executed before any OpenGL calls made after *glXWaitX()*.

Swapping Buffers

For drawables that are double−buffered, the front and back buffers can be exchanged by calling *glXSwapBuffers()*. An implicit *glFlush()* is done as part of this routine.

Using an X Font

A shortcut for using X fonts in OpenGL is provided with the command *glXUseXFont()*.

GLX Prototypes

Initialization

Determine whether the GLX extension is defined on the X server:
Bool *glXQueryExtension* (Display * *dpy*, int * *errorBase*, int * *eventBase*);
Bool *glXQueryVersion* (Display * *dpy*, int * *major*, int * *minor*);

Obtain the desired visual:
XVisualInfo* *glXChooseVisual* (Display * *dpy*, int *screen*, int * *attribList*);
int *glXGetConfig* (Display * *dpy*, XVisualInfo * *vis*, int *attrib*, int * *value*);

Controlling Rendering

Manage or query an OpenGL rendering context:
GLXContext *glXCreateContext* (Display **dpy*, XVisualInfo **vis*, GLXContext *shareList*, Bool *direct*);
void *glXDestroyContext* (Display **dpy*, GLXContext *ctx*);
void *glXCopyContext* (Display * *dpy*, GLXContext *src*, GLXContext *dst*, GLuint *mask*);
Bool *glXIsDirect* (Display * *dpy*, GLXContext *ctx*);
Bool *glXMakeCurrent* (Display * *dpy*, GLXDrawable *draw*, GLXContext *ctx*);
GLXContext *glXGetCurrentContext* (void);
GLXDrawable *glXGetCurrentDrawable* (void);

Perform off−screen rendering:
GLXPixmap *glXCreateGLXPixmap* (Display * *dpy*, XVisualInfo * *vis*, Pixmap *pixmap*);
void *glXDestroyGLXPixmap* (Display * *dpy*, GLXPixmap *pix*);

Synchronize execution:
void *glXWaitGL* (void);
void *glXWaitX* (void);

Exchange front and back buffers:
void *glXSwapBuffers* (Display * *dpy*, Window *window*);

Use an X font:
void *glXUseXFont* (Font *font*, int *first* , int *count* , int *listBase*);