OpenGL® Graphics with the X Window System® (Version 1.4)

Editors (versions 1.0-1.2): Phil Karlton, Paula Womack
Editors (version 1.3): Paula Womack, Jon Leech
Editor (version 1.4): Jon Leech

Contents

List of Figures

Chapter 2

GLX Operation

2.1 Rendering Contexts and Drawing Surfaces

GLX. The state of which buffer is displayed tracks in both extensions, independent of which extension initiates a buffer swap.

command streams. GLX relaxes these requirements. Sequentiality is still guaranteed within a command stream, but not between the X and the OpenGL com-

`GLXBadPbuffer` The `GLXPbuffer` argument does not name a `GLXPbuffer`.

`GLXBadPixmap` The `Pixmap` argument does not name a `Pixmap` that is appropriate for OpenGL rendering.

`GLXUnsupportedPrivateRequest` May be returned in response to either a **glXVendorPrivate** request or a **glXVendorPrivateWithReply** request.

`GLXBadWindow` The `GLXWindow` argument does not name a `GLXWindow`.

The following error codes may be generated by a faulty GLX implementation, but would not normally be visible to clients:

`GLXBadContextTag` A rendering request contains an invalid context tag. (Context tags are used to identify contexts in the protocol.)

`GLXBadRenderRequest` A **glXRender** request is ill-formed.

`GLXBadLargeRequest` A **glXRenderLarge** request is ill-formed.

3.2 Events

GLX introduces one new event:

`GLX_PbufferClobber` The given pbuffer has been removed from framebuffer memory and may no longer be valid. These events are generated as a result of conflicts in the framebuffer allocation between two drawreb in a52e011r bein the drawrebgs are

sumes that no other pbuffers or X resources are contending for the framebuffer memory. Thus it may not be possible to allocate a pbuffer of the size given by GLX_MAX_PBUFFER_PIXELS.

Use

```
GLXFBConfig *glXGetFBConfigs(Display *dpy, int
   screen, int *nelements);
```

to get the list of all GLXFBConfigs that are available on the specified screen. The call returns an array of GLXFBConfigs; the number of elements in the array is returned in *nelements*.

Use

```
GLXFBConfig *glXChooseFBConfig(Display *dpy, int
   screen, const int *attrib_list, int
   *nelements);
```

to get GLXFBConfigs that match a list of attributes.

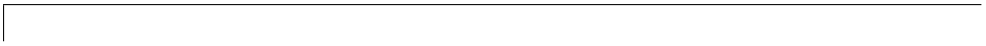
This call returns an array of GLXFBConfigs that match the specified at-

Larger GLXFBConfigs with an attribute value that meets or exceeds the specified value are returned.

Exact Only GLXFBConfigs whose attribute value exactly matches the requested value are considered.

Mask Only GLXFBConfigs for which the set bits of attribute include all the bits that are set in the requested value are considered. (Additional bits might be set in the attribute).

Some of the attributes, such as GLX_LEVEL, must match the specified value ex-



Refer to Table 3.1 and Table 3.4 for a list of valid GLX attributes.
A

This request deletes the association between the resource ID *win* and the GLX window. The storage will be freed when it is not current to any client.

If *win* is not a valid GLX window then a `GLXBadWindow` error is generated.

3.3.5 Off Screen Rendering

GLX supports two types of offscreen rendering surfaces: `GLXPixmaps` and `GLXPbuffers`. `GLXPixmaps` and `GLXPbuffers` differ in the following ways:

1. `GLXPixmaps` have an associated X pixmap and can therefore be rendered to by X. Since a `GLXPbuffer` is a GLX resource, it may not be possible to render to it using X or an X extension other than GLX.

2. The format of the color buffers and the type and size of any associated ancillary buffers for a `GLXPbuffer` can only be described with a `GLXFBConfig`. The older method of using extended X `Visuals` to de-

glXCreatePixmap creates an offscreen rendering area and returns its XID. Any GLX rendering context created with a GLXFBConfig that is compatible with *config* can be used to render into this offscreen area.

pixmap is used for the RGB planes of the front-left buffer of the resulting GLX offscreen rendering area. GLX pixmaps may be created with a *config* that includes back buffers and stereoscopic buffers. However, **glXSwapBuffers** is ignored for these pixmaps.

*attrib_*

attrib

queue for pbuffer clobber events (assuming that these events had been pulled off of

glXCreateNewContext returns NULL if it fails. If **glXCreateNewContext** succeeds, it initializes the rendering context to the initial OpenGL state and returns a handle to it. This handle can be used to render to GLX windows, GLX pixmaps and GLX pbuffers.

If *render_type* is set to GLX_RGBA_TYPE then a context that supports RGBA rendering is created; if *render_type* is set to GLX_COLOR_INDEX_

glXDestroyContext will generate a `GLXBadContext` error if *ctx* is not a valid rendering context.

To make a context current, call

```
Bool glXMakeContextCurrent(Display *dpy,
   GLXDrawable draw, GLXDrawable read,
   GLXContext ctx);
```

glXMakeContextCurrent binds *ctx* to the current rendering thread and to the *draw* and *read* drawables. *draw* is used for all OpenGL operations except:

- Any pixel data that are read based on the value of

3.3.8 Events

GLX events are returned in the X11 event stream. GLX and X11 events are selected independently; if a client selects for both, then both may be delivered to the client. The relative order of X11 and GLX events is not specified.

A client can ask to receive GLX events on a `GLXWindow`

For an unpreserved pbuffer a pbuffer clobber event, with *event_type* GLX_DAMAGED, is generated whenever a portion of the pbuffer becomes invalid.

For GLX windows, pbuffer clobber events with *event_type* GLX_SAVED occur whenever an ancillary buffer, associated with the window, gets moved out of off-screen memory. The event contains information indicating which color or ancillary

glXUseXFont is

Attribute	Type	Notes
GLX_USE_GL		

```
void glXDestroyGLXPixmap(Display *dpy, GLXPixmap
  pixmap);
```

This function is equivalent to **glXDestroyPixmap**; however, GLXPixmaps created by calls other than **glXCreateGLXPixmap** should not be passed to

If

glXCreateGLXPixmap
glXCreateNewContext
glXCreatePbuffer
glXCreatePixmap
glXCreateWindow
glXDestroyContext
glXDestroyGLXPixmap
glXDestroyPbuffer
glXDestroyPixmap
glXDestroyWindow
glXMakeContextCurrent
glXMakeCurrent
glXIsDirect
glXQueryContext
glXQueryDrawable
glXQueryExtension
glXQueryExtensionsString
glXQueryServerString
glXQueryVersion
glXSelectEvent
glXWaitGL
glXSwapBuffers

Commands in both streams, which force a rendezvous, are:

Chapter 5

Extending OpenGL

OpenGL implementors may extend OpenGL by adding new OpenGL commands or additional enumerated values for existing OpenGL commands. When a new vendor-specific command is added, GLX protocol must also be defined. If the new command is one that cannot be added to a display list, then protocol for a new **glXVendorPrivate** or **glXVendorPrivateWithReply** request is required; oth-

Chapter 7

Glossary

Address Space the set of objects or memory locations accessible through a single

similar if, and only if, they have been created with respect to the same `VisualID` and root window.

Thread

Index of GLX Commands