# GLX Extensions For OpenGL® Protocol Specification (Version 1.3)

Document Editor (version 1.3): Jon Leech (previous versions): Deanna Hohn, Paula Womack

### Copyright © 2006-2009 The Khronos Group Inc. All Rights Reserved.

This specification is protected by copyright laws and contains material proprietary to the Khronos Group, Inc. It or any components may not be reproduced, republished, distributed, transmitted, displayed, broadcast or otherwise exploited in any manner without the express prior written permission of Khronos Group. You may use this specification for implementing the functionality therein, without altering or removing any trademark, copyright or other notice from the specification, but the receipt or possession of this specification does not convey any rights to reproduce, disclose, or distribute its contents, or to manufacture, use, or sell anything that it may describe, in whole or in part.

Khronos Group grants express permission to any current Promoter, Contributor or Adopter member of Khronos to copy and redistribute UNMODIFIED versions of this specification in any fashion, provided that NO CHARGE is made for the specification and the latest available update of the specification for any version of the API is used whenever possible. Such distributed specification may be re-formatted AS LONG AS the contents of the specification are not changed in any way. The specification may be incorporated into a product that is sold as long as such product includes significant independent work developed by the seller. A link to the current version of this specification on the Khronos Group web-site should be included whenever possible with specification distributions.

Khronos Group makes no, and expressly disclaims any, representations or warranties, express or implied, regarding this specification, including, without limitation, any implied warranties of merchantability or fitness for a particular purpose or non-infringement of any intellectual property. Khronos Group makes no, and expressly disclaims any, warranties, express or implied, regarding the correctness, accuracy, completeness, timeliness, and reliability of the specification. Under no circumstances will the Khronos Group, or any of its Promoters, Contributors or Members or their respective partners, officers, directors, employees, agents or representatives be liable for any damages, whether direct, indirect, special or consequential damages for lost revenues, lost profits, or otherwise, arising from or in connection with these materials.

Khronos is a trademark of The Khronos Group Inc. OpenGL is a registered trademark, and OpenGL ES is a trademark, of Silicon Graphics International.

# **Contents**

1	Intr	oduction 1
	1.1	Overview
	1.2	Syntax
	1.3	Definitions
		Rendering Contexts
		Visuals
		GLX Drawables
		GLX FBConfig
		GLX Pixmaps
		GLX Pbuffers
		GLX Windows
		Threads
	1.4	Common Types
		ATTRIBUTE_PAIR 2
		BITFIELD
		BOOL32 3
		ENUM
		FBCONFIGID 3
		FLOAT32 3
		FLOAT64 3
		GLX_CONTEXT
		GLX_CONTEXT_TAG
		GLX_DRAWABLE
		GLX_PIXMAP 3
		GLX_PBUFFER
		GLX_RENDER_COMMAND 3
		GLX_WINDOW
		VISUAL_PROPERTY
	1.5	Errors
		GLXBadContext
		GLXBadContextState
		GLXBadDrawable
		GLXBadPixmap
		GLXBadContextTag

CONTENTS ii

		GLXBadCurrentWindow	6
		GLXBadRenderRequest	6
		GLXBadLargeRequest	6
		GLXUnsupportedPrivateRequest	7
		GLXBadFBConfig	7
		GLXBadPbuffer	8
		GLXBadCurrentDrawable	8
		GLXBadWindow	8
	1.6	Events	9
		GLX_PbufferClobber	9
	1.7	Padding and Unused Bytes	10
	1.8	Context Tags	10
		-	
2	Req	uests	11
	2.1	Requests for GLX Commands	11
		Query Extension Version	11
		Query Server String	12
		Send Client OpenGL Information to the Server	13
		Create a Rendering Context	14
		Destroy a Rendering Context	15
		Make a Rendering Context and a Drawable Current	16
		Query Whether a Rendering Context is Direct	17
		Copy State From One Rendering Context to Another	18
		Complete GL Execution Prior to Subsequent X Requests	19
		Complete X Execution Prior to Subsequent GL Requests	21
		Exchange Front and Back Buffers	22
		Create Bitmap Display Lists From an X Font	23
		Create an Offscreen Rendering Area	24
		Destroy an Offscreen Rendering Area	25
		Get List of Visual Configurations	25
		Vendor-specific Private Request	27
		Vendor-specific Private Request with Reply	28
		Get List of Frame Buffer Configurations	29
		Create an Offscreen Rendering Area from Frame Buffer Configuration	30
		Destroy an Offscreen Rendering Area Created with Frame Buffer Con-	
		figuration	31
		Create a Rendering Context from Frame Buffer Configuration	32
		Query Context Attributes	33
		Make a Rendering Context and Read/Draw Drawables Current	34
		Create an Offscreen Pixel Buffer	36
		Destroy an Offscreen Pixel Buffer	37
		Get List of Drawable Attributes	37
		Change Drawable Attributes	38
		Create a Window	39
		Destroy a Window	40
	2.2	Requests for GL Non-rendering Commands	41

CONTENTS

2.2.1 GL Non-rendering Commands	
AreTexturesResident	
DeleteLists	41
DeleteTextures	
EndList	
FeedbackBuffer	42
Finish	
Flush	
GenLists	43
GenTextures	43
GetBooleanv	44
GetClipPlane	44
GetColorTableParameterfv	45
GetColorTableParameteriv	
GetConvolutionParameterfy	
GetConvolutionParameteriv	
GetDoublev	
GetError	48
GetFloatv	
GetHistogramParameterfv	
GetHistogramParameteriv	
GetIntegerv	
GetLightfv	
GetLightiv	
GetMapdv	
GetMapfv	
GetMapiv	
GetMaterialfv	
GetMaterialiv	
GetMinmaxParameterfy	
GetMinmaxParameteriv	
GetPixelMapfv	
GetPixelMapuiv	
GetPixelMapusv	
GetString	
GetTexEnvfv	
GetTexGendv	
GetTexGenfv	61
GetTexGeniv	
GetTexLevelParameteriv	
GetTexParameterfy	
GetTexParameteriv	
IsList	
IsTexture	65

CONTENTS iv

	NewList	66
	PixelStoref	66
	PixelStorei	66
	RenderMode	67
	SelectBuffer	67
	2.2.2 GL Non-rendering Commands That Return Pixel Data	68
	GetColorTable	68
	GetConvolutionFilter	69
	GetHistogram	69
	GetMinmax	70
	GetPolygonStipple	71
	GetSeparableFilter	71
	GetTexImage	72
	ReadPixels	73
2.3	Requests for GL Rendering Commands	73
2.3		73 74
		7 <del>4</del> 75
	8	
	$\varepsilon$	78 78
	Accum	78 70
	ActiveTextureARB	78 70
	AlphaFunc	78 70
	Begin	<b>78</b>
	BindTexture	79
	BlendColor	79
	BlendEquation	79
	BlendFunc	79
	CallList	79
	Clear	79
	ClearAccum	80
	ClearColor	80
	ClearDepth	80
	ClearIndex	80
	ClearStencil	80
	ClipPlane	81
	Color3bv	81
	Color3dv	81
	Color3fv	81
	Color3iv	82
	Color3sv	82
	Color3ubv	82
	Color3uiv	82
	Color3usv	82
	Color4bv	83
	Color4dv	83
	Color4fv	83
	Color4iv	83

CONTENTS v

Color4sv	84
Color4ubv	84
Color4uiv	84
Color4usv	84
ColorMask	85
ColorMaterial	85
ColorTableParameterfv	85
ColorTableParameteriv	85
ConvolutionParameterf	86
ConvolutionParameterfy	86
ConvolutionParameteri	86
ConvolutionParameteriv	86
CopyColorSubTable	87
CopyColorTable	87
CopyConvolutionFilter1D	87
CopyConvolutionFilter2D	88
CopyPixels	88
CopyTexImage2D	88
CopyTexSubImage1D	88
CopyTexSubImage2D	89
CopyTexSubImage3D	89
CullFace	89
DepthFunc	90
DepthMask	90
DepthRange	90
DrawBuffer	90
	90
EdgeFlagv	90
	91 91
EvalCoord1dv	
EvalCoord1fv	91
EvalCoord2dv	91
EvalCoord2fv	91
EvalMesh1	91
EvalMesh2	92
EvalPoint1	92
EvalPoint2	92
Fogf	92
Fogfv	92
Fogi	93
Fogiv	93
FrontFace	93
Frustum	94
Hint	94
Histogram	94
Indexdv	94
Indexfv	94

CONTENTS vi

Indexiv	95
IndexMask	95
Indexsv	95
Indexubv	95
InitNames	95
Lightf	95
Lightfv	96
Lighti	96
Lightiv	96
LightModelf	97
LightModelfv	97
LightModeli	97
LightModeliv	97
LineStipple	98
LineWidth	98
ListBase	98
LoadIdentity	98
LoadMatrixd	98
LoadMatrixf	99
LoadName	99
LogicOp	99
MapGrid1d	99
MapGrid1f	99
MapGrid2d	100
MapGrid2f	100
Materialf	100
Material fv	100
Materiali	101
Materialiv	101
MatrixMode	101
Minmax	102
MultiTexCoord1dvARB	102
MultiTexCoord1fvARB	102
MultiTexCoord1ivARB	102
MultiTexCoord1svARB	102
MultiTexCoord2dvARB	103
MultiTexCoord2fvARB	103
MultiTexCoord2ivARB	103
MultiTexCoord2svARB	103
MultiTexCoord3dvARB	103
MultiTexCoord3fvARB	104
MultiTexCoord3ivARB	104
MultiTexCoord3svARB	104
MultiTexCoord4dvARB	104
MultiTexCoord4fvARB	105
MultiTeyCoord4iyARB	105

CONTENTS vii

/fultiTexCoord4svARB	105
IultMatrixd	105
MultMatrixf	106
Normal3bv	106
Normal3dv	106
Normal3fv	106
Normal3iv	106
Normal3sv	107
Ortho	107
assThrough	107
ixelTransferf	107
ixelTransferi	108
ixelZoom	108
ointSize	108
olygonMode	108
olygonOffset	108
opAttrib	108
opMatrix	109
opName	109
rioritizeTextures	109
PushAttrib	109
rushMatrix	109
rushName	109
RasterPos2dv	110
RasterPos2fv	110
RasterPos2iv	110
RasterPos2sv	110
RasterPos3dv	110
RasterPos3fv	111
RasterPos3iv	111
RasterPos3sv	111
RasterPos4dv	111
RasterPos4fv	112
CasterPos4iv	112
RasterPos4sv	112
LeadBuffer	112
lectdv	112
lectfv	113
lectiv	113
lectsv	113
ResetHistogram	113
ResetMinmax	114
Rotated	114
Rotatef	114
caled	114
calef	114

CONTENTE	•••
CONTENTS	V111

Scissor	115
ShadeModel	115
StencilFunc	115
StencilMask	115
StencilOp	115
TexCoord1dv	116
TexCoord1fv	116
TexCoord1iv	116
TexCoord1sv	116
TexCoord2dv	116
TexCoord2fv	117
TexCoord2iv	117
TexCoord2sv	117
TexCoord3dv	117
TexCoord3fv	117
TexCoord3iv	118
TexCoord3sv	118
TexCoord4dv	118
TexCoord4fv	118
TexCoord4iv	118
TexCoord4sv	119
TexEnvf	119
TexEnvfv	119
	119
TexEnvi	
TexEnviv	120
TexGend	120
TexGendv	120
TexGenf	120
TexGenfv	121
TexGeni	121
TexGeniv	121
TexParameterf	121
TexParameterfy	122
TexParameteri	122
TexParameteriv	122
Translated	123
Translatef	123
Vertex2dv	123
Vertex2fv	123
Vertex2iv	123
Vertex2sv	124
Vertex3dv	124
Vertex3fv	124
Vertex3iv	124
Vertex3sv	124
Vertex4dv	125

CONTENTS ix

	Vertex4fv	125
	Vertex4iv	125
	Vertex4sv	125
	Viewport	126
	xImage1D	126
	2.3.4 GL Rendering Commands That May Be Large	126
	CallLists	126
	DrawArrays	127
	PixelMapfv	129
	PixelMapuiv	130
	PixelMapusv	130
	PrioritizeTextures	131
	2.3.5 GL Rendering Commands with Evaluator Map Data	131
	Map1d	131
	Map1f	131
	÷	132
	Map2d	133
	Map2f	
	2.3.6 GL Rendering Commands with Pixel Data	134
	Bitmap	134
	ColorTable	135
	ColorSubTable	136
	ConvolutionFilter1D	137
	ConvolutionFilter2D	137
	SeparableFilter2D	138
	DrawPixels	139
	PolygonStipple	140
	TexImage1D	140
	TexImage2D	141
	TexImage3D	142
	TexSubImage1D	143
	TexSubImage2D	144
	TexSubImage3D	145
A	Pixel Data	147
	A.1 Pixel Format and Type	147
	A.2 Pixel Data in Rendering Commands	147
	A.2.1 Encoding For Pixel Types Other Than GL_BITMAP	149
	A.2.2 Encoding For Pixel Type GL_BITMAP	151
	A.3 Pixel Data in Replies	152
	A.3.1 Encoding For Pixel Types Other Than GL_BITMAP	153
	A.4 Encoding For Pixel Type GL_BITMAP	153
В	GLX Versions	155
D	B.1 Requests for GLX commands	155
		156
	1 1	
	B.3 Protocol for OpenGL rendering commands	156

CONTENTS	X

C	References	158
---	------------	-----

# **List of Figures**

A.1 Pixel Packing Paramete	rs	150
----------------------------	----	-----

# **List of Tables**

2.1	Type and size of lists	127
2.2	Values Per Control Point for <b>Map1d</b> and <b>Map1f</b>	131
2.3	Values Per Control Point for <b>Map2d</b> and <b>Map2f</b>	132
A.1	Bytes per element.	148
A.2	Elements per group	149
A.3	Pixel Packing Attributes	151
A.4	Encoding For Pixel Types Other Than GL_BITMAP	152

# **Chapter 1**

# Introduction

### 1.1 Overview

GLX is the OpenGL extension to the X Window System. It provides for OpenGL rendering in an X environment, and is an extension to X in the formal sense: connection and authentication are accomplished with the normal X mechanisms. This document describes the network protocol for GLX as it is encapsulated within the X protocol byte stream.

Many details of OpenGL and GLX are described in the OpenGL Specification<sup>1</sup> and the GLX Specification<sup>2</sup>, and those documents will be referred to frequently rather than repeating the details here.

# 1.2 Syntax

When possible, this document uses the layout and syntactic conventions used in the X encoding document. Note that all numbers are decimal, unless prefixed by 0x, in which case they are hexadecimal. Note that for entities of variable size E, the notation pad(E) indicates the number of bytes needed to pad the entity to a multiple of 4 bytes. Also, the C - like syntax (*expr*? a: b) evaluates to a if expr is true, b if it is false.

<sup>&</sup>lt;sup>1</sup> The OpenGL <sup>®</sup> Graphics System: A Specification, Version 1.2.1, Segal, Mark, and Akeley, Kurt.

<sup>&</sup>lt;sup>2</sup> OpenGL<sup>®</sup> Graphics with the X Window System, Version 1.3, Karlton, Phil.

1.3. DEFINITIONS 2

### 1.3 Definitions

**Rendering Contexts** A GLX rendering context is an abstract OpenGL state machine.

- Visuals In GLX, the definition of a Visual has been extended to include attributes describing doublebuffering capability, OpenGL rendering support, and the types, quantities, and sizes of the ancillary buffers (depth, accumulation, auxiliary, and stencil). The ancillary buffers have no meaning in the core X environment. A GLX implementation need not support OpenGL rendering for all Visuals; in this document, a *valid visual* means a visual which has rendering support.
- **GLX Drawables** A GLX drawable is the GLX equivalent of an X drawable; instead of being the union of X windows and X pixmaps, it is the union of X windows, GLX pixmaps, GLX pbuffers, and GLX windows.
- **GLX FBConfig** A GLX FBConfig describes the format, type, and size of the color and ancillary buffers for a GLX Drawable.
- **GLX Pixmaps** A GLX pixmap is the GLX equivalent of an X pixmap; the difference is that a GLX pixmap has the extended visual properties described above.
- **GLX Pbuffers** A GLX pbuffer is a GLX drawable used for offscreen rendering; pbuffers have different semantics than GLX pixmaps that make them easier to allocate in non-visible frame buffer memory
- **GLX Windows** A GLX window is a GLX drawable used for onscreen rendering; it is the GLX equivalent of an X Window.
- **Threads** The GLX protocol allows multiple threads of execution to share an X connection, with each thread possibly having its own current context and drawable. In this document, the *calling thread* of a request is the thread that issued that request.

# 1.4 Common Types

In addition to the common types described in the X core protocol, the GLX protocol adds the following types:

- ATTRIBUTE\_PAIR A 32-bit enumerated value indicating the attribute type followed by a 32-bit attribute value. The data type for the attribute value depends on the attribute type.
- BITFIELD A 32-bit mask. This is mainly used in GL rendering commands; the range of valid masks depends on the particular command in which it is used; refer to the OpenGL Spec for each command. Unless otherwise stated, a BITFIELD that is invalid under the GL API does not generate a protocol error.

- BOOL32 A 32-bit integer Boolean; 1 represents True and 0 represents False.
- ENUM A 32-bit enumerated value. This is mainly used in GL rendering commands; the range of valid enumerants depends on the particular command in which it is used; refer to the OpenGL Spec for each command. Unless otherwise stated, an ENUM that is invalid under the GL API does not generate a protocol error.
- FBCONFIGID A 32-bit identifier that refers to a frame buffer configuration.
- FLOAT32 A 32-bit floating point value in IEEE Single Format.
- FLOAT64 A 64-bit floating point value in IEEE Double Format.
- GLX\_CONTEXT A 32-bit identifier that refers to a GLX rendering context.
- GLX\_CONTEXT\_TAG A 32-bit integer used to identify the current context of a calling thread. See the description of context tags in section 1.8, "Context Tags", for more details.
- GLX\_DRAWABLE The union of { WINDOW, GLX\_PBUFFER, GLX\_PIXMAP, GLX\_WINDOW }.
- GLX\_PIXMAP A 32-bit identifier that refers to a GLX pixmap.
- GLX\_PBUFFER A 32-bit identifier that refers to a GLX pbuffer.
- GLX\_RENDER\_COMMAND An OpenGL rendering command and its associated data. See section 2.3, "Requests for GL Rendering Commands", for more details.
- GLX\_WINDOW A 32-bit identifier that refers to a GLX window. GLX windows are distinct from core X windows.
- VISUAL\_PROPERTY A ordered list of 32-bit property values followed by unordered pairs of property types and property values. The data type for the property values depends on their position in the ordered list or the property type of values in the unordered list.

### 1.5 Errors

BEC is the base error code for the extension, as returned by QueryExtension.

The GLX Protocol uses the same error codes as the X Protocol when appropriate, and adds these new errors:

### GLXBadContext

A value for a GLX rendering context identifier is illegal or does not name a defined context.

### **Encoding:**

1	O	Error
1	BEC + 0	<pre>Error code (GLXBadContext)</pre>
2	CARD16	sequence number
4	CARD32	bad context ID
2	CARD16	minor opcode
1	CARD8	major opcode
21		unused

### GLXBadContextState

The current GLX rendering context of a thread is not in rendering mode (i.e., it is in feedback or selection mode) when the thread issues a **glXMakeContextCurrent** or **glXMakeCurrent** request. Or, the current context of a thread is already in display list construction when the thread issues a **glXUseXFont** request.

### **Encoding:**

1	0	Error
1	BEC + 1	<pre>Error code (GLXBadContextState)</pre>
2	CARD16	sequence number
4	CARD32	context ID
2	CARD16	minor opcode
1	CARD8	major opcode
21		unused

### **GLXBadDrawable**

A value for a GLX drawable parameter is illegal or does not name a defined GLX drawable.

1	O	Error
1	BEC + 2	<pre>Error code (GLXBadDrawable)</pre>
2	CARD16	sequence number
4	CARD32	bad GLX Drawable ID
2	CARD16	minor opcode
1	CARD8	major opcode
21		unused

# GLXBadPixmap

A value for a GLX pixmap parameter is illegal or does not name a defined GLX pixmap.

## **Encoding:**

1	0	Error
1	BEC + 3	<pre>Error code (GLXBadPixmap)</pre>
2	CARD16	sequence number
4	CARD32	bad GLX Pixmap ID
2	CARD16	minor opcode
1	CARD8	major opcode
21		unused

# GLXBadContextTag

A value for a context tag is invalid.

1	0	Error
1	BEC + 4	<pre>Error code (GLXBadContextTag)</pre>
2	CARD16	sequence number
4	CARD32	bad context tag
2	CARD16	minor opcode
1	CARD8	major opcode
21		unused

### GLXBadCurrentWindow

The current drawable of the calling thread is a window that is no longer valid. No similar error is needed for the case when the current drawable is a GLX pixmap because, unlike windows, GLX pixmaps are reference-counted and are not freed until they are no longer referenced.

### **Encoding:**

1	O	Error
1	BEC + 5	<pre>Error code (GLXBadCurrentWindow)</pre>
2	CARD16	sequence number
4	CARD32	ID of invalid current window
2	CARD16	minor opcode
1	CARD8	major opcode
21		unused

## GLXBadRenderRequest

A glXRender request contains an invalid parameter.

### **Encoding:**

1	O	Error
1	BEC + 6	Error code (GLXBadRenderRequest)
2	CARD16	sequence number
4	CARD32	number of rendering commands before error
2	CARD16	minor opcode
1	CARD8	major opcode
21		unused

# GLXB ad Large Request

A series of glXRenderLarge requests is incomplete or invalid.

### **Encoding:**

1 O Error

1	BEC + 7	<pre>Error code (GLXBadLargeRequest)</pre>
2	CARD16	sequence number
4	CARD32	bad parameter
2	CARD16	minor opcode
1	CARD8	major opcode
21		unused

# GLX Unsupported Private Request

The opcode of a vendor-specific private request is not supported by the server.

## **Encoding:**

1	0	Error
1	BEC + 8	<pre>Error code (GLXUnsupportedPrivateRequest)</pre>
2	CARD16	sequence number
4	CARD32	unsupported opcode
2	CARD16	minor opcode
1	CARD8	major opcode
21		unused

# GLXBadFBConfig

A value for a GLX FBConfig parameter is illegal or does not name a defined GLX FBConfig.

1	O	Error
1	BEC + 9	<pre>Error code (GLXBadFBConfig)</pre>
2	CARD16	sequence number
4	CARD32	bad GLX FBConfig ID
2	CARD16	minor opcode
1	CARD8	major opcode
21		unused

### GLXBadPbuffer

A value for a GLX Pbuffer parameter is illegal or does not name a defined GLX Pbuffer.

# **Encoding:**

1	O	Error
1	BEC + 10	<pre>Error code (GLXBadPbuffer)</pre>
2	CARD16	sequence number
4	CARD32	bad GLX Pbuffer ID
2	CARD16	minor opcode
1	CARD8	major opcode
21		unused

## GLXBadCurrentDrawable

The current drawable of a thread is a window or pixmap that is no longer valid.

## **Encoding:**

1	O	Error
1	BEC + 11	<pre>Error code (GLXBadCurrentDrawable)</pre>
2	CARD16	sequence number
4	CARD32	bad current Drawable ID
2	CARD16	minor opcode
1	CARD8	major opcode
21		unused

## **GLXBadWindow**

A value for a GLX Window parameter is illegal or does not name a defined GLX Window.

1	O	Error
1	BEC + 12	<pre>Error code (GLXBadWindow)</pre>
2	CARD16	sequence number

1.6. EVENTS 9

4	CARD32	bad GLX Window ID
2	CARD16	minor opcode
1	CARD8	major opcode
21		unused

# 1.6 Events

BaseEventCode is the base event code for the extension.

GLX defines one new event,  ${\tt GLX\_PbufferClobber}.$ 

# GLX\_PbufferClobber

1	BaseEventCode + 0	<pre>Event code (GLX_PbufferClobber)</pre>
1		unused
2	CARD16	sequence number
2	CARD16	event_type
	0x8017	GLX_DAMAGED
	0x8018	GLX_SAVED
2	CARD16	draw_type
	0x8019	GLX_WINDOW
	0x801A	GLX_PBUFFER
4	GLX_DRAWABLE	drawable
4	BITFIELD	buffer_mask
2	CARD16	aux_buffer
2	CARD16	X
2	CARD16	y
2	CARD16	width
2	CARD16	height
2	CARD16	count
4		unused

# 1.7 Padding and Unused Bytes

Pad bytes are used to align values on 2, 4, or 8 byte boundaries. The contents of pad bytes are explicitly left undefined. Also, bytes marked as "unused" are specifically left undefined.

# 1.8 Context Tags

All GLX requests that operate on the current rendering context include a GLX\_CONTEXT\_TAG parameter; these *context-specific* requests are **glXWaitX**, **glXWaitGL**, **glXUseXFont**, **glXRender**, **glXRenderLarge**, all GLX non-rendering requests, and in some cases, **glXSwapBuffers** and **glXCopyContext**. (The term *non-rendering request* is defined in Chapter 2, "Requests".) A client may have multiple threads of execution, each possibly having a current context and current drawable; the context tag can be used by the server to identify the current context of the calling thread. Since each context can be current to at most one drawable at a time, the context tag can also be used by the server to identify the current drawable of the calling thread. Any request that contains a context tag can potentially generate a GLXBadContextTag error.

Context tags are generated by the server when a **glXMakeContextCurrent** or **glX-MakeCurrent** request succeeds, returned to the client in the reply, and then sent back to the server in each context specific request. The server may choose any algorithm for generating context tags, but these points should be kept in mind:

- A context tag must be unique per client.
- A context tag may not be freed until the context is no longer current. This is why
  the context resource ID (GLX\_CONTEXT) cannot be used for the tag; the resource
  ID can be freed with the glXDestroyContext request, even while the context is
  current for some client.
- A context tag of 0 has a specific meaning for some requests; see the descriptions
  for each request. glXCopyContext, glXSwapBuffers, glXMakeContextCurrent, and glXMakeCurrent are the only requests where a context tag of zero is
  legal. For all others, a zero tag generates a GLXBadContextTag error.

# **Chapter 2**

# **Requests**

GLX requests can be categorized into three groups:

### Requests for GLX commands

There is a distinct GLX request for most GLX commands.

### Requests for OpenGL non-rendering commands

OpenGL non-rendering commands are those that cannot be placed in a display list. There is a distinct GLX request for each non-rendering command. These requests will be referred to as *GLX non-rendering requests* in the rest of this document.

#### Requests for OpenGL rendering commands

There are two requests, **glXRender** and **glXRenderLarge**, that are used to send OpenGL rendering commands. Rendering commands are exactly the set of OpenGL commands that can be placed in a display list. These two requests will be referred to as *GLX rendering requests* in the rest of this document.

# 2.1 Requests for GLX Commands

## **Query Extension Version**

Name: glXQueryVersion

**Request:** 

client\_major\_version: CARD32

client\_minor\_version: CARD32

### Reply:

server\_major\_version: CARD32
server\_minor\_version: CARD32

Errors: None

#### **Description:**

Client\_major\_version and client\_minor\_version indicate the version of the protocol that the client wants the server to use. If the client and server are compatible then the server returns the version that can actually be supported on the connection – that is, it returns the minimum of the client's minor version number and the server's minor version number. The two protocol versions are compatible if the major versions are the same. If the server does not return a compatible version and the client is not able to use the server's version, the client should terminate.

### **Encoding:**

	1	CARD8	opcode (X assigned)
	1	7	GLX opcode (glXQueryVersion)
	2	3	request length
	4	CARD32	client major version
	4	CARD32	client minor version
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	0	reply length
	4	CARD32	server major version
	4	CARD32	server minor version
	16		unused

## **Query Server String**

Name: glXQueryServerString

**Request:** 

screen: CARD32
name: ENUM

Reply:

length: CARD32

server\_string: STRING8

Errors: BadValue

### **Description:**

This request returns a string describing some aspect of the server's GLX extension. The possible values for name are <code>GLX\_VENDOR</code>, <code>GLX\_VERSION</code>, and <code>GLX\_EXTENSIONS</code>. The format and contents of the vendor string is implementation dependent. The version string is laid out as follows:

```
< major\_version.minor\_version > < space > < vendor - specific - info >
```

Both the major and minor portions of the version number are of arbitrary length. The vendor-specific information is optional. However, if it is present, the format and contents are implementation specific.

The extension string contains a space-separated list of extension names – the extension names themselves do not contain spaces. If there are no extensions to GLX, then the reply length is zero. Note that this string only contains tokens pertaining to GLX extensions.

If screen does not exist, a BadValue error is generated.

#### **Encoding:**

	1	CARD8	opcode (X assigned)
	1	19	<pre>GLX opcode (glXQueryServerString)</pre>
	2	3	request length
	4	CARD32	screen
	4	ENUM	name
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	(n+p)/4	reply length
	4		unused
	4	CARD32	n
	16		unused
	n	STRING8	server string
	p		unused, p=pad(n)

## Send Client OpenGL Information to the Server

Name: glXClientInfo

### **Request:**

client\_major\_number : CARD32
client\_minor\_number : CARD32

length: CARD32

extension\_string: STRING8

Errors: None

### **Description:**

This request is used to inform the server of the OpenGL version and extensions supported by the client library. Note that the client only needs to send the names of the extensions that require support from the server. When the server receives a **GetString** request it uses this information to compute the version and extensions which can be supported on the connection. The GLX client library should append any client-side only extensions to the extension string returned by the **GetString** request.

If this request is never sent to the server, then the server assumes that the client supports OpenGL major version 1 and minor version 0 and doesn't support any extensions.

### **Encoding:**

1	CARD8	opcode (X assigned)
1	20	GLX opcode (glXClientInfo)
2	4+(n+p)/4	request length
4	CARD32	client major OpenGL version number
4	CARD32	client minor OpenGL version number
4	CARD32	number of bytes in extension_string
4	STRING8	extension string
p		unused, p=pad(n)

### **Create a Rendering Context**

Name: glXCreateContext

## **Request:**

context: GLX\_CONTEXT
visual: VISUALID
screen: CARD32

share\_list: GLX\_CONTEXT

is\_direct: BOOL

Errors: BadAlloc, BadMatch, BadValue, GLXBadContext

15

### **Description:**

This request creates a rendering context. The context may be used to render into any GLX drawable created with *visual* on *screen*. If *share\_list* is not 0, then all display list and texture object indices and definitions will be shared by *share\_list* and the newly created rendering context; *share\_list* must share an address space with the new context. If *is\_direct* is False, a rendering context that renders through the X server is created. If *is\_direct* is True, the semantics of this request are implementation dependent.

If *screen* does not exist, a BadValue error is generated. If *visual* is not a valid visual (i.e., it is not a valid X visual, or the GLX implementation does not support this visual on *screen*), a BadValue error is generated. If *share\_list* is not a valid rendering context and is not 0, a GLXBadContext error is generated. If *share\_list* specifies an address space that cannot be shared with the new context, a BadMatch error is generated. BadAlloc is generated if the server does not have enough resources to allocate the new context.

### **Encoding:**

1	CARD8	opcode (X assigned)
1	3	GLX opcode (glXCreateContext)
2	6	request length
4	GLX_CONTEXT	context
4	VISUALID	visual
4	CARD32	screen
4	GLX_CONTEXT	share_list
1	BOOL	is_direct
3		unused

#### **Destroy a Rendering Context**

Name: glXDestroyContext

**Request:** 

context: GLX\_CONTEXT

Errors: GLXBadContext

### **Description:**

This request destroys the resource ID of *context*, and *context* cannot subsequently be made current for any thread of any connection. In addition, for an indirect context, the context itself is freed when it is no longer current to a thread.

If *context* is not a valid rendering context, a GLXBadContext error is generated.

### **Encoding:**

1 CARD8 opcode (X assigned)
1 4 GLX opcode (glXDestroyContext)
2 2 request length
4 GLX\_CONTEXT context

### Make a Rendering Context and a Drawable Current

Name: glXMakeCurrent

### **Request:**

drawable: GLX\_DRAWABLE
context: GLX\_CONTEXT
old\_tag: GLX\_CONTEXT\_TAG

#### Reply:

new\_tag: GLX\_CONTEXT\_TAG

Errors: BadAlloc, BadAccess, BadMatch, GLXBadContext,
GLXBadContextState, GLXBadDrawable, GLXBadContextTag,
GLXBadCurrentWindow

#### **Description:**

This request makes both *context* and *drawable* current to a thread. If the calling thread already has a current context, its tag is sent as *old\_tag* in the request, and that context is designated as no longer being current; additionally, if the context is indirect, any pending GL commands for that context are flushed. If the calling thread does not have a current context, *old\_tag* is 0. If both *context* and *drawable* are 0, the thread is designated as having neither a current context nor a current drawable, and 0 is returned for *new\_tag*; otherwise, a tag referring to the new current context is returned as *new\_tag*.

*new\_tag* will be sent in subsequent requests as described in section 1.8.

If there is already a current context and it is not in rendering mode (i.e., it is in feedback or selection mode), a GLXBadContextState error is generated. If *context* is not a valid rendering context, a GLXBadContext error is generated. If *context* is already current for another thread of any client, a BadAccess error is generated. If *drawable* and *context* are not similar (i.e., they were not created on the same screen and with the

is generated. If the previous context has pending GL commands that have not been flushed (i.e., <code>old\_tag</code> is nonzero), and the previous drawable is a window that is no longer valid, then <code>GLXBadCurrentWindow</code> is generated. A <code>BadAlloc</code> error may be generated if the server tried to allocate resources for the ancillary buffers and failed.

### **Encoding:**

	1	CARD8	opcode (X assigned)
	1	5	<pre>GLX opcode (glXMakeCurrent)</pre>
	2	4	request length
	4	GLX_DRAWABLE	drawable
	4	GLX_CONTEXT	context
	4	GLX_CONTEXT_TAG	old context tag
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	0	reply length
	4	GLX_CONTEXT_TAG	new context tag
	20		unused

# **Query Whether a Rendering Context is Direct**

Name: glXIsDirect

**Request:** 

context: GLX\_CONTEXT

Reply:

is\_direct: BOOL

Errors: GLXBadContext

### **Description:**

This request determines whether *context* is a direct rendering context. If *context* is direct, *is\_direct* is returned as True, otherwise False is returned.

If *context* is not a valid rendering context, a GLXBadContext error is generated.

1	CARD8	opcode (X assigned)
1	6	GLX opcode (glXIsDirect)

	2	2	request length
	4	GLX_CONTEXT	context
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	0	reply length
	1	BOOL	is_direct
	23		unused

# **Copy State From One Rendering Context to Another**

Name: glXCopyContext

### **Request:**

source: GLX\_CONTEXT
dest: GLX\_CONTEXT
mask: BITFIELD

source\_tag: GLX\_CONTEXT\_TAG

Errors: BadAccess, BadMatch, BadValue, GLXBadContext,
GLXBadContextTag, GLXBadCurrentWindow

### **Description:**

Selected groups of state variables are copied from *source* to *dest. Mask* determines which groups of state variables are to be copied; it is the bitwise OR of these symbolic names:

0x0000001	GL_CURRENT_BIT	Current attributes
0x00000002	GL_POINT_BIT	Point attributes
0x0000004	GL_LINE_BIT	Line attributes
0x0000008	GL_POLYGON_BIT	Polygon attributes
0x0000010	GL_POLYGON_STIPPLE_BIT	Polygon stipple attributes
0x00000020	GL_PIXEL_MODE_BIT	Pixel attributes
0x00000040	GL_LIGHTING_BIT	Lighting attributes
0x00000080	GL_FOG_BIT	Fog attributes
0x00000100	GL_DEPTH_BUFFER_BIT	Depth buffer attributes
0x00000200	GL_ACCUM_BUFFER_BIT	Accumulation buffer attributes
0x00000400	GL_STENCIL_BUFFER_BIT	Stencil buffer attributes
0x00000800	GL_VIEWPORT_BIT	Viewport attributes
0x00001000	GL_TRANSFORM_BIT	Transform attributes
0x00002000	GL_ENABLE_BIT	State of modes that can be enabled or disabled
0x00004000	GL_COLOR_BUFFER_BIT	Color buffer attributes

0x00008000	GL_HINT_BIT	Hints
0x00010000	GL_EVAL_BIT	Evaluator attributes
0x00020000	GL_LIST_BIT	List attributes
0x00040000	GL_TEXTURE_BIT	Texture attributes
0x00080000	GL_SCISSOR_BIT	Scissor attributes
0x000fffff	GL_ALL_ATTRIB_BITS	All possible attributes

These are the same symbolic names used for glPushAttrib in the OpenGL Spec.

If <code>source\_tag</code> is not 0, any pending GL commands for the context identified by <code>source\_tag</code> are completed before the copy occurs. In this case, <code>GLXBadContextTag</code> is generated if the tag is invalid, and <code>GLXBadCurrentWindow</code> is generated if the current drawable associated with the context is a window that is no longer valid.

If *source* and *dest* do not share an address space, or were not created on the same screen, a BadMatch error is generated. If *source\_tag* does not refer to the same context as *source*, a BadMatch error is generated. If *dest* is current for some thread, even if it's the calling thread, a BadAccess error is generated. If either *source* or *dest* is not a valid rendering context, a GLXBadContext error is generated.

It is not an error to specify undefined bits in mask.

### **Encoding:**

1	CARD8	opcode (X assigned)
1	10	GLX opcode (glXCopyContext)
2	5	request length
4	GLX_CONTEXT	source context
4	GLX_CONTEXT	destination context
4	BITFIELD	mask
4	GLX_CONTEXT_TAG	source context tag

### **Complete GL Execution Prior to Subsequent X Requests**

Name: glXWaitGL

**Request:** 

tag: GLX\_CONTEXT\_TAG

Errors: GLXBadContextTag, GLXBadCurrentWindow

**Sequentiality:** 

Before describing the semantics of this request, the sequentiality of GLX requests is discussed. Although GLX and X requests are transported by a connection in one physical stream, they are logically in separate streams: a GL stream for each calling thread, and one single X stream.

Requests that are only in the GL stream are:

```
all GLX non-rendering requests (see Section 2.2 for a definition)
glXRender
glXRenderLarge
glXWaitX
glXSwapBuffers, if the context tag parameter is nonzero
```

Requests that are only in the X stream are:

```
glXCreateContext
glXDestroyContext
glXMakeCurrent
glXIsDirect
glXGetVisualConfigs
glXQueryExtensionsString
glXQueryServerString
glXQueryVersion
glXWaitGL
glXCreateGLXPixmap
glXDestroyGLXPixmap
glXSwapBuffers, if the context tag parameter is zero
glXCopyContext, if the context tag parameter is zero
glXCreatePbuffer
glXDestroyPbuffer
glXCreatePixmap
glXDestroyPixmap
glXCreateWindow
glXDestroyWindow
glXMakeContextCurrent
glXCreateNewContext
glXGetFBConfigs
glXQueryContext
glXGetDrawableAttributes
{\bf glXChangeDrawableAttributes}
```

Requests that are in both the GL and X streams are:

glXUseXFont
glXCopyContext, if the context tag parameter is nonzero

All requests that are in the GL stream (including those that are in both streams) of the calling thread will contain the context tag of the current context for that thread.

### **Description:**

Requests in the GL stream (of the calling thread) that precede the **glXWaitGL** request are guaranteed to be executed before requests in the X stream that follow the **glXWaitGL** request.

*Tag* is the tag for the current context of the calling thread.

A GLXBadContextTag error is generated if *tag* is an invalid tag. A GLXBadCurrentWindow error is generated if the current drawable of the calling thread is a window that is no longer valid.

#### **Encoding:**

1	CARD8	opcode (X assigned)
1	8	GLX opcode (glXWaitGL)
2	2	request length
4	GLX_CONTEXT_TAG	context tag

### **Complete X Execution Prior to Subsequent GL Requests**

Name: glXWaitX

**Request:** 

taq: GLX\_CONTEXT\_TAG

 $\textbf{Errors:} \ \texttt{GLXBadContextTag}, \ \texttt{GLXBadCurrentWindow}$ 

#### **Description:**

Requests in the X stream that precede the **glXWaitX** request are guaranteed to be executed before requests in the GL stream (of the calling thread) that follow the **glXWaitX** request. See discussion of **glXWaitGL** for a description of the two streams.

*Tag* is the tag for the current context of the calling thread.

A GLXBadContextTag error is generated if *tag* is an invalid tag. A GLXBadCurrentWindow error is generated if the current drawable of the calling

thread is a window that is no longer valid.

### **Encoding:**

1	CARD8	opcode (X assigned)
1	9	<pre>GLX opcode (glXWaitX)</pre>
2	2	request length
4	GLX_CONTEXT_TAG	context tag

### **Exchange Front and Back Buffers**

Name: glXSwapBuffers

### **Request:**

tag: GLX\_CONTEXT\_TAG
drawable: GLX\_DRAWABLE

Errors: GLXBadContextTag, GLXBadCurrentWindow, GLXBadDrawable

#### **Description:**

**glXSwapBuffers** exchanges the front and back buffers of *drawable*. This exchange typically takes place during the vertical retrace of the monitor, rather than immediately after the **glXSwapBuffers** request is received. All rendering contexts using this drawable share the same notion of which are front buffers and which are back buffers. This notion is also shared with the X double-buffering extension (DBE).

If *tag* is not 0, any pending GL commands for the context identified by *tag* are completed before the buffer swap occurs.

If *drawable* was not created with respect to a doublebuffer visual, or if *drawable* is a GLX pixmap then **glXSwapBuffers** has no effect, and no error is generated.

If drawable is not a valid GLX drawable, a GLXBadDrawable error is generated.

Two errors may be generated if *tag* is not 0: a GLXBadContextTag error is generated if *tag* is an invalid tag, and a GLXBadCurrentWindow error is generated if the current drawable of the calling thread is a window that is no longer valid.

1	CARD8	opcode (X assigned)
1	11	GLX opcode (glXSwapBuffers)
2	3	request length

4 GLX\_CONTEXT\_TAG context tag
4 GLX\_DRAWABLE drawable

## Create Bitmap Display Lists From an X Font

Name: glXUseXFont

#### **Request:**

tag: GLX\_CONTEXT\_TAG

font: FONT first: CARD32 count: CARD32 list\_base: CARD32

Errors: BadFont, GLXBadContextState, GLXBadContextTag,
GLXBadCurrentWindow

#### **Description:**

**glXUseXFont** generates count display lists, named  $list\_base$  through  $list\_base + count - 1$ , each containing a single **Bitmap** command. The parameters of the **Bitmap** command of display list  $list\_base + i$  are derived from glyph first + i of font, where  $0 \le i < count$ . **Bitmap** parameters xorig, yorig, width, and height are computed from font metrics as -lbearing, descent - 1, rbearing - lbearing, and ascent + descent respectively. Xmove is taken from the glyph's width metric, and ymove is set to zero. Finally, the glyph's image is converted to the appropriate format for **Bitmap**.

Empty display lists are created for all glyphs that are requested and not defined in font.

*Tag* is the tag for the current context of the calling thread. Any pending GL commmands for this context are flushed.

A BadFont error is generated if font is not a valid X font. A GLXBadContextState error is generated if the current context is already constructing a display list. A GLXBadContextTag error is generated if tag is an invalid tag. A GLXBadCurrentWindow error is generated if the current drawable of the calling thread is a window that is no longer valid.

1	CARD8	code (X assigned)
1	12	GLX opcode (glXUseXFont)
2	6	request length
4	GLX_CONTEXT_TAG	context tag

4	FONT	font
4	CARD32	first
4	CARD32	count
4	CARD32	list base

### Create an Offscreen Rendering Area

Name: glXCreateGLXPixmap

## **Request:**

screen: CARD32
visual: VISUALID
pixmap: PIXMAP

glx\_pixmap: GLX\_PIXMAP

Errors: BadAlloc, BadMatch, BadPixmap, BadValue

#### **Description:**

glXCreateGLXPixmap creates an offscreen rendering area. Any rendering context that is created with respect to *visual* on *screen* can be used to render into this offscreen area.

The X pixmap identified by *pixmap* is used for the RGB planes of the front-left buffer of the resulting GLX offscreen rendering area. All other buffers specified by *visual* are created without externally visible names. GLX pixmaps may be created with a visual that includes back buffers and stereoscopic buffers; however, the **glXSwapBuffers** request is ignored for these pixmaps. The resource ID of the new GLX pixmap is  $g/x_-$  *pixmap*.

A direct rendering context might not be able to be made current with a GLX pixmap.

A BadMatch error is generated if the depth of *pixmap* does not match the depth value reported by core X11 for *visual*, or if *pixmap* was not created with respect to the same screen as *visual*. A BadValue error is generated if *visual* is not a valid visual (i.e., the GLX implementation does not support this visual on *screen*). A BadPixmap error is generated if *pixmap* is not a valid pixmap. If the server cannot allocate the GLX pixmap, a BadAlloc error is generated.

1	CARD8	opcode (X assigned)
1	13	<pre>GLX opcode (glXCreateGLXPixmap)</pre>
2	5	request length
4	CARD32	screen

property\_list consists of num\_visuals groups each containing num\_properties words. Each group describes a visual and consists of 18 ordered properties followed by an unordered list of properties. All the property values are 32 bits. The ordered properties are:

visual: VISUALID class: CARD32 rgba: BOOL32 red\_size: CARD32 areen\_size: CARD32 blue\_size: CARD32 alpha\_size: CARD32 accum\_red\_size: CARD32 accum\_green\_size: CARD32 accum\_blue\_size : CARD32 accum\_alpha\_size: CARD32 double\_buffer: BOOL32

stereo: BOOL32 buffer\_size: CARD32 depth\_size: CARD32 stencil\_size: CARD32 aux\_buffers: CARD32 level: INT32

Each entry in the list of visual properties that follows consists of a 32 bit property type and a 32 bit property value.

Errors: BadValue

#### **Description:**

This request asks for the configurations of all visuals that the GLX implementation supports on the given screen. Class is the class of the visual. Rgba is a boolean indicating whether RGBA or color index rendering is supported. Red\_size, green\_size, blue\_size and alpha\_size respectively specify the number of bits of red, green, blue, and alpha in the color buffer. Accum\_red\_size, accum\_green\_size, accum\_blue\_size, and accum\_alpha\_size specify the number of bits for the respective component in the accumulation buffer. Double\_buffer indicates whether color buffers have front/back pairs that can be swapped, and stereo indicates whether color buffers have left/right pairs. Buffer\_size specifies the depth of the color buffer; for TrueColor and DirectColor visuals buffer\_size is the sum of red\_size, green\_size, blue\_size, and alpha\_size, and for PseudoColor and StaticColor visuals it is the size of the indexes stored in the framebuffer. Depth\_size specifies the number of bits in the depth buffer, and stencil\_size specifies the number of bits in the stencil buffer. Aux\_buffers

Currently *property\_list* is for vendor-specific visual properties. In the future new GLX visual properties may be returned in the list.

If the GLX implementation does not support the given screen, both *num\_visuals* and *num\_properties* will be 0, and no properties will be returned.

If screen is not a valid screen, a BadValue error is generated.

#### **Encoding:**

	1	CARD8	opcode (X assigned)
	1	14	GLX opcode (glXGetVisualConfigs)
	2	2	request length
	4	CARD32	screen
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	n	reply length
	4	CARD32	num_visuals
	4	CARD32	num_properties
	16		unused
	4*n	List of 32 bit values	properties

Where  $n=num\_visuals*num\_properties$ . Each property value is either a VISUALID, CARD32, BOOL32, or INT32. The first 18 properties are ordered; the remaining properties consist of a property type and property value. Thus, the actual number of property values is (num\\_properties > 18)? ((num\\_properties - 18)/2 + 18): (num\\_properties)

## **Vendor-specific Private Request**

Name: glXVendorPrivate

#### **Request:**

opcode: CARD32
data: LISTofBYTE

Errors: GLXUnsupportedPrivateRequest

#### **Description:**

This request is for vendor-specific commands.

 $\boldsymbol{A}$  GLXUnsupportedPrivateRequest error is generated if the server does not support the opcode.

#### **Encoding:**

```
1
                                 opcode (X assigned)
      CARD8
1
      16
                                 GLX opcode (glXVendorPrivate)
                                 request length
2
      2+(n+p)/4
4
                                 vendor-specific opcode
      CARD32
n
      LISTofBYTE
                                 vendor-specific data
                                 unused, p=pad(n)
p
```

## **Vendor-specific Private Request with Reply**

Name: glXVendorPrivateWithReply

#### **Request:**

opcode: CARD32
data: LISTofBYTE

#### **Reply:**

returned\_data: LISTofBYTE

Errors: GLXUnsupportedPrivateRequest

### **Description:**

This request is for vendor-specific commands that need returned data.

 $\boldsymbol{A}$  GLXUnsupportedPrivateRequest error is generated if the server does not support the opcode.

	1	CARD8	opcode (X assigned)
	1	17	GLX opcode (glXVendorPrivateWithReply)
	2	2+(m+p)/4	request length
	4	CARD32	vendor-specific opcode
	m	LISTOfBYTE	vendor-specific data
	p		unused, p=pad(m)
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number

4	n	reply length
24	LISTofBYTE	returned data
4*n	LISTofBYTE	more returned data

## **Get List of Frame Buffer Configurations**

Name: glXGetFBConfigs

**Request:** 

screen: CARD32

Reply:

num\_fbconfigs: CARD32
num\_properties: CARD32

property\_list: LISTOfATTRIBUTE\_PAIR

property\_list consists of <code>num\_fbconfigs</code> groups each containing <code>num\_properties</code> entries. Each group describes a frame buffer configuration and consists of an unordered list of properties. Each entry in the list consists of a 32 bit property type and a 32 bit property value. The property types are the same as the <code>GLXBadfbConfig</code> attributes described in the GLX Specification.

This request may be used by the **glXChooseFBConfig** and **glXGetFBConfigs** entry points.

Errors: BadValue

#### **Description:**

This request asks for all the frame buffer configurations that the GLX implementation supports on the given screen.

If the GLX implementation does not support the given screen, both *num\_fbconfigs* and *num\_properties* will be 0, and no properties will be returned.

If screen is not a valid screen, a BadValue error is generated.

1	CARD8	opcode (X assigned)
1	21	GLX opcode (glXGetFBConfigs)
2	2	request length
4	CARD32	screen

1	1	Reply
1		unused
2	CARD16	sequence number
4	n	reply length
4	CARD32	num_fbconfigs
4	CARD32	num_properties
16		unused
4*n	LISTofATTRIBUTE_PAIR	attribute, value pairs

Where  $n=2*num\_fbconfigs*num\_properties$ . Each property value is either an FBCONFIGID, CARD32, BOOL32, or INT32. The properties consist of a property type and property value.

# Create an Offscreen Rendering Area from Frame Buffer Configuration

Name: glXCreatePixmap

#### **Request:**

screen: CARD32
fbconfig: FBCONFIGID
pixmap: PIXMAP

glx\_pixmap: glx\_pixmap
num\_attributes: card32

attrib\_list: LISTOFATTRIBUTE\_PAIR

attrib\_list contains num\_attributes entries. Each entry in the list consists of a 32 bit attribute type and a 32 bit attribute value.

Errors: BadAlloc, BadMatch, BadPixmap, GLXBadFBConfig

#### **Description:**

This request creates an offscreen rendering area. Any rendering context that is created with respect to *fbconfig* on *screen* can be used to render into this offscreen area.

The X pixmap identified by *pixmap* is used for the RGB planes of the front-left buffer of the resulting GLX offscreen rendering area. All other buffers specified by *fbconfig* are created without externally visible names. GLX pixmaps may be created with a visual that includes back buffers and stereoscopic buffers; however, the **glXSwapBuffers** request is ignored for these pixmaps. The resource ID of the new GLX pixmap is glx-pixmap.

A direct rendering context might not be able to be made current with a GLX pixmap.

A BadMatch error is generated if *pixmap* was not created with respect to the same screen as *fbconfig*, or if the depth of *pixmap* does not match the color buffer depth of *fbconfig*. GLXBadFBConfig is generated if *fbconfig* is not a valid fbconfig (i.e., the GLX implementation does not support this fbconfig on *screen*), or if *fbconfig* does not support rendering to pixmaps. BadPixmap is generated if *pixmap* is not a valid pixmap. Finally, if the server cannot allocate the GLX pixmap, a BadAlloc error is generated.

#### **Encoding:**

1	CARD8	opcode (X assigned)
1	22	GLX opcode (glXCreatePixmap)
2	6+n	request length
4	CARD32	screen
4	FBCONFIGID	fbconfig
4	PIXMAP	pixmap
4	GLX_PIXMAP	glx_pixmap
4	CARD32	num_attributes
4*n	LISTofATTRIBUTE_PAIR	attribute, value pairs

Where  $n = 2 * num\_attributes$ . No attributes are currently allowed.

# **Destroy an Offscreen Rendering Area Created with Frame Buffer Configuration**

Name: glXDestroyPixmap

**Request:** 

glx\_pixmap: GLX\_PIXMAP

Errors: GLXBadPixmap

## **Description:**

This request destroys the resource ID of *glx\_pixmap*, and *glx\_pixmap* cannot subsequently be made current to any thread of any connection. In addition, the GLX pixmap itself is freed when it is no longer current to a thread. The X pixmap that the GLX pixmap was created with is not freed until there are no references to it.

This request should be used only for resource IDs created by **glXCreatePixmap**. GLX pixmaps created by **glXCreateGLXPixmap** should be destroyed with **glXDestroyGLXPixmap**.

GLXBadPixmap is generated if *glx\_pixmap* is not a valid GLX pixmap.

#### **Encoding:**

1 CARD8 opcode (X assigned)
1 23 GLX opcode (glXDestroyPixmap)
2 2 request length
4 GLX\_PIXMAP glx\_pixmap

## **Create a Rendering Context from Frame Buffer Configuration**

Name: glXCreateNewContext

## Request:

context: GLX\_CONTEXT
fbconfig: FBCONFIGID
render\_type: CARD32
Screen: CARD32

share\_list: GLX\_CONTEXT

is\_direct: BOOL

**Errors:** BadAlloc, BadMatch, BadValue, GLXBadContext, GLXBadFBConfig

#### **Description:**

This request creates a rendering context with respect to the specified frame buffer configuration. The context may be used to render into any *compatible* GLX drawable created on *screen* (the definition of compatible is given in the GLX Specification). If *share\_list* is not 0, then all display list and texture object indices and definitions will be shared by *share\_list* and the newly created rendering context; *share\_list* must share an address space with the new context. If *is\_direct* is False, a rendering context that renders through the X server is created. If *is\_direct* is True, the semantics of this request are implementation dependent.

If *screen* does not exist, or if *render\_type* does not refer to a valid rendering type, a BadValue error is generated. If *fbconfig* is not a valid fbconfig, a GLXBadFBConfig error is generated. If *share\_list* is not a valid rendering context and is not 0, a GLXBadContext error is generated. If *share\_list* specifies an address space that cannot be shared with the new context, a BadMatch error is generated. BadAlloc is generated if the server does not have enough resources to allocate the new context.

#### **Encoding:**

1 CARD8 opcode (X assigned)

1	24	<pre>GLX opcode (glXCreateNewContext)</pre>
2	7	request length
4	GLX_CONTEXT	context
4	FBCONFIGID	fbconfig
4	CARD32	screen
4	CARD32	render_type
4	GLX_CONTEXT	share_list
1	BOOL	is_direct
1	CARD8	reserved1
2	CARD16	reserved2

# **Query Context Attributes**

Name: glXQueryContext

**Request:** 

ctx: GLX\_CONTEXT

#### Reply:

num\_attributes: CARD32

attribute\_list: LISTofATTRIBUTE\_PAIR

attribute\_list contains num\_attributes entries. Each entry in the list consists of a 32 bit attribute type and a 32 bit attribute value.

Errors: GLXBadContext

## **Description:**

This request asks for all the attributes of the specified context. Attributes include  $\texttt{GLX\_FBCONFIG\_ID}$ ,  $\texttt{GLX\_RENDER\_TYPE}$ , and  $\texttt{GLX\_SCREEN}$ .

	1	CARD8	opcode (X assigned)
	1	25	GLX opcode (glXQueryContext)
	2	2	request length
	4	GLX_CONTEXT	context
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	n	reply length

4 CARD32 num\_attributes
20 unused
4\*n LISTOFATTRIBUTE\_PAIR attribute, value pairs

Where  $n=2*num\_attributes$ . Each attribute value is either an FBCONFIGID, CARD32, or INT32.

If *context* is not a valid rendering context, a GLXBadContext error is generated, *num\_attributes* will be 0, and no attributes will be returned.

## Make a Rendering Context and Read/Draw Drawables Current

Name: glXMakeContextCurrent

#### **Request:**

old\_tag: GLX\_CONTEXT\_TAG
drawable: GLX\_DRAWABLE
read\_drawable: GLX\_DRAWABLE

 $context: {\tt GLX\_CONTEXT}$ 

#### **Reply:**

new\_tag: GLX\_CONTEXT\_TAG

**Errors:** BadAccess, BadAlloc, BadMatch, GLXBadContext, GLXBadContextState, GLXBadCurrentDrawable, GLXBadDrawable, GLXBadWindow

#### **Description:**

This request makes all of *context*, *drawable*, and *read\_drawable* current to a thread. If the calling thread already has a current context, its tag is sent as *old\_tag* in the request, and that context is designated as no longer being current; additionally, if the context is indirect, any pending GL commands for that context are flushed.

If the calling thread does not have a current context, *old\_tag* is 0. If all of *context*, *drawable*, and *read\_drawable* are 0, the thread is designated as having neither a current context nor a current drawable or read drawable and 0 is returned for *new\_tag*; otherwise, a tag referring to the new current context is returned as *new\_tag*. *new\_tag* will be sent in subsequent requests as described in section 1.8.

If *context* is current to another thread, a BadAccess error is generated. If either of *drawable* or *read\_drawable* are not compatible with *context*, a BadMatch error is generated. If *context* is not a valid rendering context, a GLXBadContext error is generated. If another context is current and its render mode is either GL\_-

FEEDBACK or GL\_SELECT, a GLXBadContextState error is generated. If the previous context has unflushed commands, and the previous drawable is no longer valid, a GLXBadCurrentDrawable error is generated. If either of *drawable* or *read\_drawable* are not valid drawables, a GLXBadDrawable error is generated. If the X Window underlying either *drawable* or *read\_drawable* is no longer valid, a GLXBadWindow error is generated. If the server does not have enough resources to allocate the new context, a BadAlloc error is generated.

Finally, implementations may generate a BadMatch error under the following conditions:

- If *drawable* and *read\_drawable* cannot fit into framebuffer memory simultaneously.
- If drawable or read\_drawable is a GLXPixmap and context is a direct rendering context.
- If drawable or read\_drawable is a GLXPixmap and context was previously bound to a GLXWindow or GLXPbuffer.
- If drawable or read\_drawable is a GLXWindow or GLXPbuffer and context was previously bound to a GLXPixmap.
- If *context* is NULL and *drawable* and *read\_drawable* are not None, or if *drawable* or *read\_drawable* are set to None and *context* is not NULL.

	1	CARD8	opcode (X assigned)
	1	26	<pre>GLX opcode (glXMakeContextCurrent)</pre>
	2	5	request length
	4	GLX_CONTEXT_TAG	old context tag
	4	GLX_DRAWABLE	drawable
	4	GLX_DRAWABLE	read_drawable
	4	GLX_CONTEXT	context
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	0	reply length
	4	GLX_CONTEXT_TAG	new context tag
	20		unused

#### Create an Offscreen Pixel Buffer

Name: glXCreatePbuffer

#### **Request:**

screen: CARD32
fbconfig: FBCONFIGID
glx\_pbuffer: GLX\_PBUFFER
num\_attributes: CARD32

attribute\_list: LISTOFATTRIBUTE\_PAIR

attribute\_list contains num\_attributes entries. Each entry in the list consists of a 32 bit attribute type and a 32 bit attribute value.

Errors: BadAlloc, BadMatch, GLXBadFBConfig

#### **Description:**

This request creates an offscreen rendering area designed to be located in non-visible frame buffer memory. Any rendering context that is created with respect to *fbconfig* on *screen* can be used to render into this offscreen area.

All buffers specified by *fbconfig* are created without externally visible names. GLX pbuffers may be created with an fbconfig that includes back buffers and stereoscopic buffers; however, the **glXSwapBuffers** request is ignored for these pbuffers. The resource ID of the new GLX pbuffer is *glx\_pbuffer*.

A direct rendering context must be able to be made current with a GLX pbuffer.

If config is not a valid GLXFBConfig, a GLXBadFBConfig error is generated. If fbconfig does not support GLXPbuffers, a BadMatch error is generated. If the server does not have enough resources to allocate the pbuffer, a BadAlloc error is generated.

1	CARD8	opcode (X assigned)
1	27	GLX opcode (glXCreatePbuffer)
2	5+n	request length
4	CARD32	screen
4	FBCONFIGID	fbconfig
4	GLX_PBUFFER	glx_pbuffer
4	CARD32	num_attributes
4*n	LISTofATTRIBUTE_PAIR	attribute, value pairs

Where  $n = 2 * num\_attributes$ . Each attribute value is either an BOOL32 or CARD32.

## Destroy an Offscreen Pixel Buffer

Name: glXDestroyPbuffer

**Request:** 

glx\_pbuffer: GLX\_PBUFFER

Errors: GLXBadPbuffer

#### **Description:**

This request destroys the resource ID of *glx\_pbuffer*, and *glx\_pbuffer* cannot subsequently be made current to any thread of any connection. In addition, the GLX pbuffer itself is freed when it is no longer current to a thread.

If *glx\_pbuffer* is not a valid GLX pbuffer, a GLXBadPbuffer error is generated.

## **Encoding:**

1	CARD8	opcode (X assigned)
1	28	GLX opcode (glXDestroyPbuffer)
2	2	request length
4	GLX_PBUFFER	glx_pbuffer

#### **Get List of Drawable Attributes**

Name: glXGetDrawableAttributes

**Request:** 

drawable: GLX\_DRAWABLE

Reply:

num\_attributes: CARD32

attribute\_list: LISTOFATTRIBUTE\_PAIR

attribute\_list contains num\_attributes entries. Each entry in the list consists of a 32 bit attribute type and a 32 bit attribute value.

This request may be used by the **glXGetSelectedEvent** and **glXQueryDrawable** entry points.

Errors: GLXBadDrawable

### **Description:**

This request asks for all the attributes of the specified drawable. Attributes may include GLX\_WIDTH, GLX\_HEIGHT, GLX\_PRESERVED\_CONTENTS, GLX\_LARGEST\_PBUFFER, GLX\_FBCONFIG\_ID, and GLX\_EVENT\_MASK.

If drawable is not a valid GLX drawable, a GLXBadDrawable error is generated.

## **Encoding:**

	1	CARD8 <b>29</b>	<pre>opcode (X assigned) GLX opcode (glXGetDrawableAttributes)</pre>
	2	2	request length
	4	GLX_DRAWABLE	drawable
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	n	reply length
	4	CARD32	num_attributes
	20		unused
	4*n	LISTofATTRIBUTE_PAIR	attribute, value pairs

Where  $n=2*num\_attributes$ . Each attribute value is either an <code>FBCONFIGID</code>, <code>BOOL32</code>, or <code>CARD32</code>.

## **Change Drawable Attributes**

Name: glXChangeDrawableAttributes

#### **Request:**

drawable: GLX\_DRAWABLE
num\_attributes: CARD32

attribute\_list: LISTOFATTRIBUTE\_PAIR

attribute\_list contains num\_attributes entries. Each entry in the list consists of a 32 bit attribute type and a 32 bit attribute value.

This request may be used by the **glXSelectEvent** entry point.

Errors: BadDrawable, BadValue

#### **Description:**

This request changes attributes of the specified drawable. Currently the only attribute which may be changed is GLX\_EVENT\_MASK.

If *drawable* is not a valid GLX drawable, a GLXBadDrawable error is generated. If an attribute other than GLX\_EVENT\_MASK is specified, or if the attribute value for GLX\_EVENT\_MASK has any bits set other than GLX\_PBUFFER\_CLOBER\_MASK, a BadValue error is generated.

#### **Encoding:**

1	CARD8	opcode (X assigned)
1	30	GLX opcode (glXChangeDrawableAttributes)
2	3+n	request length
4	GLX_DRAWABLE	drawable
4	CARD32	num_attributes
4*n	LISTofATTRIBUTE_PAIR	attribute, value pairs

Where  $n = 2 * num\_attributes$ . Each attribute value is a CARD32.

#### Create a Window

Name: glXCreateWindow

#### **Request:**

screen: CARD32
fbconfig: FBCONFIGID
window: WINDOW
glx\_window: GLX\_WINDOW
num\_attributes: CARD32

attribute\_list: LISTOFATTRIBUTE\_PAIR

attribute\_list contains num\_attributes entries. Each entry in the list consists of a 32 bit attribute type and a 32 bit attribute value.

Errors: BadMatch, GLXBadFBConfig, BadWindow, BadAlloc

## **Description:**

This request creates an onscreen rendering area. Any rendering context that is created with respect to *fbconfiq* on *screen* can be used to render into this onscreen area.

The X window identified by *window* is used for the RGB planes of the front-left buffer of the resulting GLX onscreen rendering area. All other buffers specified by *fbconfig* 

are created without externally visible names. The resource ID of the new GLX window is *glx\_window*.

A BadMatch error is generated if *window* was not created with respect to the same screen as *fbconfig*, if the depth value reported by core X11 for *window* does not match the color buffer depth of *fbconfig*, or if *fbconfig* does not support rendering to windows. GLXBadFBConfig is generated if *fbconfig* is not a valid fbconfig (i.e., the GLX implementation does not support this fbconfig on *screen*). GLXBadWindow is generated if *window* is not a valid X window. Finally, if the server cannot allocate the GLX window, or if there is already an fbconfig associated with *window*, a BadAlloc error is generated.

#### **Encoding:**

1	CARD8	opcode (X assigned)
1	31	GLX opcode (glXCreateWindow)
2	6+n	request length
4	CARD32	screen
4	FBCONFIGID	fbconfig
4	WINDOW	window
4	GLX_WINDOW	glx_window
4	CARD32	num_attributes
4*n	LISTOFATTRIBUTE_PAIR	attribute, value pairs

Where  $n = 2 * num\_attributes$ . No attributes are currently defined.

## **Destroy a Window**

Name: glXDestroyWindow

**Request:** 

glx\_window: GLX\_WINDOW

Errors: GLXBadWindow

#### **Description:**

This request destroys the resource ID of *glx\_window*, and *glx\_window* cannot subsequently be made current to any thread of any connection. In addition, the GLX window itself is freed when it is no longer current to a thread. The X window that the GLX window was created with is not freed until there are no references to it.

GLXBadWindow is generated if *glx\_window* is not a valid GLX window.

## **Encoding:**

1 CARD8 opcode (X assigned)
1 32 GLX opcode (glXDestroyWindow)
2 2 request length
4 GLX\_WINDOW glx\_window

# 2.2 Requests for GL Non-rendering Commands

# 2.2.1 GL Non-rendering Commands That Do Not Return Pixel Data

The requests in this section correspond to GL commands that cannot be put into a display list. Unlike the **glXRender** request, each of these requests always contains just one GL command.

These requests are all context-specific; hence, they all include a context tag. All of these requests will generate a GLXBadContextTag error if the context tag parameter is invalid.

#### AreTexturesResident

	1	CARD8	opcode (X assigned)
	1	143	GLX opcode
	2	1	request length
	4	GLX_CONTEXT_TAG	context tag
	4	INT32	n
	n*4	LISTofCARD32	textures
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	(n+p)/4	reply length
	4	BOOL32	return value
	20		unused
	n*1	LISTofBOOL	residences
	p		unused, p=pad(n)

#### **DeleteLists**

1	CARD8	opcode (X assigned)
1	103	GLX opcode
2	4	request length
4	GLX_CONTEXT_TAG	context tag
4	CARD32	list
4	INT32	range

## **DeleteTextures**

CARD8	opcode (X assigned)
144	GLX opcode
1	request length
GLX_CONTEXT_TAG	context tag
INT32	n
LISTofCARD32	textures
	144 1 GLX_CONTEXT_TAG INT32

## **EndList**

1	CARD8	opcode (X assigned)
1	102	GLX opcode
2	2	request length
4	GLX_CONTEXT_TAG	context tag

## FeedbackBuffer

1	CARD8	opcode (X assigned)
1	105	GLX opcode
2	4	request length
4	GLX_CONTEXT_TAG	context tag
4	INT32	size
4	ENUM	type

Feedback data is returned in the reply of the next **RenderMode** request.

## Finish

1	CARD8	opcode (X assigned)
1	108	GLX opcode

	2	2	request length
	4	GLX_CONTEXT_TAG	context tag
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	0	reply length
	24		unused

## Flush

1	CARD8	opcode (X assigned)
1	142	GLX opcode
2	2	request length
4	GLX_CONTEXT_TAG	context tag

# GenLists

⇒	1 1 2 4 4	CARD8 104 3 GLX_CONTEXT_TAG INT32	opcode (X assigned) GLX opcode request length context tag range
7	1 1 2 4 4 20	1 CARD16 0 CARD32	Reply unused sequence number reply length return value unused

## GenTextures

1	CARD8	opcode (X assigned)
1	145	GLX opcode
2	2 3	request length
4	GLX_CONTE	EXT_TAG context tag
4	INT32	n
$\Rightarrow$		
1	. 1	Reply

1		unused
2	CARD16	sequence number
4	n	reply length
24		unused
n*4	LISTofCARD32	textures

## GetBooleanv

	1 1 2 4 4	CARD8 112 3 GLX_CONTEXT_TAG ENUM	opcode (X assigned) GLX opcode request length context tag pname
$\Rightarrow$	1 1 2 4 4 4	1 CARD16 m CARD32	Reply unused sequence number reply length, m = (n==1 ? 0 : (n+p)/4) unused n
	if (n=1	) this follows:	
	1 15	BOOL	params unused
	otherw	ise this follows:	
	16 n p	LISTofBOOL	unused params unused, p=pad(n)

Note that n may be zero, indicating that a GL error occured.

## GetClipPlane

```
1 CARD8 opcode (X assigned)
1 113 GLX opcode
2 3 request length
4 GLX_CONTEXT_TAG context tag
4 ENUM plane
```

If the command succeeds, 4 doubles are sent in the reply:

1	1	Reply
1		unused
2	CARD16	sequence number
4	8	reply length
24		unused
32	LISTofFLOAT64	equation

Otherwise an empty reply is sent, indicating that a GL error occurred:

1	1	Reply
1		unused
2	CARD16	sequence number
4	0	reply length
24		unused

## GetColorTableParameterfv

	1 1 2 4 4 4	CARD8 148 4 GLX_CONTEXT_TAG ENUM ENUM	opcode (X assigned) GLX opcode request length context tag target pname
$\Rightarrow$	1 1 2 4 4 4	1 CARD16 m CARD32	Reply unused sequence number reply length, m = (n==1 ? 0 : n) unused n
	if (n=1)	) this follows:	
	4 12	FLOAT32	params unused
	otherw	ise this follows:	
	16 n*4	LISTofFLOAT32	unused params

Note that n may be zero, indicating that a GL error occurred.

## **GetColorTableParameteriv**

	1	CARD8	opcode (X assigned)
	1	149	GLX opcode
	2	4	request length
	4	GLX_CONTEXT_TAG	context tag
	4	ENUM	target
	4	ENUM	pname
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	m	reply length, $m = (n==1 ? 0 : n)$
	4		unused
	4	CARD32	n
	if (n=1	) this follows:	
	4	INT32	params
	12		unused
	otherw	ise this follows:	
	16		unused
	n*4	LISTofINT32	params

Note that n may be zero, indicating that a GL error occurred.

## **GetConvolutionParameterfy**

	1	CARD8	opcode (X assigned)
	1	151	GLX opcode
	2	4	request length
	4	GLX_CONTEXT_TAG	context tag
	4	ENUM	target
	4	ENUM	pname
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	m	reply length, $m = (n==1 ? 0 : n)$

Version 1.3 - 2 June 1999

4 unused 4 CARD32 n

if (n=1) this follows:

4 FLOAT32 params 12 unused

otherwise this follows:

16 unused n\*4 LISTofFLOAT32 params

Note that n may be zero, indicating that a GL error occurred.

#### **GetConvolutionParameteriv**

⇒	1 1 2 4 4 4	CARD8 152 4 GLX_CONTEXT_TAG ENUM ENUM	opcode (X assigned) GLX opcode request length context tag target pname
⇒	1 1 2 4 4 4	1 CARD16 m CARD32	Reply unused sequence number reply length, m = (n==1 ? 0 : n) unused n
	if (n=1	) this follows:	
	4 12	INT32	params unused

otherwise this follows:

16 unused n\*4 LISTofINT32 params

Note that n may be zero, indicating that a GL error occurred.

## GetDoublev

	1	CARD8	opcode (X assigned)
	1	114	GLX opcode
	2	3	request length
	4	GLX_CONTEXT_TAG	context tag
	4	ENUM	pname
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	m	reply length, $m = (n==1 ? 0 : n*2)$
	4		unused
	4	CARD32	n
	if (n=1)	) this follows:	
	8	FLOAT64	params
	8	1 20111 0 1	unused
	otherw	ise this follows:	
	16		unused
	n*8	LISTOFFLOAT64	params
	-		F · · · · ·

Note that n may be zero, indicating that a GL error occured.

## GetError

	1 1 2 4	CARD8 115 2 GLX_CONTEXT_TAG	opcode (X assigned) GLX opcode request length context tag
$\Rightarrow$		4	D 1
	1	I	Reply
	1		unused
	2	CARD16	sequence number
	4	0	reply length
	4	ENUM	error
	20		unused

## GetFloatv

	1 1 2 4 4	CARD8 116 3 GLX_CONTEXT_TAG ENUM	opcode (X assigned) GLX opcode request length context tag pname
⇒	1 1 2 4 4 4	1 CARD16 m CARD32	Reply unused sequence number reply length, m = (n==1 ? 0 : n) unused n
	if (n=1	) this follows:	
	4 12	FLOAT32	params unused
	otherw	ise this follows:	
	16 n*4	LISTofFLOAT32	unused params

Note that n may be zero, indicating that a GL error occured.

## GetHistogramParameterfy

```
1
                                  opcode (X assigned)
      CARD8
1
      155
                                  GLX opcode
2
      4
                                 request length
4
      GLX_CONTEXT_TAG
                                  context tag
4
      ENUM
                                  target
4
                                  pname
      ENUM
1
      1
                                  Reply
1
                                  unused
2
      CARD16
                                  sequence number
4
                                  reply length, m = (n==1 ? 0 : n)
      m
4
                                  unused
      CARD32
                                  n
if (n=1) this follows:
      FLOAT32
                                  params
```

12 unused

otherwise this follows:

 $\begin{array}{ccc} 16 & & unused \\ n*4 & \texttt{LISTOFFLOAT32} & params \end{array}$ 

Note that n may be zero, indicating that a GL error occurred.

## GetHistogramParameteriv

⇒	1 1 2 4 4 4	CARD8 156 4 GLX_CONTEXT_TAG ENUM ENUM	opcode (X assigned) GLX opcode request length context tag target pname
,	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	m	reply length, $m = (n==1 ? 0 : n)$
	4		unused
	4	CARD32	n
	if (n=1	) this follows:	
	4	INT32	params
	12		unused
	otherw	rise this follows:	
	16		unused
	n*4	LISTofINT32	params
			-

Note that n may be zero, indicating that a GL error occurred.

## GetIntegerv

1	CARD8	opcode (X assigned)
1	117	GLX opcode
2	3	request length

	4	GLX_CONTEXT_TAG	context tag
	4	ENUM	pname
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	m	reply length, $m = (n==1 ? 0 : n)$
	4		unused
	4	CARD32	n
	if (n=1	1) this follows:	
	4	INT32	params
	12		unused
	otherv	vise this follows:	
	16		unused
	n*4	LISTofINT32	params

Note that n may be zero, indicating that a GL error occured.

# GetLightfv

	1	CARD8	opcode (X assigned)
	1	118	GLX opcode
	2	4	request length
	4	GLX_CONTEXT_TAG	context tag
	4	ENUM	light
	4	ENUM	pname
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	m	reply length, $m = (n==1 ? 0 : n)$
	4		unused
	4	CARD32	n
	if (n=1)	) this follows:	
	4	FLOAT32	params
	12		unused
	otherw	ise this follows:	

Version 1.3 - 2 June 1999

```
\begin{array}{ccc} 16 & & unused \\ n*4 & \texttt{LISTofFLOAT32} & params \end{array}
```

Note that n may be zero, indicating that a GL error occurred.

## GetLightiv

	1 1 2 4 4 4	CARD8 119 4 GLX_CONTEXT_TAG ENUM ENUM	opcode (X assigned) GLX opcode request length context tag light pname
$\Rightarrow$	1 1 2 4 4	1 CARD16 m	Reply unused sequence number reply length, $m = (n==1 ? 0 : n)$ unused
	4 if (n=1)	CARD32 ) this follows:	n
	4 12	INT32	params unused
	otherwi	ise this follows:	
	16 n*4	LISTOFINT32	unused params

Note that n may be zero, indicating that a GL error occurred.

# GetMapdv

1	CARD8	opcode (X assigned)
1	120	GLX opcode
2	4	request length
4	GLX_CONTEXT_TAG	context tag
4	ENUM	target
4	ENUM	query

Version 1.3 - 2 June 1999

$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	m	reply length, $m = (n==1 ? 0 : n*2)$
	4		unused
	4	CARD32	n
	if (n=1 8	) this follows:	v
	8		unused
	otherw	ise this follows:	
	16		unused
	n*8	LISTOFFLOAT64	v

Note that n may be zero, indicating that a GL error occurred.

# GetMapfv

	1	CARD8	opcode (X assigned)
	1	121	GLX opcode
	2	4	request length
	4	GLX_CONTEXT_TAG	context tag
	4	ENUM	target
	4	ENUM	query
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	m	reply length, $m = (n==1 ? 0 : n)$
	4		unused
	4	CARD32	n
	if (n=1)	) this follows:	
	4	FLOAT32	V
	12		unused
	otherwi	ise this follows:	
			_
	16		unused

n\*4 LISTOFFLOAT32 v

Note that n may be zero, indicating that a GL error occurred.

# GetMapiv

1	[ (	CARD8	opcode (X assigned)
1	[	122	GLX opcode
2	2	4	request length
4	1 (	GLX_CONTEXT_TAG	context tag
4	1	ENUM	target
4	1	ENUM	query
$\Rightarrow$			
1	1 :	1	Reply

1 2 4 4 4	CARD16 m CARD32	unused sequence number reply length, m = (n==1 ? 0 : n) unused n	
if (n=	1) this follows:		
4 12	FLOAT32	params unused	
otherwise this follows:			
16 n*4	LISTofFLOAT32	unused params	

Note that n may be zero, indicating that a GL error occurred.

# GetMaterialiv

	1	CARD8	opcode (X assigned)
	1	124	GLX opcode
	2	4	request length
	4	GLX_CONTEXT_TAG	context tag
	4	ENUM	face
	4	ENUM	pname
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	m	reply length, $m = (n==1 ? 0 : n)$
	4		unused
	4	CARD32	n
	if (n=1) this follows:		
	4	INT32	params
	12		unused
	otherwise this follows:		
	16		unused
	n*4	LISTofINT32	params

Note that n may be zero, indicating that a GL error occurred.

#### **GetMinmaxParameterfy**

```
1
                                   opcode (X assigned)
      CARD8
1
       158
                                   GLX opcode
2
                                   request length
      4
4
                                   context tag
      GLX_CONTEXT_TAG
4
      ENUM
                                   target
4
      ENUM
                                   pname
       1
1
                                   Reply
1
                                   unused
2
      CARD16
                                   sequence number
                                   reply length, m = (n==1 ? 0 : n)
4
      m
4
                                   unused
4
       CARD32
                                   n
if (n=1) this follows:
4
      FLOAT32
                                   params
12
                                   unused
otherwise this follows:
16
                                   unused
n*4
      LISTofFLOAT32
                                   params
```

Note that n may be zero, indicating that a GL error occurred.

#### **GetMinmaxParameteriv**

	1	CARD8	opcode (X assigned)
	1	159	GLX opcode
	2	4	request length
	4	GLX_CONTEXT_TAG	context tag
	4	ENUM	target
	4	ENUM	pname
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number

Version 1.3 - 2 June 1999

if (n=1) this follows:

4 INT32 params 12 unused

otherwise this follows:

 $\begin{array}{ccc} 16 & & unused \\ n*4 & \texttt{LISTOFINT32} & & params \end{array}$ 

Note that n may be zero, indicating that a GL error occurred.

## GetPixelMapfv

n\*4

LISTofFLOAT32

1 CARD8 opcode (X assigned) 1 125 GLX opcode 2 3 request length 4 context tag GLX\_CONTEXT\_TAG 4 ENUM map 1 1 Reply unused 1 sequence number 2 CARD16 reply length, m = (n==1 ? 0 : n)4 4 unused CARD32 n if (n=1) this follows: 4 FLOAT32 values 12 unused otherwise this follows: 16 unused

Note that n may be zero, indicating that a GL error occurred.

values

## GetPixelMapuiv

```
1
                                    opcode (X assigned)
       CARD8
1
       126
                                    GLX opcode
2
       3
                                    request length
4
       GLX_CONTEXT_TAG
                                    context tag
4
       ENUM
                                    map
                                    Reply
1
       1
                                   unused
1
2
                                    sequence number
       CARD16
4
       m
                                    reply length, m = (n==1 ? 0 : n)
4
                                    unused
       CARD32
                                   n
if (n=1) this follows:
4
       CARD32
                                    values
12
                                    unused
otherwise this follows:
16
                                    unused
n*4
                                    values
       LISTofCARD32
```

Note that n may be zero, indicating that a GL error occurred.

#### GetPixelMapusv

if (n=1) this follows:

```
1
                                   opcode (X assigned)
       CARD8
1
       127
                                   GLX opcode
                                   request length
2
       3
4
      GLX_CONTEXT_TAG
                                   context tag
4
      ENUM
                                   map
1
       1
                                   Reply
                                   unused
1
2
      CARD16
                                   sequence number
4
                                   reply length, m = (n==1 ? 0 : (n*2+p)/4)
      m
4
                                   unused
4
       CARD32
                                   n
```

2	CARD16	values
14		unused

otherwise this follows:

```
\begin{array}{ccc} 16 & & unused \\ n*2 & \texttt{LISTofCARD16} & & values \end{array}
```

unused, p=pad(n\*2)

Note that n may be zero, indicating that a GL error occurred.

# GetString

	1	CARD8	opcode (X assigned)
	1	129	GLX opcode
	2	3	request length
	4	GLX_CONTEXT_TAG	context tag
	4	ENUM	name
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	(n+p)/4	reply length
	4		unused
	4	CARD32	n
	16		unused
	n	STRING8	string
	p		unused, $p = pad(n)$

# GetTexEnvfv

	1	CARD8	opcode (X assigned)
	1	130	GLX opcode
	2	4	request length
	4	GLX_CONTEXT_TAG	context tag
	4	ENUM	target
	4	ENUM	pname
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number

Version 1.3 - 2 June 1999

4 m reply length, m = (n==1 ? 0 : n)
4 unused
4 CARD32 n

if (n=1) this follows:

4 FLOAT32 params
12 unused

otherwise this follows:

n\*4 LISTofFLOAT32 params

Note that n may be zero, indicating that a  $\ensuremath{\mathsf{GL}}$  error occurred.

#### **GetTexEnviv**

	1	CARD8	opcode (X assigned)
	1	131	GLX opcode
	2	4	request length
	4	GLX_CONTEXT_TAG	context tag
	4	ENUM	target
	4	ENUM	pname
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	m	reply length, $m = (n==1 ? 0 : n)$
	4		unused
	4	CARD32	n
	if (n=1	) this follows:	
	4	INT32	params
	12		unused
	otherw	ise this follows:	
	16		unused
	n*4	LISTofINT32	params
			-

Note that n may be zero, indicating that a GL error occurred.

#### GetTexGendv

	1	CARD8	opcode (X assigned)
	1	132	GLX opcode
	2	4	request length
	4	GLX_CONTEXT_TAG	context tag
	4	ENUM	coord
	4	ENUM	pname
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	m	reply length, $m = (n==1 ? 0 : n*2)$
	4		unused
	4	CARD32	n
	if (n=1	) this follows:	
	8	FLOAT64	params
	8		unused
	otherw	ise this follows:	
	16		unused
	n*8	LISTOFFLOAT64	params

Note that n may be zero, indicating that a GL error occurred.

### GetTexGenfv

```
1
                                  opcode (X assigned)
      CARD8
                                  GLX opcode
1
      133
2
                                  request length
4
                                  context tag
      GLX_CONTEXT_TAG
4
      ENUM
                                  coord
4
      ENUM
                                  pname
                                  Reply
1
1
                                  unused
2
                                  sequence number
      CARD16
4
      m
                                  reply length, m = (n==1 ? 0 : n)
4
                                  unused
4
      CARD32
                                  n
```

Version 1.3 - 2 June 1999

# if (n=1) this follows:

4 FLOAT32 params 12 unused

otherwise this follows:

 $\begin{array}{ccc} 16 & & unused \\ n*4 & \texttt{LISTOFFLOAT32} & params \end{array}$ 

Note that n may be zero, indicating that a GL error occurred.

#### GetTexGeniv

	1	CARD8	opcode (X assigned)
	1	134	GLX opcode
	2	4	request length
	4	GLX_CONTEXT_TAG	context tag
	4	ENUM	coord
	4	ENUM	pname
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	m	reply length, $m = (n==1 ? 0 : n)$
	4		unused
	4	CARD32	n
	if (n=1)	) this follows:	
	4	INT32	params
	12		unused
	otherw	ise this follows:	
	16		unused
	n*4	LISTofINT32	params

Note that n may be zero, indicating that a GL error occurred.

#### GetTexLevelParameterfv

_	1 1 2 4 4 4 4	CARD8 138 5 GLX_CONTEXT_TAG ENUM INT32 ENUM	opcode (X assigned) GLX opcode request length context tag target level pname
$\Rightarrow$	1 1 2 4 4 4	1 CARD16 m CARD32	Reply unused sequence number reply length, m = (n==1 ? 0 : n) unused n
	if (n=1)	this follows:	
	4 12	FLOAT32	params unused
	otherwi	ise this follows:	
	16 n*4	LISTofFLOAT32	unused params

Note that n may be zero, indicating that a GL error occurred.

#### GetTexLevelParameteriv

```
1
                                 opcode (X assigned)
      CARD8
1
      139
                                 GLX opcode
2
      5
                                 request length
4
      GLX_CONTEXT_TAG
                                 context tag
4
                                 target
      ENUM
                                 level
4
      INT32
4
      ENUM
                                 pname
1
                                 Reply
      1
                                 unused
1
2
                                 sequence number
      CARD16
4
                                 reply length, m = (n==1 ? 0 : n)
      m
4
                                 unused
4
      CARD32
                                 n
```

if (n=1) this follows:

4 INT32 params 12 unused

otherwise this follows:

 $\begin{array}{ccc} 16 & & unused \\ n*4 & \texttt{LISTofINT32} & params \end{array}$ 

Note that n may be zero, indicating that a GL error occurred.

#### GetTexParameterfy

⇒	1 1 2 4 4 4	CARD8 136 4 GLX_CONTEXT_TAG ENUM ENUM	opcode (X assigned) GLX opcode request length context tag target pname
,	1 1	1	Reply unused
	2 4	CARD16 m	sequence number reply length, m = (n==1 ? 0 : n)
	4 4	CARD32	unused n
	if (n=1	) this follows:	
	4 12	FLOAT32	params unused
	otherw	ise this follows:	

Note that n may be zero, indicating that a GL error occurred.

LISTofFLOAT32

#### GetTexParameteriv

16

n\*4

1 CARD8 opcode (X assigned)

Version 1.3 - 2 June 1999

unused

params

$\Rightarrow$	1 2 4 4 4	137 4 GLX_CONTEXT_TAG ENUM ENUM	GLX opcode request length context tag target pname
~	1 1 2 4 4 4	1 CARD16 m CARD32	Reply unused sequence number reply length, $m = (n==1 ? 0 : n)$ unused $n$
	if (n=1)	) this follows:	
	4 12	INT32	params unused
	otherwi	ise this follows:	
	16 n*4	LISTOFINT32	unused params

Note that n may be zero, indicating that a GL error occurred.

### **IsList**

	1 1 2 4	CARD8 141 3 GLX_CONTEXT_TAG	opcode (X assigned) GLX opcode request length context tag
	4	CARD32	list
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	0	reply length
	4	BOOL32	return value
	20		unused

# IsTexture

1 CARD8 opcode (X assigned)

	1	146	GLX opcode
	2	3	request length
	4	GLX_CONTEXT_TAG	context tag
	4	CARD32	texture
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	0	reply length
	4	BOOL32	return value
	20		unused

### NewList

1	CARD8	opcode (X assigned)
1	101	GLX opcode
2	4	request length
4	GLX_CONTEXT_TAG	context tag
4	CARD32	list
4	ENUM	mode

# **PixelStoref**

1	CARD8	opcode (X assigned)
1	109	GLX opcode
2	4	request length
4	GLX_CONTEXT_TAG	context tag
4	ENUM	pname
4	FLOAT32	param

# PixelStorei

1	CARD8	opcode (X assigned)
1	110	GLX opcode
2	4	request length
4	GLX_CONTEXT_TAG	context tag
4	ENUM	pname
4	INT32	param

#### RenderMode

1 1 2	CARD8 107 3	opcode (X assigned) GLX opcode request length
4	GLX_CONTEXT_TAG	context tag
4	ENUM	mode
	0x1C00	GL_RENDER
	0x1C01	GL_FEEDBACK
	0x1C02	GL_SELECT

 $\Rightarrow$ 

If the calling thread was previously in feedback mode, the reply is:

1	1	Reply
1		unused
2	CARD16	sequence number
4	n	reply length
4	INT32	return value
4	CARD32	n
4	ENUM	new_mode
12		unused
n*4	LISTofFLOAT32	feedback data

If the calling thread was previously in selection mode, the reply is:

1	1	Reply
1		unused
2	CARD16	sequence number
4	n	reply length
4	INT32	return value
4	CARD32	n
4	ENUM	new_mode
12		unused
n*4	LISTofCARD32	selection data

If the calling thread was previously in rendering mode, there is no reply.

Note that n may be zero, indicating that a GL error occured.

#### SelectBuffer

1	CARD8	opcode (X assigned)
1	106	GLX opcode

2	3	request length
4	GLX_CONTEXT_TAG	context tag
4	INT32	size

Selection data is returned in the reply of the next **RenderMode** request.

# 2.2.2 GL Non-rendering Commands That Return Pixel Data

These commands return images of pixel data; for more details about the encoding of pixel images, see Appendix A.

The valid values for the *format* and *type* parameters of these commands are listed in the "Encoding" column of Table A.1 and Table A.2 in Appendix A. If *format* or *type* is not valid, then the command is erroneous. No extra padding is needed after pixel data, because the image format already pads to 32 bits.

#### GetColorTable

1	CARD8	opcode (X assigned)
1	147	GLX opcode
2	6	request length
4	GLX_CONTEXT_TAG	context tag
4	ENUM	target
4	ENUM	format
4	ENUM	type
1	BOOL	swap_bytes
3		unused

 $\Rightarrow$ 

If the command succeeds, the table is sent in the reply:

1	1	reply
1		unused
2	CARD16	sequence number
4	n	reply length
8		unused
4	INT32	width
12		unused
4*n	LISTofBYTE	table

Note that n may be zero, indicating that a GL error occured.

The structure of *table* is described in more detail in Appendix A, using the parameters *swap\_bytes*, *format*, and *type* as given in the request, *width* and *height* as given in the reply, and height = 1.

#### GetConvolutionFilter

```
1
                                  opcode (X assigned)
      CARD8
1
      150
                                  GLX opcode
2
                                  request length
4
      GLX_CONTEXT_TAG
                                  context tag
4
                                  target
      ENUM
4
      ENUM
                                  format
4
      ENUM
                                  type
1
                                  swap_bytes
      BOOL
3
                                  unused
```

 $\Rightarrow$ 

If the command succeeds, the table is sent in the reply:

1	1	reply
1		unused
2	CARD16	sequence number
4	n	reply length
8		unused
4	INT32	width
4	INT32	height
8		unused
4*n	LISTOfBYTE	pixels

Note that n may be zero, indicating that a GL error occured.

The structure of *pixels* is described in more detail in Appendix A, using the parameters *swap\_bytes*, *format*, and *type* as given in the request, and *width* and *height* as given in the reply.

### GetHistogram

1	CARD8	opcode (X assigned)
1	154	GLX opcode
2	6	request length
4	GLX_CONTEXT_TAG	context tag
4	ENUM	target
4	ENUM	format
4	ENUM	type
1	BOOL	swap_bytes

Version 1.3 - 2 June 1999

```
1 BOOL reset unused
```

 $\Rightarrow$ 

If the command succeeds, the table is sent in the reply:

1	1	reply
1		unused
2	CARD16	sequence number
4	n	reply length
8		unused
4	INT32	width
12		unused
4*n	LISTofBYTE	pixels

Note that n may be zero, indicating that a GL error occured.

The structure of *pixels* is described in more detail in Appendix A, using the parameters *swap\_bytes*, *format*, and *type* as given in the request, *width* and *height* as given in the reply, and height = 1.

#### **GetMinmax**

1	CARD8	opcode (X assigned)
1	157	GLX opcode
2	6	request length
4	GLX_CONTEXT_TAG	context tag
4	ENUM	target
4	ENUM	format
4	ENUM	type
1	BOOL	swap_bytes
1	BOOL	reset
2		unused

 $\Rightarrow$ 

If the command succeeds, the table is sent in the reply:

1	1	reply
1		unused
2	CARD16	sequence number
4	n	reply length
24		unused
4*n	LISTofBYTE	pixels

Note that n may be zero, indicating that a GL error occured.

The structure of *pixels* is described in more detail in Appendix A, using the parameters  $swap\_bytes$ , format, and type as given in the request, width = 2, and height = 1.

#### GetPolygonStipple

```
1
                                   opcode (X assigned)
       CARD8
1
       128
                                   GLX opcode
2
       3
                                   request length
4
       GLX_CONTEXT_TAG
                                   context tag
1
                                   lsb\_first
3
                                   unused
```

 $\Rightarrow$ 

If the command succeeds, the stipple is sent in the reply:

1	1	Reply
1		unused
2	CARD16	sequence number
4	32	reply length
24		unused
128	LISTofBYTE	stipple

Otherwise an empty reply is sent, indicating that a GL error occurred:

1	1	Reply
1		unused
2	CARD16	sequence number
4	0	reply length
24		unused

The structure of *stipple* is described in more detail in Appendix A, using the parameter *lsb\_first* as given in the request, and *format*=GL\_COLOR\_INDEX, *type*=GL\_BITMAP, *width*=32, and *height*=32.

#### GetSeparableFilter

1	CARD8	opcode (X assigned)
1	153	GLX opcode
2	6	request length
4	GLX_CONTEXT_TAG	context tag
4	ENUM	target
4	ENUM	format
4	ENUM	type
1	BOOL	swap_bytes
3		unused

Version 1.3 - 2 June 1999

 $\Rightarrow$ 

If the command succeeds, the filters are sent in the reply:

1	1	reply
1		unused
2	CARD16	sequence number
4	n	reply length
8		unused
4	INT32	row_width
4	INT32	col_height
8		unused
4*n	LISTofBYTE	row followed by column

Note that n may be zero, indicating that a GL error occured.

The structure of *row* and *column* are described in more detail in Appendix A, using the parameters *swap\_bytes*, *format*, and *type* as given in the request, and *row\_width* and *col\_height* as given in the reply. For *row*, the image has  $width = row_width$  and height = 1; for *column*, the image has width = 1 and  $height = col_height$ .

### **GetTexImage**

	1	CARD8	opcode (X assigned)
	1	135	GLX opcode
	2	7	request length
	4	GLX_CONTEXT_TAG	context tag
	4	ENUM	target
	4	INT32	level
	4	ENUM	format
	4	ENUM	type
	1	BOOL	swap_bytes
	3		unused
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	n	reply length
	8		unused
	4	INT32	width
	4	INT32	height
	4	INT32	depth
	4		unused
	4*n	LISTofBYTE	teximage

Note that n may be zero, indicating that a GL error occured.

The structure of *teximage* is described in more detail in Appendix A, using the parameters *swap\_bytes*, *format*, and *type* as given in the request, and *width*, *height*, and *depth* as given in the reply.

### ReadPixels

	1	CARD8	opcode (X assigned)
	1	111	GLX opcode
	2	9	request length
	4	GLX_CONTEXT_TAG	context tag
	4	INT32	X
	4	INT32	y
	4	INT32	width
	4	INT32	height
	4	ENUM	format
	4	ENUM	type
	1	BOOL	swap_bytes
	1	BOOL	lsb_first
	2		unused
$\Rightarrow$			
	1	1	Reply
	1		unused
	2	CARD16	sequence number
	4	n	reply length
	24		unused
	4*n	LISTofBYTE	pixels

Note that n may be zero, indicating that a GL error occured.

If width < 0 or height < 0, the command is erroneous. The structure of *pixels* is described in more detail in Appendix A, using the parameters *width*, *height*, *format*, *type*, *swap\_bytes*, and *lsb\_first* as given in the request.

# 2.3 Requests for GL Rendering Commands

There are two requests used to send GL rendering commands. The **glXRender** request is used to send multiple, relatively small commands in a single request, and **glXRenderLarge** is used to send a single large command, split into multiple requests.

### 2.3.1 Send Multiple GL Rendering Commands

Name: glXRender

#### **Request:**

tag: glx\_context\_tag

commands: LISTofGLX\_RENDER\_COMMAND

Where a GLX\_RENDER\_COMMAND may be any of the GL rendering commands described in Section 2.3.3, "GL Rendering Commands". The general format of a GLX\_RENDER\_-COMMAND is:

Each render\_command\_opcode specifies the rendering command. Each render\_command\_length specifies the length of the GLX\_RENDER\_COMMAND in bytes, including the length and opcode fields. Each rendering command is padded to a multiple of 4 bytes.

Errors: BadLength, GLXBadRenderRequest, GLXBadContextTag

#### **Description:**

This request is used to send one or more GL rendering commands; a **glXRender** request will typically contain multiple rendering commands.

GLXBadRenderRequest is generated if any  $render\_command\_opcode$  is invalid. BadLength is generated if the sum of all the  $render\_command\_length$  fields does not match the length field given in the request header. GLXBadContextTag is generated if tag is not a valid context tag.

```
1 CARD8 opcode (X assigned)
1 1 GLX opcode
2 2+n request length
4 GLX_CONTEXT_TAG context tag
4*n LISTOFGLX_RENDER_COMMANDommands
```

A  $GLX_RENDER_COMMAND$  can be any of the commands described in Section 2.3.3, and has the general format:

```
2
                                       rendering command length
       4+m+p
2
                                       rendering command opcode
       CARD16
                                       1^{st} parameter
s_1
       type_1
                                       2^{nd} parameter
       type_2
s_2
                                       N^{th} parameter
       type_N
s_N
                                       unused, p=pad(m)
Where m = s_1 + s_2 + ... + s_N.
```

# 2.3.2 Send a Large GL Rendering Command

Some GL rendering commands may be so large that they cannot fit into a single glXRender request, which is limited by the maximum size of an X request: CallLists, DrawArrays, Map1d, Map2d, Map1f, Map2f, PixelMapfv, PixelMapuiv, PixMapusv, Bitmap, PolygonStipple, PrioritizeTextures, TexSubImagexD, TexImagexD, ColorTable, ColorSubTable, ConvolutionFilterxD, SeparableFilter2D, and Draw-Pixels. These commands contain a number of small parameters followed by one potentially large parameter; if the parameter is so large that the command cannot fit into a glXRender request, the command is sent in a series of glXRenderLarge requests instead.

Name: glXRenderLarge

#### Request:

The first **glXRenderLarge** request contains the small parameters:

```
tag: GLX_CONTEXT_TAG
request_number: CARD16
request_total: CARD16
n<sub>0</sub>: CARD32
render_command_length: CARD32
render_command_opcode: CARD32
param1: type1
param2: type2
...
...
paramN: typeN
```

The large parameter is split into P pieces, which are sent in P subsequent requests; each  $i^{th}$  request,  $1 \le i \le P$ , is:

tag: GLX\_CONTEXT\_TAG
request\_number: CARD16
request\_total: CARD16

 $n_i$ : CARD32

 $i^{th}$  piece: LISTofBYTE

Errors: BadLength, BadAlloc, GLXBadLargeRequest,
GLXBadContextTag

#### **Description:**

As with the small encoding for rendering commands, <code>render\_command\_opcode</code> is an opcode identifying the rendering command, and <code>render\_command\_length</code> is the length of the command in bytes (the length consists of 8 bytes for the opcode and length fields, plus the length of the small parameters, plus the length of the large parameter). Note that, unlike the small encoding, <code>render\_command\_length</code> is a <code>CARD32</code> rather than a <code>CARD16</code>; this is to accommodate the larger total length.

The parameters  $request\_number$  and  $request\_total$  are used for error checking the **glXRenderLarge** requests in the series.  $Request\_number$  is the number of the request within the series, and  $request\_total$  is the total number of requests in the series. For example, if a series of 3 **glXRenderLarge** requests is needed to send the entire GL command, the first request should have  $request\_number$  set to 1 and  $request\_total$  set to 3, the second should have 2 and 3, and the third should have 3 and 3. The  $n_i$  parameter is the number of bytes in the request that are actually used as part of the rendering command; its purpose is to allow for pad bytes that might follow.

GLXBadLargeRequest is generated if any  $render\_command\_opcode$  is invalid, if the sum of the  $n_i$  fields of all the requests does not match  $render\_command\_length$ , if not enough requests are received, or if the  $request\_number$  or  $request\_total$  fields are not what is expected. BadAlloc is generated if the server cannot allocate enough resources to hold the large command. GLXBadContextTag is generated if tag is not a valid context tag.

A rendering command that can be large, i.e., those described in sections Sections 2.3.4 - 2.3.6, is one with N small, fixed-size parameters followed by 1 potentially large, variable-size parameter. It has an encoding in this general form when the command is small (packed in a **glXRender** request):

Version 1.3 - 2 June 1999

```
A type_A potentially large parameter p unused, p=pad(m) Where m = s_1 + s_2 + ... + s_N + A.
```

When the parameter is so large that the command cannot fit into a **glXRender** request, the command is sent in multiple **glXRenderLarge** requests. The first request contains the small parameters:

```
1
                                      opcode (X assigned)
       CARD8
1
                                      GLX opcode
       2
2
       6 + (n_0 + p_0)/4
                                      request length
4
       GLX_CONTEXT_TAG
                                      context tag
2
       CARD16
                                      request number (explained below)
2
                                      request total (explained below)
       CARD16
4
                                      n_0 = s_1 + s_2 + \dots + s_N
       CARD32
4
                                      rendering command length, L (see below)
       CARD32
4
                                      rendering command opcode
       CARD32
                                      1^{st} small parameter
       type_1
s_1
                                      2^{nd} small parameter
s_2
       type_2
                                      N^{th} small parameter
s_N
       type_N
                                      unused
p_0
```

Then P requests follow, where P is the number of pieces that the large parameter is split into; each subsequent  $i^{th}$  request  $(1 \le i \le P)$  contains a piece:

```
1
                                      opcode (X assigned)
       CARD8
1
                                      GLX opcode
2
       4 + (n_i + p_i)/4
                                      request length
4
       GLX_CONTEXT_TAG
                                      context tag
2
       CARD16
                                      request number
2
       CARD16
                                      request total
4
       CARD32
                                      i<sup>th</sup> piece of large parameter
n_i
       LISTOfBYTE
                                      unused
p_i
```

The total length of the large parameter is  $n_1+n_2+...+n_P$ . Hence, the total length of the rendering command is  $L=s_1+s_2+...+s_N+n_1+n_2+...+n_P$ .

# 2.3.3 GL Rendering Commands

This section describes the protocol formats for GL rendering commands. These formats were referred to as  ${\tt GLX\_RENDER\_COMMAND}$  in the preceding description of the  ${\tt glXRender}$  request. The header of a  ${\tt GLX\_RENDER\_COMMAND}$  contains a command length and a command opcode:

command\_length: CARD16
command\_opcode: CARD16

Followed by the parameters of the command.

The following section lists the parameters of each rendering command.

#### Accum

2	12	rendering command length
2	137	rendering command opcode
4	ENUM	op
4	FLOAT32	value

#### ActiveTextureARB

2	8	rendering command length
2	197	rendering command opcode
4	ENUM	texture

### AlphaFunc

2	12	rendering command length
2	159	rendering command opcode
4	ENUM	func
4	FLOAT32	ref

# Begin

2	8	rendering command length
2	4	rendering command opcode
4	ENUM	mode

# **BindTexture**

2	12	rendering command length
2	4117	rendering command opcode
4	ENUM	target
4	CARD32	texture

# BlendColor

2	20	rendering command length
2	4096	rendering command opcode
4	FLOAT32	red
4	FLOAT32	green
4	FLOAT32	blue
4	FLOAT32	alpha

# BlendEquation

2	8	rendering command length
2	4097	rendering command opcode
4	ENUM	mode

### BlendFunc

2	12	rendering command length
2	160	rendering command opcode
4	ENUM	sfactor
4	ENUM	dfactor

# CallList

2	8	rendering command length
2	1	rendering command opcode
4	CARD32	list

### Clear

2	8	rendering command length
2	127	rendering command opcode
4	BITFIELD	mask

# ClearAccum

2	20	rendering command length
2	128	rendering command opcode
4	FLOAT32	red
4	FLOAT32	green
4	FLOAT32	blue
4	FLOAT32	alpha

### ClearColor

2	20	rendering command length
2	130	rendering command opcode
4	FLOAT32	red
4	FLOAT32	green
4	FLOAT32	blue
4	FLOAT32	alpha

# ClearDepth

2	12	rendering command length
2	132	rendering command opcode
8	FLOAT64	depth

### ClearIndex

2	8	rendering command length
2	129	rendering command opcode
4	FLOAT32	c

# ClearStencil

2	8	rendering command length
2	131	rendering command opcode
4	TNT32	S

# ClipPlane

2	40	rendering command length
2	77	rendering command opcode
8	FLOAT64	equation[0]
8	FLOAT64	equation[1]
8	FLOAT64	equation[2]
8	FLOAT64	equation[3]
4	ENUM	plane

### Color3by

2	8	rendering command length
2	6	rendering command opcode
1	INT8	v[0]
1	INT8	v[1]
1	INT8	v[2]
1		unused

# Color3dv

2 2	28 7	rendering command length rendering command opcode
8	FLOAT64	v[0]
8	FLOAT64	v[1]
8	FLOAT64	v[2]

### Color3fv

2	16	rendering command length
2	8	rendering command opcode
4	FLOAT32	v[0]
4	FLOAT32	v[1]
4	FLOAT32	v[2]

# Color3iv

2	16	rendering command length
2	9	rendering command opcode
4	INT32	v[0]
4	INT32	v[1]
4	INT32	v[2]

### Color3sv

2	12	rendering command length
2	10	rendering command opcode
2	INT16	v[0]
2	INT16	v[1]
2	INT16	v[2]
2		unused

# Color3ubv

2	8	rendering command length
2	11	rendering command opcode
1	CARD8	v[0]
1	CARD8	v[1]
1	CARD8	v[2]
1		unused

### Color3uiv

2	16	rendering command length
2	12	rendering command opcode
4	CARD32	v[0]
4	CARD32	v[1]
4	CARD32	v[2]

# Color3usv

2 12 rendering command length

2	13	rendering command opcode
2	CARD16	v[0]
2	CARD16	v[1]
2	CARD16	v[2]
2		unused

### Color4by

2	8	rendering command length
2	14	rendering command opcode
1	INT8	v[0]
1	INT8	v[1]
1	INT8	v[2]
1	INT8	v[3]

### Color4dv

2	36	rendering command length
2	15	rendering command opcode
8	FLOAT64	v[0]
8	FLOAT64	v[1]
8	FLOAT64	v[2]
8	FLOAT64	v[3]

### Color4fv

2	20	rendering command length
2	16	rendering command opcode
4	FLOAT32	v[0]
4	FLOAT32	v[1]
4	FLOAT32	v[2]
4	FLOAT32	v[3]

# Color4iv

2	20	rendering command length
2	17	rendering command opcode
4	INT32	v[0]

4	INT32	v[1]
4	INT32	v[2]
4	INT32	v[3]

### Color4sv

2	12	rendering command length
2	18	rendering command opcode
2	INT16	v[0]
2	INT16	v[1]
2	INT16	v[2]
2	INT16	v[3]

# Color4ubv

2	8	rendering command length
2	19	rendering command opcode
1	CARD8	v[0]
1	CARD8	v[1]
1	CARD8	v[2]
1	CARD8	v[3]

# Color4uiv

2	20	rendering command length
2	20	rendering command opcode
4	CARD32	v[0]
4	CARD32	v[1]
4	CARD32	v[2]
4	CARD32	v[3]

# Color4usv

2	12	rendering command length
2	21	rendering command opcode
2	CARD16	v[0]
2	CARD16	v[1]
2	CARD16	v[2]

2 CARD16 v[3]

### ColorMask

2	8	rendering command length
2	134	rendering command opcode
1	BOOL	red
1	BOOL	green
1	BOOL	blue
1	BOOL	alpha

### ColorMaterial

2	12	rendering command length
2	78	rendering command opcode
4	ENUM	face
4	ENUM	mode

# ${\bf Color Table Parameter fv}$

2	12+4*n	rendering command length
2	2054	rendering command opcode
4	ENUM	target
4	ENUM	pname
	0x80D6 n=4	GL_COLOR_TABLE_SCALE
	0x80D7 n=4	GL_COLOR_TABLE_BIAS
	else n=0	command is erroneous
4*n	LISTofFLOAT32	params

#### ColorTableParameteriv

2	12+4*n	rendering command length
2	2055	rendering command opcode
4	ENUM	target
4	ENUM	pname
	0x80D6 n=4	GL_COLOR_TABLE_SCALE
	0x80D7 n=4	GL_COLOR_TABLE_BIAS
	else n=0	command is erroneous

4\*n LISTOFINT32 params

### ConvolutionParameterf

2	16	rendering command length
2	4103	rendering command opcode
4	ENUM	target
4	ENUM	pname
4	FLOAT32	params

#### ConvolutionParameterfy

2	12+4*n	rendering command length
2	4104	rendering command opcode
4	ENUM	target
4	ENUM	pname
	0x8013 n=1	GL_CONVOLUTION_BORDER_MODE
	0x8014 n=4	GL_CONVOLUTION_FILTER_SCALE
	0x8015 n=4	GL_CONVOLUTION_FILTER_BIAS
	0x8017 n=1	GL_CONVOLUTION_FORMAT
	0x8018 n=1	GL_CONVOLUTION_WIDTH
	0x8019 n=1	GL_CONVOLUTION_HEIGHT
	0x801A n=1	GL_MAX_CONVOLUTION_WIDTH
	0x801B n=1	GL_MAX_CONVOLUTION_HEIGHT
	else n=0	command is erroneous
4*n	LISTofFLOAT32	params

### ConvolutionParameteri

2	16	rendering command length
2	4105	rendering command opcode
4	ENUM	target
4	ENUM	pname
4	INT32	params

### ConvolutionParameteriv

2 12+4\*n rendering command length

2	4106	rendering command opcode
4	ENUM	target
4	ENUM	pname
	0x8013 n=1	GL_CONVOLUTION_BORDER_MODE
	0x8014 n=4	GL_CONVOLUTION_FILTER_SCALE
	0x8015 n=4	GL_CONVOLUTION_FILTER_BIAS
	0x8017 n=1	GL_CONVOLUTION_FORMAT
	0x8018 n=1	GL_CONVOLUTION_WIDTH
	0x8019 n=1	GL_CONVOLUTION_HEIGHT
	0x801A n=1	GL_MAX_CONVOLUTION_WIDTH
	0x801B n=1	GL_MAX_CONVOLUTION_HEIGHT
	else n=0	command is erroneous
4*n	LISTofINT32	params

# Copy Color Sub Table

2	24	rendering command length
2	196	rendering command opcode
4	ENUM	target
4	INT32	start
4	INT32	X
4	INT32	У
4	INT32	width

# CopyColorTable

2	24	rendering command length
2	2056	rendering command opcode
4	ENUM	target
4	ENUM	internalformat
4	INT32	X
4	INT32	y
4	INT32	width

# Copy Convolution Filter 1D

2	24	rendering command length
2	4107	rendering command opcode
4	ENUM	target
4	ENUM	internalformat

4	INT32	X
4	INT32	у
4	INT32	width

# Copy Convolution Filter 2D

2	28	rendering command length
2	4108	rendering command opcode
4	ENUM	target
4	ENUM	internalformat
4	INT32	X
4	INT32	y
4	INT32	width
4	INT32	height

# CopyPixels

2	24	rendering command length
2	172	rendering command opcode
4	INT32	X
4	INT32	у
4	INT32	width
4	INT32	height
4	ENUM	type

# CopyTexImage2D

2	36	rendering command length
2	4120	rendering command opcode
4	ENUM	target
4	INT32	level
4	ENUM	internalformat
4	INT32	X
4	INT32	y
4	INT32	width
4	INT32	height
4	INT32	border

# CopyTexSubImage1D

2	28	rendering command length
2	4121	rendering command opcode
4	ENUM	target
4	INT32	level
4	INT32	xoffset
4	INT32	X
4	INT32	У
4	INT32	width

# CopyTexSubImage2D

2	36	rendering command length
2	4122	rendering command opcode
4	ENUM	target
4	INT32	level
4	INT32	xoffset
4	INT32	yoffset
4	INT32	X
4	INT32	y
4	INT32	width
4	INT32	height

# CopyTexSubImage3D

2	40	rendering command length
2	4123	rendering command opcode
4	ENUM	target
4	INT32	level
4	INT32	xoffset
4	INT32	yoffset
4	INT32	zoffset
4	INT32	X
4	INT32	y
4	INT32	width
4	INT32	height

# CullFace

2	8	rendering command length
2	79	rendering command opcode

4 ENUM mode

# DepthFunc

2	8	rendering command length
2	164	rendering command opcode
4	ENUM	func

# DepthMask

2	8	rendering command length
2	135	rendering command opcode
1	BOOL	flag
3		unused

# DepthRange

2	20	rendering command length
2	174	rendering command opcode
8	FLOAT64	zNear
8	FLOAT64	zFar

# DrawBuffer

2	8	rendering command length
2	126	rendering command opcode
4	ENUM	mode

# EdgeFlagv

2	8	rendering command length
2	22	rendering command opcode
1	BOOL	flag[0]
3		unused

# End

2	4	rendering command length
2	23	rendering command opcode

# EvalCoord1dv

2	12	rendering command length
2	151	rendering command opcode
8	FLOAT64	u[0]

### Eval Coord 1 fv

2	8	rendering command length
2	152	rendering command opcode
4	FLOAT32	u[0]

### EvalCoord2dv

2	20	rendering command length
2	153	rendering command opcode
8	FLOAT64	u[0]
8	FLOAT64	u[1]

### EvalCoord2fv

2	12	rendering command length
2	154	rendering command opcode
4	FLOAT32	u[0]
4	FLOAT32	u[1]

# EvalMesh1

2	16	rendering command length
2	155	rendering command opcode

4	ENUM	mode
4	INT32	i1
4	INT32	i2

### EvalMesh2

2	24	rendering command length
2	157	rendering command opcode
4	ENUM	mode
4	INT32	i1
4	INT32	i2
4	INT32	j1
4	INT32	j2

# EvalPoint1

2	8	rendering command length
2	156	rendering command opcode
4	INT32	i

# EvalPoint2

2	12	rendering command length
2	158	rendering command opcode
4	INT32	i
4	INT32	i

# Fogf

2	12	rendering command length
2	80	rendering command opcode
4	ENUM	pname
4	FLOAT32	param

# Fogfv

2 2	8+4*n 81	rendering command length rendering command opcode
4	ENUM	pname
	0x0B61 n=1	GL_FOG_INDEX
	0x0B62 n=1	GL_FOG_DENSITY
	0x0B63 n=1	GL_FOG_START
	0x0B64 n=1	GL_FOG_END
	0x0B65 n=1	GL_FOG_MODE
	0x0B66 n=4	GL_FOG_COLOR
	else n=0	command is erroneous
4*n	LISTofFLOAT32	params

# Fogi

2	12	rendering command length
2	82	rendering command opcode
4	ENUM	pname
4	TNT32	param

# Fogiv

2	8+4*n	rendering command length
2	83	rendering command opcode
4	ENUM	pname
	0x0B61 n=1	GL_FOG_INDEX
	0x0B62 n=1	GL_FOG_DENSITY
	0x0B63 n=1	GL_FOG_START
	0x0B64 n=1	GL_FOG_END
	0x0B65 n=1	GL_FOG_MODE
	0x0B66 n=4	GL_FOG_COLOR
	else n=0	command is erroneous
4*n	LISTofINT32	params

### **FrontFace**

2	8	rendering command length
2	84	rendering command opcode
4	ENUM	mode

# Frustum

2	52	rendering command length
2	175	rendering command opcode
8	FLOAT64	left
8	FLOAT64	right
8	FLOAT64	bottom
8	FLOAT64	top
8	FLOAT64	zNear
8	FLOAT64	zFar

# Hint

2	12	rendering command length
2	85	rendering command opcode
4	ENUM	target
4	ENUM	mode

# Histogram

2	20	rendering command length
2	4110	rendering command opcode
4	ENUM	target
4	INT32	width
4	ENUM	internalformat
1	BOOL	sink
3		unused

### Indexdv

2	12	rendering command length
2	24	rendering command opcode
8	FLOAT64	c[0]

# Indexfv

2 8 rendering command length

2	25	rendering command opcode
4	FLOAT32	c[0]

## Indexiv

2	8	rendering command length
2	26	rendering command opcode
4	INT32	c[0]

#### IndexMask

2	8	rendering command length
2	136	rendering command opcode
4	CARD32	mask

#### Indexsv

2	8	rendering command length
2	27	rendering command opcode
2	INT16	c[0]
2		unused

#### Indexubv

2	8	rendering command length
2	194	rendering command opcode
1	CARD8	c[0]
3		unused

#### **InitNames**

2	4	rendering command length
2	121	rendering command opcode

## Lightf

2	16	rendering command length
2	86	rendering command opcode
4	ENUM	light
4	ENUM	pname
4	FLOAT32	param

## Lightfv

2	12+4*n	rendering command length
2	87	rendering command opcode
4	ENUM	light
4	ENUM	pname
	0x1200 n=4	GL_AMBIENT
	0x1201 n=4	GL_DIFFUSE
	0x1202 n=4	GL_SPECULAR
	0x1203 n=4	GL_POSITION
	0x1204 n=3	GL_SPOT_DIRECTION
	0x1205 n=1	GL_SPOT_EXPONENT
	0x1206 n=1	GL_SPOT_CUTOFF
	0x1207 n=1	GL_CONSTANT_ATTENUATION
	0x1208 n=1	GL_LINEAR_ATTENUATION
	0x1209 n=1	GL_QUADRATIC_ATTENUATION
	else n=0	command is erroneous
4*n	LISTofFLOAT32	params

# Lighti

2	16	rendering command length
2	88	rendering command opcode
4	ENUM	light
4	ENUM	pname
4	INT32	param

# Lightiv

2	12+4*n	rendering command length
2	89	rendering command opcode
4	ENUM	light
4	ENUM	pname
	0x1200 n=4	GL_AMBIENT

Version 1.3 - 2 June 1999

	0x1201 n=4	GL_DIFFUSE
	0x1202 n=4	GL_SPECULAR
	0x1203 n=4	GL_POSITION
	0x1204 n=3	GL_SPOT_DIRECTION
	0x1205 n=1	GL_SPOT_EXPONENT
	0x1206 n=1	GL_SPOT_CUTOFF
	0x1207 n=1	GL_CONSTANT_ATTENUATION
	0x1208 n=1	GL_LINEAR_ATTENUATION
	0x1209 n=1	GL_QUADRATIC_ATTENUATION
	else n=0	command is erroneous
4*n	LISTofINT32	params

## LightModelf

2	12	rendering command length
2	90	rendering command opcode
4	ENUM	pname
4	FLOAT32	param

## LightModelfv

2 2	8+4*n 91	rendering command length rendering command opcode
4	ENUM	pname
	0x81F8 n=1	GL_LIGHT_MODEL_COLOR_CONTROL
	0x0B51 n=1	GL_LIGHT_MODEL_LOCAL_VIEWER
	0x0B52 n=1	GL_LIGHT_MODEL_TWO_SIDE
	0x0B53 n=4	GL_LIGHT_MODEL_AMBIENT
	else n=0	command is erroneous
4*n	LISTofFLOAT32	params

## LightModeli

2	12	rendering command length
2	92	rendering command opcode
4	ENUM	pname
4	INT32	param

## LightModeliv

2 2	8+4*n 93	rendering command length rendering command opcode
4	ENUM	pname
	0x81F8 n=1	GL_LIGHT_MODEL_COLOR_CONTROL
	0x0B51 n=1	GL_LIGHT_MODEL_LOCAL_VIEWER
	0x0B52 n=1	GL_LIGHT_MODEL_TWO_SIDE
	0x0B53 n=4	GL_LIGHT_MODEL_AMBIENT
	else n=0	command is erroneous
4*n	LISTofINT32	params

## LineStipple

2	12	rendering command length
2	94	rendering command opcode
4	INT32	factor
2	CARD16	pattern
2		unused

#### LineWidth

2	8	rendering command length
2	95	rendering command opcode
4	FLOAT32	width

## ListBase

2	8	rendering command length
2	3	rendering command opcode
4	CARD32	base

## LoadIdentity

2	4	rendering command length
2	176	rendering command opcode

#### LoadMatrixd

2	132	rendering command length
2	178	rendering command opcode
128	LISTofFLOAT64	m

## Load Matrix f

2	68	rendering command length
2	177	rendering command opcode
64	LISTofFLOAT32	m

#### LoadName

2	8	rendering command length
2	122	rendering command opcode
4	CARD32	name

# LogicOp

2	8	rendering command length
2	161	rendering command opcode
4	ENUM	opcode

# MapGrid1d

pcode

# MapGrid1f

2	16	rendering command length
2	148	rendering command opcode
4	TNT32	ıın

4	FLOAT32	u1
4	FLOAT32	u2

## MapGrid2d

2	44	rendering command length
2	149	rendering command opcode
8	FLOAT64	u1
8	FLOAT64	u2
8	FLOAT64	v1
8	FLOAT64	v2
4	INT32	un
4	TNT32	vn

## MapGrid2f

2	28	rendering command length
2	150	rendering command opcode
4	INT32	un
4	FLOAT32	u1
4	FLOAT32	u2
4	INT32	vn
4	FLOAT32	v1
4	FLOAT32	v2

#### Materialf

2	16	rendering command length
2	96	rendering command opcode
4	ENUM	face
4	ENUM	pname
4	FLOAT32	param

#### Materialfv

2	12+4*n	rendering command length
2	97	rendering command opcode
4	ENUM	face

4	ENUM	pname
	0x1200 n=4	GL_AMBIENT
	0x1201 n=4	GL_DIFFUSE
	0x1202 n=4	GL_SPECULAR
	0x1600 n=4	GL_EMISSION
	0x1601 n=1	GL_SHININESS
	0x1602 n=4	GL_AMBIENT_AND_DIFFUSE
	0x1603 n=3	GL_COLOR_INDEXES
	else n=0	command is erroneous
4*n	LISTofFLOAT32	params

## Materiali

2	16	rendering command length
2	98	rendering command opcode
4	ENUM	face
4	ENUM	pname
4	INT32	param

#### Materialiv

2	12+4*n	rendering command length
2	99	rendering command opcode
4	ENUM	face
4	ENUM	pname
	0x1200 n=4	GL_AMBIENT
	0x1201 n=4	GL_DIFFUSE
	0x1202 n=4	GL_SPECULAR
	0x1600 n=4	GL_EMISSION
	0x1601 n=1	GL_SHININESS
	0x1602 n=4	GL_AMBIENT_AND_DIFFUSE
	0x1603 n=3	GL_COLOR_INDEXES
	else n=0	command is erroneous
4*n	LISTofINT32	params

## MatrixMode

2	8	rendering command length
2	179	rendering command opcode
4	ENUM	mode

## Minmax

2	16	rendering command length
2	4111	rendering command opcode
4	ENUM	target
4	ENUM	internalformat
1	BOOL	sink
3		unused

#### MultiTexCoord1dvARB

2	16	rendering command length
2	198	rendering command opcode
8	FLOAT64	v[0]
4	ENUM	target

## MultiTexCoord1fvARB

2	12	rendering command length
2	199	rendering command opcode
4	ENUM	target
4	FLOAT32	v[0]

#### MultiTexCoord1ivARB

2	12	rendering command length
2	200	rendering command opcode
4	ENUM	target
4	INT32	v[0]

#### MultiTexCoord1svARB

2	12	rendering command length
2	201	rendering command opcode
4	ENUM	target
2	INT16	v[0]
2		unused

## MultiTexCoord2dvARB

2	24	rendering command length
2	202	rendering command opcode
8	FLOAT64	v[0]
8	FLOAT64	v[1]
4	ENUM	target

#### MultiTexCoord2fvARB

2	16	rendering command length
2	203	rendering command opcode
4	ENUM	target
4	FLOAT32	v[0]
4	FLOAT32	v[1]

## MultiTexCoord2ivARB

2	16	rendering command length
2	204	rendering command opcode
4	ENUM	target
4	INT32	v[0]
4	INT32	v[1]

#### MultiTexCoord2svARB

2	12	rendering command length
2	205	rendering command opcode
4	ENUM	target
2	INT16	v[0]
2	INT16	v[1]

## MultiTexCoord3dvARB

2	32	rendering command length
2	206	rendering command opcode
8	FLOAT64	v[0]

8	FLOAT64	v[1]
8	FLOAT64	v[2]
4	ENUM	target

## MultiTexCoord3fvARB

2	20	rendering command length
2	207	rendering command opcode
4	ENUM	target
4	FLOAT32	v[0]
4	FLOAT32	v[1]
4	FLOAT32	v[2]

#### MultiTexCoord3ivARB

2	20	rendering command length
2	208	rendering command opcode
4	ENUM	target
4	INT32	v[0]
4	INT32	v[1]
4	INT32	v[2]

#### MultiTexCoord3svARB

2	16	rendering command length
2	209	rendering command opcode
4	ENUM	target
2	INT16	v[0]
2	INT16	v[1]
2	INT16	v[2]
2		unused

## MultiTexCoord4dvARB

2	40	rendering command length
2	210	rendering command opcode
8	FLOAT64	v[0]
8	FLOAT64	v[1]

8	FLOAT64	v[2]
8	FLOAT64	v[3]
4	ENUM	target

#### MultiTexCoord4fvARB

2	24	rendering command length
2	211	rendering command opcode
4	ENUM	target
4	FLOAT32	v[0]
4	FLOAT32	v[1]
4	FLOAT32	v[2]
4	FLOAT32	v[3]

#### MultiTexCoord4ivARB

2 2	24 212	rendering command length rendering command opcode
4	ENUM	target
4	INT32	v[0]
4	INT32	v[1]
4	INT32	v[2]
4	INT32	v[3]

#### MultiTexCoord4svARB

2	16	rendering command length
2	213	rendering command opcode
4	ENUM	target
2	INT16	v[0]
2	INT16	v[1]
2	INT16	v[2]
2	INT16	v[3]

## MultMatrixd

2	132	rendering command length
2	181	rendering command opcode

128 LISTOFFLOAT64 m

#### MultMatrixf

2	68	rendering command length
2	180	rendering command opcode
64	LISTofFLOAT32	m

#### Normal3bv

2	8	rendering command length
2	28	rendering command opcode
1	INT8	v[0]
1	INT8	v[1]
1	INT8	v[2]
1		unused

#### Normal 3 dv

2	28	rendering command length
2	29	rendering command opcode
8	FLOAT64	v[0]
8	FLOAT64	v[1]
8	FLOAT64	v[2]

#### Normal3fv

2	16	rendering command length
2	30	rendering command opcode
4	FLOAT32	v[0]
4	FLOAT32	v[1]
4	FLOAT32	v[2]

### Normal3iv

2 16 rendering command length

2	31	rendering command opcode
4	INT32	v[0]
4	INT32	v[1]
4	INT32	v[2]

#### Normal3sv

2	12	rendering command length
2	32	rendering command opcode
2	INT16	v[0]
2	INT16	v[1]
2	INT16	v[2]
2		unused

## Ortho

2	52	rendering command length
2	182	rendering command opcode
8	FLOAT64	left
8	FLOAT64	right
8	FLOAT64	bottom
8	FLOAT64	top
8	FLOAT64	zNear
8	FLOAT64	zFar

# PassThrough

2	8	rendering command length
2	123	rendering command opcode
4	FLOAT32	token

## PixelTransferf

ng command length
ng command opcode

## PixelTransferi

2	12	rendering command length
2	167	rendering command opcode
4	ENUM	pname
4	INT32	param

#### **PixelZoom**

2	12	rendering command length
2	165	rendering command opcode
4	FLOAT32	xfactor
4	FLOAT32	yfactor

## **PointSize**

2	8	rendering command length
2	100	rendering command opcode
4	FLOAT32	size

# PolygonMode

2	12	rendering command length
2	101	rendering command opcode
4	ENUM	face
4	ENUM	mode

# PolygonOffset

2	12	rendering command length
2	192	rendering command opcode
4	FLOAT32	factor
4	FLOAT32	units

# PopAttrib

2	4	rendering command length
2	141	rendering command opcode

## **PopMatrix**

2	4	rendering command length
2	183	rendering command opcode

# PopName

2	4	rendering command length
2	124	rendering command opcode

## ${\bf Prioritize Textures}$

2	cmdlen	rendering command length
2	4118	rendering command opcode
4	INT32	n
n*4	LISTofCARD32	textures
n*4	LISTofFLOAT32	priorities

### **PushAttrib**

2	8	rendering command length
2	142	rendering command opcode
4	BITFIELD	mask

## PushMatrix

2	4	rendering command length
2	184	rendering command opcode

#### PushName

2	8	rendering command length
2	125	rendering command opcode
4	CARD32	name

#### RasterPos2dv

2	20	rendering command length
2	33	rendering command opcode
8	FLOAT64	v[0]
8	FLOAT64	v[1]

#### RasterPos2fv

2	12	rendering command length
2	34	rendering command opcode
4	FLOAT32	v[0]
4	FLOAT32	v[1]

#### RasterPos2iv

2	12	rendering command length
2	35	rendering command opcode
4	INT32	v[0]
4	INT32	v[1]

#### RasterPos2sv

2	8	rendering command length
2	36	rendering command opcode
2	INT16	v[0]
2	INT16	v[1]

## RasterPos3dv

2 28 rendering command length

2	37	rendering command opcode
8	FLOAT64	v[0]
8	FLOAT64	v[1]
8	FLOAT64	v[2]

## RasterPos3fv

2	16	rendering command length
2	38	rendering command opcode
4	FLOAT32	v[0]
4	FLOAT32	v[1]
4	FLOAT32	v[2]

## RasterPos3iv

2	16	rendering command length
2	39	rendering command opcode
4	INT32	v[0]
4	INT32	v[1]
4	INT32	v[2]

## RasterPos3sv

2	12	rendering command length
2	40	rendering command opcode
2	INT16	v[0]
2	INT16	v[1]
2	INT16	v[2]
2		unused

#### RasterPos4dv

2	36	rendering command length
2	41	rendering command opcode
8	FLOAT64	v[0]
8	FLOAT64	v[1]
8	FLOAT64	v[2]
8	FLOAT64	v[3]

## RasterPos4fv

2	20	rendering command length
2	42	rendering command opcode
4	FLOAT32	v[0]
4	FLOAT32	v[1]
4	FLOAT32	v[2]
4	FLOAT32	v[3]

## RasterPos4iv

2	20	rendering command length
2	43	rendering command opcode
4	INT32	v[0]
4	INT32	v[1]
4	INT32	v[2]
4	INT32	v[3]

## RasterPos4sv

2	12	rendering command length
2	44	rendering command opcode
2	INT16	v[0]
2	INT16	v[1]
2	INT16	v[2]
2	INT16	v[3]

#### ReadBuffer

2	8	rendering command length
2	171	rendering command opcode
4	ENUM	mode

## Rectdy

2	36	rendering command length
2	45	rendering command opcode

8	FLOAT64	v1[0]
8	FLOAT64	v1[1]
8	FLOAT64	v2[0]
8	FLOAT64	v2[1]

## Rectfv

2	20	rendering command length
2	46	rendering command opcode
4	FLOAT32	v1[0]
4	FLOAT32	v1[1]
4	FLOAT32	v2[0]
4	FLOAT32	v2[1]

## Rectiv

2	20	rendering command length
2	47	rendering command opcode
4	INT32	v1[0]
4	INT32	v1[1]
4	INT32	v2[0]
4	INT32	v2[1]

#### Rectsv

2	12	rendering command length
2	48	rendering command opcode
2	INT16	v1[0]
2	INT16	v1[1]
2	INT16	v2[0]
2	INT16	v2[1]

## ResetHistogram

2	8	rendering command length
2	4112	rendering command opcode
4	ENUM	target

## ResetMinmax

2	8	rendering command length
2	4113	rendering command opcode
4	ENUM	target

#### Rotated

2	36	rendering command length
2	185	rendering command opcode
8	FLOAT64	angle
8	FLOAT64	X
8	FLOAT64	у
8	FLOAT64	Z

#### Rotatef

2	20	rendering command length
2	186	rendering command opcode
4	FLOAT32	angle
4	FLOAT32	X
4	FLOAT32	y
4	FLOAT32	Z

### Scaled

2	28	rendering command length
2	187	rendering command opcode
8	FLOAT64	X
8	FLOAT64	y
8	FLOAT64	Z

## Scalef

2	16	rendering command length
2	188	rendering command opcode
4	FLOAT32	X

4	FLOAT32	у
4	FLOAT32	Z

#### Scissor

2	20	rendering command length
2	103	rendering command opcode
4	INT32	X
4	INT32	у
4	INT32	width
4	INT32	height

#### ShadeModel

2	8	rendering command length
2	104	rendering command opcode
4	ENUM	mode

#### StencilFunc

2	16	rendering command length
2	162	rendering command opcode
4	ENUM	func
4	INT32	ref
4	CARD32	mask

#### StencilMask

2	8	rendering command length
2	133	rendering command opcode
4	CARD32	mask

# StencilOp

2 16 rendering command length

2	163	rendering command opcode
4	ENUM	fail
4	ENUM	zfail
4	ENUM	zpass

## TexCoord1dv

2	12	rendering command length
2	49	rendering command opcode
8	FLOAT64	v[0]

#### TexCoord1fv

2	8	rendering command length
2	50	rendering command opcode
4	FLOAT32	v[0]

## TexCoord1iv

2	8	rendering command length
2	51	rendering command opcode
4	INT32	v[0]

## TexCoord1sv

2	8	rendering command length
2	52	rendering command opcode
2	INT16	v[0]
2		unused

#### TexCoord2dv

2	20	rendering command length
2	53	rendering command opcode
8	FLOAT64	v[0]
8	FLOAT64	v[1]

## TexCoord2 fv

2	12	rendering command length
2	54	rendering command opcode
4	FLOAT32	v[0]
4	FLOAT32	v[1]

#### TexCoord2iv

2	12	rendering command length
2	55	rendering command opcode
4	INT32	v[0]
4	INT32	v[1]

#### TexCoord2sv

2	8	rendering command length
2	56	rendering command opcode
2	INT16	v[0]
2	INT16	v[1]

#### TexCoord3dv

2	28	rendering command length
2	57	rendering command opcode
8	FLOAT64	v[0]
8	FLOAT64	v[1]
8	FLOAT64	v[2]

## TexCoord3 fv

2	16	rendering command length
2	58	rendering command opcode
4	FLOAT32	v[0]
4	FLOAT32	v[1]
4	FLOAT32	v[2]

## TexCoord3iv

2	16	rendering command length
2	59	rendering command opcode
4	INT32	v[0]
4	INT32	v[1]
4	INT32	v[2]

#### TexCoord3sv

2	12	rendering command length
2	60	rendering command opcode
2	INT16	v[0]
2	INT16	v[1]
2	INT16	v[2]
2		unused

## TexCoord4dv

2	36	rendering command length
2	61	rendering command opcode
8	FLOAT64	v[0]
8	FLOAT64	v[1]
8	FLOAT64	v[2]
8	FLOAT64	v[3]

## TexCoord4fv

2	20	rendering command length
2	62	rendering command opcode
4	FLOAT32	v[0]
4	FLOAT32	v[1]
4	FLOAT32	v[2]
4	FLOAT32	v[3]

## TexCoord4iv

2 20 rendering command length

2	63	rendering command opcode
4	INT32	v[0]
4	INT32	v[1]
4	INT32	v[2]
4	INT32	v[3]

#### TexCoord4sv

2	12	rendering command length
2	64	rendering command opcode
2	INT16	v[0]
2	INT16	v[1]
2	INT16	v[2]
2	INT16	v[3]

#### TexEnvf

2	16	rendering command length
2	111	rendering command opcode
4	ENUM	target
4	ENUM	pname
4	FLOAT32	param

#### TexEnvfv

2	12+4*n	rendering command length
2	112	rendering command opcode
4	ENUM	target
4	ENUM	pname
	0x2200 n=1	GL_TEXTURE_ENV_MODE
	0x2201 n=4	GL_TEXTURE_ENV_COLOR
	else n=0	command is erroneous
4*n	LISTofFLOAT32	params

## TexEnvi

2	16	rendering command length
2	113	rendering command opcode

4	ENUM	target
4	ENUM	pname
4	INT32	param

## **TexEnviv**

2	12+4*n	rendering command length
2	114	rendering command opcode
4	ENUM	target
4	ENUM	pname
	0x2200 n=1	GL_TEXTURE_ENV_MODE
	0x2201 n=4	GL_TEXTURE_ENV_COLOR
	else n=0	command is erroneous
4*n	LISTofINT32	params

#### TexGend

2	20	rendering command length
2	115	rendering command opcode
8	FLOAT64	param
4	ENUM	coord
4	ENUM	pname

#### TexGendv

2	12+8*n	rendering command length
2	116	rendering command opcode
4	ENUM	coord
4	ENUM	pname
	0x2500 n=1	GL_TEXTURE_GEN_MODE
	0x2501 n=4	GL_OBJECT_PLANE
	0x2502 n=4	GL_EYE_PLANE
	else n=0	command is erroneous
8*n	LISTofFLOAT64	params

## TexGenf

2 16 rendering command length

2	117	rendering command opcode
4	ENUM	coord
4	ENUM	pname
4	FLOAT32	param

## TexGenfv

2	12+4*n	rendering command length
2	118	rendering command opcode
4	ENUM	coord
4	ENUM	pname
	0x2500 n=1	GL_TEXTURE_GEN_MODE
	0x2501 n=4	GL_OBJECT_PLANE
	0x2502 n=4	GL_EYE_PLANE
	else n=0	command is erroneous
4*n	LISTofFLOAT32	params

#### TexGeni

2	16	rendering command length
2	119	rendering command opcode
4	ENUM	coord
4	ENUM	pname
4	INT32	param

## TexGeniv

2	12+4*n	rendering command length
2	120	rendering command opcode
4	ENUM	coord
4	ENUM	pname
	0x2500 n=1	GL_TEXTURE_GEN_MODE
	0x2501 n=4	GL_OBJECT_PLANE
	0x2502 n=4	GL_EYE_PLANE
	else n=0	command is erroneous
4*n	LISTofINT32	params

#### **TexParameterf**

2	16	rendering command length
2	105	rendering command opcode
4	ENUM	target
4	ENUM	pname
4	FLOAT32	param

#### **TexParameterfy**

2	12+4*n	rendering command length
2	106	rendering command opcode
4	ENUM	target
4	ENUM	pname
	0x1004 n=4	GL_TEXTURE_BORDER_COLOR
	0x2800 n=1	GL_TEXTURE_MAG_FILTER
	0x2801 n=1	GL_TEXTURE_MIN_FILTER
	0x2802 n=1	GL_TEXTURE_WRAP_S
	0x2803 n=1	GL_TEXTURE_WRAP_T
	else n=0	command is erroneous
4*n	LISTofFLOAT32	params

## TexParameteri

2	16	rendering command length
2	107	rendering command opcode
4	ENUM	target
4	ENUM	pname
4	INT32	param

## **TexParameteriv**

2	12+4*n	rendering command length
2	108	rendering command opcode
4	ENUM	target
4	ENUM	pname
	0x1004 n=4	GL_TEXTURE_BORDER_COLOR
	0x2800 n=1	GL_TEXTURE_MAG_FILTER
	0x2801 n=1	GL_TEXTURE_MIN_FILTER
	0x2802 n=1	GL_TEXTURE_WRAP_S
	0x2803 n=1	GL_TEXTURE_WRAP_T
	else n=0	command is erroneous

Version 1.3 - 2 June 1999

4\*n LISTOFINT32 params

#### **Translated**

2	28	rendering command length
2	189	rendering command opcode
8	FLOAT64	X
8	FLOAT64	y
8	FLOAT64	Z

#### Translatef

2	16	rendering command length
2	190	rendering command opcode
4	FLOAT32	X
4	FLOAT32	y
4	FLOAT32	Z

#### Vertex2dv

2	20	rendering command length
2	65	rendering command opcode
8	FLOAT64	v[0]
8	FLOAT64	v[1]

#### Vertex2fv

2	12	rendering command length
2	66	rendering command opcode
4	FLOAT32	v[0]
4	FLOAT32	v[1]

### Vertex2iv

2 12 rendering command length

2	67	rendering command opcode
4	INT32	v[0]
4	INT32	v[1]

#### Vertex2sv

2	8	rendering command length
2	68	rendering command opcode
2	INT16	v[0]
2	INT16	v[1]

#### Vertex3dv

2	28	rendering command length
2	69	rendering command opcode
8	FLOAT64	v[0]
8	FLOAT64	v[1]
8	FLOAT64	v[2]

## Vertex3fv

2	16	rendering command length
2	70	rendering command opcode
4	FLOAT32	v[0]
4	FLOAT32	v[1]
4	FLOAT32	v[2]

#### Vertex3iv

2	16	rendering command length
2	71	rendering command opcode
4	INT32	v[0]
4	INT32	v[1]
4	INT32	v[2]

#### Vertex3sv

2	12	rendering command length
2	72	rendering command opcode
2	INT16	v[0]
2	INT16	v[1]
2	INT16	v[2]
2		unused

#### Vertex4dv

2	36	rendering command length
2	73	rendering command opcode
8	FLOAT64	v[0]
8	FLOAT64	v[1]
8	FLOAT64	v[2]
8	FLOAT64	v[3]

## Vertex4fv

2	20	rendering command length
2	74	rendering command opcode
4	FLOAT32	v[0]
4	FLOAT32	v[1]
4	FLOAT32	v[2]
4	FLOAT32	v[3]

#### Vertex4iv

2	20	rendering command length
2	75	rendering command opcode
4	INT32	v[0]
4	INT32	v[1]
4	INT32	v[2]
4	INT32	v[3]

## Vertex4sv

2	12	rendering command length
2	76	rendering command opcode

2	INT16	v[0]
2	INT16	v[1]
2	INT16	v[2]
2	INT16	v[3]

## Viewport

2	20	rendering command length
2	191	rendering command opcode
4	INT32	X
4	INT32	y
4	INT32	width
4	INT32	height

## xImage1D

2	32	rendering command length
2	4119	rendering command opcode
4	ENUM	target
4	INT32	level
4	ENUM	internalformat
4	INT32	X
4	INT32	y
4	INT32	width
4	TNT32	border

# 2.3.4 GL Rendering Commands That May Be Large

These commands are potentially large, and hence can be sent in a **glXRender** or **glXRenderLarge** request.

#### **CallLists**

2	12+m+p	rendering command length
2	2	rendering command opcode
4	INT32	n
4	ENUM	type
m	(see below)	lists

p unused, p=pad(m)

The type and size of lists is determined by type, as shown in Table 2.1.

type	encoding of type	type of lists	m (bytes)
GL_BYTE	0x1400	LISTofINT8	n
GL_UNSIGNED_BYTE	0x1401	LISTofCARD8	n
GL_SHORT	0x1402	LISTofINT16	n*2
GL_UNSIGNED_SHORT	0x1403	LISTofCARD16	n*2
GL_INT	0x1404	LISTofINT32	n*4
GL_UNSIGNED_INT	0x1405	LISTofCARD32	n*4
GL_FLOAT	0x1406	LISTofFLOAT32	n*4
GL_2_BYTES	0x1407	LISTofBYTE	n*2
GL_3_BYTES	0x1408	LISTofBYTE	n*3
GL_4_BYTES	0x1409	LISTofBYTE	n*4

Table 2.1: Type and size of lists

If type is not one of the types in this table, the command is erroneous and m = 0.

If *type* is GL\_2\_BYTES, GL\_3\_BYTES, or GL\_4\_BYTES, *lists* is treated as an array of unsigned bytes, and each successive 2, 3, or 4 bytes are used to construct a list index, as described for this command in the OpenGL Spec.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	16+m+p	rendering command length
4	2	rendering command opcode

#### **DrawArrays**

2	16+(12*m)+(s*n)	rendering command length
2	193	rendering command opcode
4	CARD32	n (number of array elements)
4	CARD32	m (number of enabled arrays)
4	ENUM	mode (GL_POINTS etc.)
12*m	LISTofARRAY_INFO	
s*n	LISTofVERTEX_DATA	

Where s = ns + cs + is + ts + es + vs + np + cp + ip + tp + ep + vp. (See description below, under VERTEX\_DATA.) Note that if an array is disabled then no information is sent for it. For example, when the normal array is disabled, there is no ARRAY\_INFO record for the normal array and ns and np are both zero.

Note that the list of ARRAY\_INFO is unordered: since the ARRAY\_INFO record contains the array type, the arrays in the list may be stored in any order. Also, the VERTEX\_-DATA list is a packed list of vertices. For each vertex, data is retrieved from the enabled arrays, and stored in the list.

ARRAY\_INFO

4	ENUM		data type
	0x1400	i=1	GL_BYTE
	0x1401	i=1	UNSIGNED_BYTE
	0x1402	i=2	SHORT
	0x1403	i=2	UNSIGNED_SHORT
	0x1404	i=4	INT
	0x1405	i=4	UNSIGNED_INT
	0x1406	i=4	FLOAT
	0x140A	i=8	DOUBLE
4	INT32		j (number of values in array element)
4	ENUM		array type
	0x8074	j=2/3/4	VERTEX_ARRAY
	0x8075	j=3	NORMAL_ARRAY
	0x8076	j=3/4	COLOR_ARRAY
	0x8077	j=1	INDEX_ARRAY
	0x8078	j=1/2/3/4	TEXTURE_COORD_ARRAY
	0x8079	j=1	EDGE_FLAG_ARRAY

For each array, the size of an array element is i\*j. Some arrays (e.g., the texture coordinate array) support different data sizes; for these arrays, the size, j, is specified when the array is defined.

VERTEX\_DATA

```
If the edge flag array is enabled:
es LISTOFBYTE edge flag array element
ep unused, ep=pad(es)

If the texture coord array is enabled:
ts LISTOFBYTE texture coord array element
```

tp unused, tp=pad(ts)

If the color array is enabled:
cs LISTofBYTE color array element
cp unused, cp=pad(cs)

If the index array is enabled:
is LISTofBYTE index array element
ip unused, ip=pad(is)

If the normal array is enabled:
ns LISTofBYTE normal array element
np unused, np=pad(ns)

LISTOFBYTE vertex array element

If the vertex array is enabled:

where ns, cs, is, ts, es, vs are the size of the normal, color, index, texture, edge and vertex array elements and np, cp, ip, tp, ep, vp are the padding for the normal, color, index, texture, edge and vertex array elements, respectively.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	20+(12*m)+(s*n)	rendering command length
4	4116	rendering command opcode

unused, vp=pad(vs)

#### **PixelMapfv**

vp

length
opcode

If (mapsize  $\geq$  0), n=4\*mapsize; otherwise, the command is erroneous and n = 0.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	16+n	rendering command length
4	168	rendering command opcode

#### **PixelMapuiv**

2	12+n	rendering command length
2	169	rendering command opcode
4	ENUM	map
4	INT32	mapsize
n	LISTofCARD32	values

If (mapsize  $\geq$  0), n=4\*mapsize; otherwise, the command is erroneous and n = 0.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	16+n	rendering command length
4	169	rendering command opcode

#### **PixelMapusv**

2	12+n+p	rendering command length
2	170	rendering command opcode
4	ENUM	map
4	INT32	mapsize
n	LISTofCARD16	values
p		unused, p=pad(n)

If (mapsize  $\geq$  0), n=2\*mapsize; otherwise, the command is erroneous and n = 0.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	16+n+p	rendering command length
4	170	rendering command opcode

### **PrioritizeTextures**

2	8+8*n	rendering command length
2	4118	rendering command opcode
4	INT32	n
n*4	LISTofCARD32	textures
n*4	LISTofFLOAT32	priorities

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	12+8*n	rendering command length
4	4118	rendering command opcode

### 2.3.5 GL Rendering Commands with Evaluator Map Data

These commands have arrays of evaluator control points, whose structure is described below. These commands are also potentially large, and can be sent in a **glXRender** or **glXRenderLarge** request.

For the commands **Map1d**, **Map1d**, **Map2d**, and **Map2f**, the number of floating-point values per control point, k, is determined from the target parameter:

target	encoding of target	k
GL_MAP1_COLOR_4	0x0D90	4
GL_MAP1_INDEX	0x0D91	1
GL_MAP1_NORMAL	0x0D92	3
GL_MAP1_TEXTURE_COORD_1	0x0D93	1
GL_MAP1_TEXTURE_COORD_2	0x0D94	2
GL_MAP1_TEXTURE_COORD_3	0x0D95	3
GL_MAP1_TEXTURE_COORD_4	0x0D96	4
GL_MAP1_VERTEX_3	0x0D97	3
GL_MAP1_VERTEX_4	0x0D98	4

Table 2.2: Values Per Control Point for Map1d and Map1f

### Map1d

2 28+n rendering command length

Version 1.3 - 2 June 1999

target	encoding of target	k
GL_MAP2_COLOR_4	0x0DB0	4
GL_MAP2_INDEX	0x0DB1	1
GL_MAP2_NORMAL	0x0DB2	3
GL_MAP2_TEXTURE_COORD_1	0x0DB3	1
GL_MAP2_TEXTURE_COORD_2	0x0DB4	2
GL_MAP2_TEXTURE_COORD_3	0x0DB5	3
GL_MAP2_TEXTURE_COORD_4	0x0DB6	4
GL_MAP2_VERTEX_3	0x0DB7	3
GL_MAP2_VERTEX_4	0x0DB8	4

Table 2.3: Values Per Control Point for Map2d and Map2f

2	143	rendering command opcode		
8	FLOAT64	u1		
8	FLOAT64	u2		
4	ENUM	target		
4	INT32	order		
n	LISTOFFLOAT64	points		

Determine k from Table 2.2; then  $n = order \cdot k \cdot 8$ . The control point  $\mathbf{R}_i$ , consisting of k values, starts at byte  $(i \cdot k \cdot 8)$  of points;  $0 \le i < order$ . If  $order \le 0$  or target is not one of the ones listed in Table 2.2, the command is erroneous and n = 0.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	32+n	rendering command length
4	143	rendering command opcode

### Map1f

2	20+n	rendering command length
2	144	rendering command opcode
4	ENUM	target
4	FLOAT32	u1
4	FLOAT32	u2
4	INT32	order
n	LISTofFLOAT32	points

Determine k from Table 2.2; then  $n = order \cdot k \cdot 4$ . The control point  $\mathbf{R}_i$ , consisting of k values, starts at byte  $(i \cdot k \cdot 4)$  of points;  $0 \le i < order$ . If  $order \le 0$  or target is not one of the ones listed in Table 2.2, the command is erroneous and n = 0.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	24+n	rendering command length
4	144	rendering command opcode

### Map2d

2	48+n	rendering command length
2	145	rendering command opcode
8	FLOAT64	u1
8	FLOAT64	u2
8	FLOAT64	v1
8	FLOAT64	v2
4	ENUM	target
4	INT32	uorder
4	INT32	vorder
n	LISTOFFLOAT64	points

Determine k from Table 2.3; then  $\mathbf{n} = vorder \cdot uorder \cdot k \cdot 8$ . The control point  $\mathbf{R}_{ij}$ , consisting of k values, starts at byte  $[(i \cdot vorder + j) \cdot k \cdot 8]$  of points;  $0 \leq i < uorder$  and  $0 \leq j < vorder$ . If  $vorder \leq 0$  or  $uorder \leq 0$  or target is not one of the ones listed in Table 2.3, the command is erroneous and  $\mathbf{n} = 0$ .

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	52+n	rendering command length
4	145	rendering command opcode

### Map2f

2	32+n	rendering command length
2	146	rendering command opcode
4	ENUM	target

Version 1.3 - 2 June 1999

4	CARD32	alignment
4	INT32	width
4	INT32	height
4	FLOAT32	xorig
4	FLOAT32	yorig
4	FLOAT32	xmove
4	FLOAT32	ymove
n	LISTofBYTE	bitmap
p		unused, p=pad(n)

If width < 0 or height < 0, the command is erroneous and n = 0.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	52+n+p	rendering command length
4	5	rendering command opcode

The structure of *bitmap* is described in more detail in Appendix A, using the parameters *lsb\_first*, *row\_length*, *skip\_rows*, *skip\_pixels*, *alignment*, *width*, and *height* as given in the request, *format*=GL\_COLOR\_INDEX, and *type*=GL\_BITMAP.

### ColorTable

2	44+n+p	rendering command length
2	2053	rendering command opcode
1	BOOL	swap_bytes
1	BOOL	lsb_first
2		unused
4	CARD32	row_length
4	CARD32	skip_rows
4	CARD32	skip_pixels
4	CARD32	alignment
4	ENUM	target
4	ENUM	internalformat
4	INT32	width
4	ENUM	format
4	ENUM	type
n	LISTofBYTE	table
p		unused, p=pad(n)

If width < 0, then the command is erroneous and n = 0.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	48+n+p	rendering command length
4	2053	rendering command opcode

The structure of *table* is described in more detail in Appendix A, using the parameters *swap\_bytes*, *lsb\_first*, *row\_length*, *skip\_rows*, *skip\_pixels*, *alignment*, *width*, *format*, and *type* as given in the request, and *height*=1.

#### ColorSubTable

2	44+n+p	rendering command length
2	195	rendering command opcode
1	BOOL	swap_bytes
1	BOOL	lsb_first
2		unused
4	CARD32	row_length
4	CARD32	skip_rows
4	CARD32	skip_pixels
4	CARD32	alignment
4	ENUM	target
4	INT32	start
4	INT32	count
4	ENUM	format
4	ENUM	type
n	LISTOfBYTE	table
p		unused, p=pad(n)

If count < 0, then the command is erroneous and n = 0.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	48+n+p	rendering command length
4	195	rendering command opcode

The structure of *table* is described in more detail in Appendix A, using the parameters

swap\_bytes, Isb\_first, row\_length, skip\_rows, skip\_pixels, alignment, format, and type as given in the request, a width of count, and height=1.

#### ConvolutionFilter1D

2	48+n+p	rendering command length
2	4101	rendering command opcode
1	BOOL	swap_bytes
1	BOOL	lsb_first
2		unused
4	CARD32	row_length
4	CARD32	skip_rows
4	CARD32	skip_pixels
4	CARD32	alignment
4	ENUM	target
4	ENUM	internalformat
4	INT32	width
4	INT32	height
4	ENUM	format
4	ENUM	type
n	LISTofBYTE	pixels
p		unused, p=pad(n)

If width < 0, then the command is erroneous and n = 0.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	52+n+p	rendering command length
4	4101	rendering command opcode

The structure of *pixels* is described in more detail in Appendix A, using the parameters *swap\_bytes*, *lsb\_first*, *row\_length*, *skip\_rows*, *skip\_pixels*, *alignment*, *width*, *format*, and *type* as given in the request, and *height*=1.

### ConvolutionFilter2D

2 48+n- 2 4102	+p	rendering command length rendering command opcode
1 BOOL		swap_bytes
1 BOOL		lsb_first
_		1 7

2		unused
4	CARD32	row_length
4	CARD32	skip_rows
4	CARD32	skip_pixels
4	CARD32	alignment
4	ENUM	target
4	ENUM	internalformat
4	INT32	width
4	INT32	height
4	ENUM	format
4	ENUM	type
n	LISTOfBYTE	pixels
p		unused, p=pad(n)

If width < 0 or height < 0, then the command is erroneous and n = 0.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	52+n+p	rendering command length
4	4102	rendering command opcode

The structure of *pixels* is described in more detail in Appendix A, using the parameters *swap\_bytes*, *lsb\_first*, *row\_length*, *skip\_rows*, *skip\_pixels*, *alignment*, *width*, *height*, *format*, and *type* as given in the request.

### SeparableFilter2D

2 2 1	48+n1+p1+n2+p2 4109 BOOL	rendering command length rendering command opcode swap_bytes
1	BOOL	lsb_first
2		unused
4	CARD32	row_length
4	CARD32	skip_rows
4	CARD32	skip_pixels
4	CARD32	alignment
4	ENUM	target
4	ENUM	internalformat
4	INT32	row_width
4	INT32	col_height
4	ENUM	format

Version 1.3 - 2 June 1999

If  $row\_width < 0$  or  $col\_height < 0$ , then the command is erroneous and n1 = n2 = 0.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	52+n1+p1+n2+p2	rendering command length
4	4109	rendering command opcode

The structure of *row* is described in more detail in Appendix A, using the parameters  $swap\_bytes$ ,  $lsb\_first$ ,  $row\_length$ ,  $skip\_rows$ ,  $skip\_pixels$ , alignment, format, and type as given in the request, a width of  $row\_width$ , and height=1. The structure of column is the same (it is also a one-dimensional image) except that it has parameters width=1 and a height of  $col\_height$ .

### **DrawPixels**

2	40+n+p	rendering command length
2	173	rendering command opcode
1	BOOL	swap_bytes
1	BOOL	lsb_first
2		unused
4	CARD32	row_length
4	CARD32	skip_rows
4	CARD32	skip_pixels
4	CARD32	alignment
4	INT32	width
4	INT32	height
4	ENUM	format
4	ENUM	type
n	LISTofBYTE	pixels
p		unused, $p=pad(n)$

If width < 0 or height < 0, then the command is erroneous and n = 0.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	44+n+p	rendering command length
4	173	rendering command opcode

The structure of *pixels* is described in more detail in Appendix A, using the parameters *swap\_bytes*, *lsb\_first*, *row\_length*, *skip\_rows*, *skip\_pixels*, *alignment*, *width*, *height*, *format*, and *type* as given in the request.

### PolygonStipple

2	24+n+p	rendering command length
2	102	rendering command opcode
1		unused
1	BOOL	lsb_first
2		unused
4	CARD32	row_length
4	CARD32	skip_rows
4	CARD32	skip_pixels
4	CARD32	alignment
n	LISTOfBYTE	mask
p		unused, p=pad(n)

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	28+n+p	rendering command length
4	102	rendering command opcode

The structure of *mask* is described in more detail in Appendix A, using the parameters *lsb\_first*, *row\_length*, *skip\_rows*, *skip\_pixels*, and *alignment* as given in the request, and *format*=GL\_COLOR\_INDEX, *type*=GL\_BITMAP, *width*=32, and *height*=32.

### TexImage1D

2	56+n+p	rendering command length
2	109	rendering command opcode
1	BOOT	swap_bytes

1	BOOL	lsb_first
2		unused
4	CARD32	row_length
4	CARD32	skip_rows
4	CARD32	skip_pixels
4	CARD32	alignment
4	ENUM	target
4	INT32	level
4	INT32	components
4	INT32	width
4		unused
4	INT32	border
4	ENUM	format
4	ENUM	type
n	LISTofBYTE	image
р		unused, $p=pad(n)$

If width < 0, then the command is erroneous and n = 0.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	60+n+p	rendering command length
4	109	rendering command opcode

The structure of *image* is described in more detail in Appendix A, using the parameters *swap\_bytes*, *lsb\_first*, *row\_length*, *skip\_rows*, *skip\_pixels*, *alignment*, *width*, *format*, and *type* as given in the request, and *height*=1.

### TexImage2D

2 56+n+p rendering command	l length
2 110 rendering command	l opcode
1 BOOL swap_bytes	
1 BOOL lsb_first	
2 unused	
4 CARD32 row_length	
4 CARD32 skip_rows	
4 CARD32 skip_pixels	
4 CARD32 alignment	
4 ENUM target	
4 INT32 level	

Version 1.3 - 2 June 1999

```
4
                                 components
      INT32
4
                                 width
      INT32
4
                                height
      INT32
4
                                border
      INT32
4
      ENUM
                                 format
4
      ENUM
                                type
n
      LISTofBYTE
                                 image
                                 unused, p=pad(n)
p
```

If width < 0 or height < 0, then the command is erroneous and n = 0.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	60+n+p	rendering command length
4	110	rendering command opcode

The structure of *image* is described in more detail in Appendix A, using the parameters *swap\_bytes*, *lsb\_first*, *row\_length*, *skip\_rows*, *skip\_pixels*, *alignment*, *width*, *height*, *format*, and *type* as given in the request.

### TexImage3D

2	84+n+p	rendering command length
2	4114	rendering command opcode
1	BOOL	swap_bytes
1	BOOL	lsb_first
2		unused
4	CARD32	row_length
4	CARD32	image_height
4	CARD32	image_depth
4	CARD32	skip_rows
4	CARD32	skip_images
4	CARD32	skip_volumes
4	CARD32	skip_pixels
4	CARD32	alignment
4	ENUM	target
4	INT32	level
4	ENUM	internalformat
4	INT32	width
4	INT32	height
4	INT32	depth

Version 1.3 - 2 June 1999

4	INT32	size4d
4	INT32	border
4	ENUM	format
4	ENUM	type
4	CARD32	null_image
n	LISTofBYTE	pixels
р		unused, $p=pad(n)$

If width < 0, height < 0, or depth < 0, then the command is erroneous and n = 0. The size4d,  $image\_depth$ , and  $skip\_volumes$  parameters in this request are ignored.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	88+n+p	rendering command length
4	4114	rendering command opcode

The structure of *image* is described in more detail in Appendix A, using the parameters *swap\_bytes*, *Isb\_first*, *row\_length*, *image\_height*, *skip\_rows*, *skip\_images*, *skip\_pixels*, *alignment*, *width*, *height*, *depth*, *format*, and *type* as given in the request.

### TexSubImage1D

2	60+n+p	rendering command length
2	4099	rendering command opcode
1	BOOL	swap_bytes
1	BOOL	lsb_first
2		unused
4	CARD32	row_length
4	CARD32	skip_rows
4	CARD32	skip_pixels
4	CARD32	alignment
4	ENUM	target
4	INT32	level
4	INT32	xoffset
4	INT32	yoffset
4	INT32	width
4	INT32	height
4	ENUM	format
4	ENUM	type
4	CARD32	unused
n	LISTofBYTE	image

Version 1.3 - 2 June 1999

p unused, p=pad(n)

If width < 0, then the command is erroneous and n = 0. The *yoffset* and *height* parameters in this request are ignored.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	64+n+p	rendering command length
4	4099	rendering command opcode

The structure of *image* is described in more detail in Appendix A, using the parameters *swap\_bytes*, *lsb\_first*, *row\_length*, *skip\_rows*, *skip\_pixels*, *alignment*, *width*, *format*, and *type* as given in the request, and *height*=1.

### TexSubImage2D

2	60+n+p	rendering command length
2	4100	rendering command opcode
1	BOOL	swap_bytes
1	BOOL	lsb_first
2		unused
4	CARD32	row_length
4	CARD32	skip_rows
4	CARD32	skip_pixels
4	CARD32	alignment
4	ENUM	target
4	INT32	level
4	INT32	xoffset
4	INT32	yoffset
4	INT32	width
4	INT32	height
4	ENUM	format
4	ENUM	type
4	CARD32	unused
n	LISTOfBYTE	image
p		unused, p=pad(n)

If width < 0 or height < 0, then the command is erroneous and n = 0.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	64+n+p	rendering command length
4	4100	rendering command opcode

The structure of *image* is described in more detail in Appendix A, using the parameters *swap\_bytes*, *lsb\_first*, *row\_length*, *skip\_rows*, *skip\_pixels*, *alignment*, *width*, *height*, *format*, and *type* as given in the request.

### TexSubImage3D

2	88+n+p	rendering command length
2	4115	rendering command opcode
1	BOOL	swap_bytes
1	BOOL	lsb_first
2		unused
4	CARD32	row_length
4	CARD32	image_height
4	CARD32	image_depth
4	CARD32	skip_rows
4	CARD32	skip_images
4	CARD32	skip_volumes
4	CARD32	skip_pixels
4	CARD32	alignment
4	ENUM	target
4	INT32	level
4	INT32	xoffset
4	INT32	yoffset
4	INT32	zoffset
4	INT32	woffset
4	INT32	width
4	INT32	height
4	INT32	depth
4	INT32	size4d
4	ENUM	format
4	ENUM	type
4	CARD32	unused
n	LISTofBYTE	image
p		unused, $p=pad(n)$

If width < 0, height < 0, or depth < 0, then the command is erroneous and n =

0. The  $woffset, size4d, image\_depth$ , and  $skip\_volumes$  parameters in this request are ignored.

If the command is encoded in a **glXRenderLarge** request, the command opcode and command length fields above are expanded to 4 bytes each:

4	92+n+p	rendering command length
4	4115	rendering command opcode

The structure of *image* is described in more detail in Appendix A, using the parameters *swap\_bytes*, *Isb\_first*, *row\_length*, *image\_height*, *skip\_rows*, *skip\_images*, *skip\_pixels*, *alignment*, *width*, *height*, *depth*, *format*, and *type* as given in the request.

# Appendix A

# **Pixel Data**

The GLX protocol encodes bitmaps, color tables, convolution, histogram, and minmax filters, pixel images, texture images, and polygon stipples in a similar and consistent manner. For convenience, all of these types of data will be referred to as *pixel data* in the following discussion. Pixel data for the rendering commands **Bitmap**, **ColorSubTable**, **ColorTable**, **ConvolutionFilterxD**, **DrawPixels**, **PolygonStipple**, **SeparableFilter2D**, **TexImagexD**, and **TexSubImagexD**, is described in Section 2.3.6; pixel data for the query commands **GetColorTable**, **GetConvolutionFilter**, **GetHistogram**, **GetMinmax**, **GetPolygonStipple**, **ReadPixels**, **GetSeparableFilter**, and **GetTexImage** is described in Section 2.2.2.

### A.1 Pixel Format and Type

As discussed in the OpenGL Spec, a unit of pixel data is a *group* of one or more *elements*. The *format* of the group determines the number of elements, and the *type* determines the size of each element. These values will be used below to describe the encoding of pixel images.

## A.2 Pixel Data in Rendering Commands

This section describes the encoding for images of pixel data for the rendering commands in Section 2.3.6. Pixel data in the rendering commands **glTexImage3D** and **glTexSubImage3D** is described separately in the section "Encoding of Three-Dimensional Images" below.

type	Encoding	Protocol Type	nbytes
GL_UNSIGNED_BYTE	0x1401	CARD8	1
GL_BYTE	0x1400	INT8	1
GL_UNSIGNED_SHORT	0x1403	CARD16	2
GL_SHORT	0x1402	INT16	2
GL_UNSIGNED_INT	0x1405	CARD32	4
GL_INT	0x1404	INT32	4
GL_FLOAT	0x1406	FLOAT32	4
UNSIGNED_BYTE_3_3_2	0x8032	CARD8	1
UNSIGNED_BYTE_2_3_3_REV	0x8363	CARD8	1
UNSIGNED_SHORT_5_6_5	0x8362	CARD16	2
UNSIGNED_SHORT_5_6_5_REV	0x8364	CARD16	2
UNSIGNED_SHORT_4_4_4_4	0x8033	CARD16	2
UNSIGNED_SHORT_4_4_4_4_REV	0x8365	CARD16	2
UNSIGNED_SHORT_5_5_5_1	0x8034	CARD16	2
UNSIGNED_SHORT_1_5_5_5_REV	0x8366	CARD16	2
UNSIGNED_INT_8_8_8_8	0x8035	CARD32	4
UNSIGNED_INT_8_8_8_8_REV	0x8367	CARD32	4
UNSIGNED_INT_10_10_10_2	0x8036	CARD32	4
UNSIGNED_INT_2_10_10_10_REV	0x8368	CARD32	4
GL_BITMAP <sup>I</sup>	0x1A00	n/a	n/a

Table A.1: Bytes per element.

At the API level, the GL allows the user to specify that only a *subimage* within a larger *containing image* be used for rendering; see the **glPixelStorei** and **glPixelStoref** commands in the OpenGL Spec. The GLX client library must send this subimage in some form to the X server, and the server must supply it to the GL; by the time the subimage is rendered, it must be fully unpacked from the containing image.

The GLX protocol has been designed so that the amount of unpacking done by the client is parameterized in the request. In other words, the client can do as much unpacking as it wants, and then tell the server what unpacking remains to be done by sending the appropriate pixel storage parameters along with the image. At one extreme, the client can do all the unpacking needed and only send the subimage. At the other extreme, the client can do none of the unpacking, and send the entire original containing image.

In the general case, the result of the unpacking done by the client is another containing image, possibly smaller than that supplied by the user, and which is put into the rendering request; the encoding is described below.

<sup>1</sup> type GL\_BITMAP is valid only if format is GL\_COLOR\_INDEX or GL\_STENCIL\_-INDEX.

format	Encoding	nelements
GL_RGB	0x1907	3
GL_RGBA	0x1908	4
GL_BGR	0x80E0	3
GL_BGRA	0x80E1	4
GL_COLOR_INDEX <sup>2</sup>	0x1900	1
GL_STENCIL_INDEX <sup>3</sup>	0x1901	1
GL_DEPTH_COMPONENT <sup>3</sup>	0x1902	1
GL_RED	0x1903	1
GL_GREEN	0x1904	1
GL_BLUE	0x1905	1
GL_ALPHA	0x1906	1
GL_LUMINANCE	0x1909	1
GL_LUMINANCE_ALPHA	0x190A	2

Table A.2: Elements per group.

The encoding of the image is described by the attributes in table A.3, which are given in the encoding for each of the commands listed above:

### A.2.1 Encoding For Pixel Types Other Than GL\_BITMAP

Let:

```
nbytes = number of bytes in an element (see Table A.1)
nelements = number of elements in a group (see Table A.2)
ngroups = number of groups in a row
k = number of bytes in a row
```

Then:

$$ngroups = \begin{cases} width, & row\_length = 0 \\ row\_length, & row\_length > 0 \end{cases}$$
 
$$k = \begin{cases} nbytes \cdot nelements \cdot ngroups, & nbytes \ge alignment \\ alignment \cdot \lceil \frac{nbytes \cdot nelements \cdot ngroups}{alignment} \rceil, & nbytes < alignment \end{cases}$$

The  $i^{th}$  group of the  $j^{th}$  row of the subimage begins at byte

$$((j + skip\_rows) \cdot k) + ((i + skip\_pixels) \cdot nelements \cdot nbytes)$$

<sup>&</sup>lt;sup>2</sup> format GL\_COLOR\_INDEX is not valid for **GetTexImage**.

<sup>&</sup>lt;sup>3</sup> formats GL\_STENCIL\_INDEX and GL\_DEPTH\_COMPONENT are not valid for **GetTex-Image**, **TexImagexD** and **TexSubImagexD**.

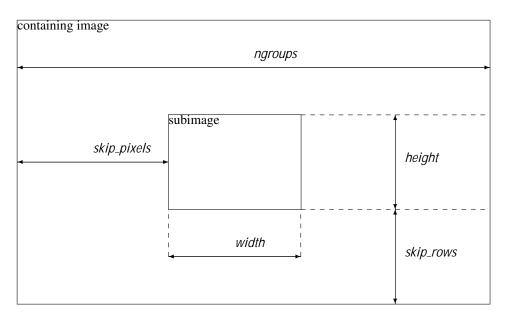


Figure A.1: Pixel Packing Parameters

of the encoding, and occupies  $(nelements \cdot nbytes)$  bytes, for all

 $0 \leq i < width, 0 \leq$ 

 $0917 \quad 9543$ 

 $03\ 543$ 

1001247 152497 123 m[]0d0

Attribute	Description
width	width of the subimage, in groups
height	height of the subimage, in groups
$depth^4$	depth of the subimage, in groups
format	enumerant that specifies the format of groups
type	enumerant that specifies the type of elements
$image\_height^4$	number of rows in each 2D image
$row\_length$	number of groups in a row
$skip\_images^4$	number of unused 2D images preceding the subimage
$skip\_rows$	number of unused rows preceding the subimage
$skip\_pixels$	number of unused groups preceding the subimage
alignment	byte alignment for the start of each row;
	must be 1, 2, 4, or 8
$swap\_bytes$	applicable for pixel types other than GL_BITMAP
$lsb\_first$	applicable only for pixel type GL_BITMAP

Table A.3: Pixel Packing Attributes. width, height, skip\_rows, and skip\_pixels are shown in Figure A.1.

*images* is positive, then the image begins at an offset of (*skip\_images* times the number of elements in one 2-dimensional image) bytes into the encoding; otherwise the image begins at the first byte of the encoding. Then *depth* 2-dimensional images are read, each having a subimage extracted in the manner described above.

### A.2.2 Encoding For Pixel Type GL\_BITMAP

GL\_BITMAP is used only with the pixel formats GL\_COLOR\_INDEX or GL\_STENCIL\_-INDEX.

Let:

$$ngroups =$$
 number of groups in a row   
  $k =$  number of bytes in a row

Then:

$$\begin{split} ngroups &= \left\{ \begin{array}{ll} width, & row\_length = 0 \\ row\_length, & row\_length > 0 \end{array} \right. \\ k &= alignment \cdot \left\lceil \frac{ngroups}{8 \cdot alignment} \right\rceil \end{split}$$

Version 1.3 - 2 June 1999

<sup>&</sup>lt;sup>4</sup> depth, image\_height, and skip\_images are used only for 3D images.

For pixel type GL\_BITMAP, each group contains 1 element, a single bit. In this discussion the least significant bit of a byte is numbered bit 0, and the most significant bit is numbered bit 7.

The  $i^{th}$  bit of the  $j^{th}$  row of the subimage is in byte

$$\frac{((j+skip\_rows)\cdot k) + \lfloor (i+skip\_pixels)}{8\rfloor}$$

of the encoding, and is the  $h^{th}$  bit of that byte, where

$$h = \left\{ \begin{array}{ll} (i + skip\_pixels) \bmod 8, & lsb\_first = \texttt{True} \\ 7 - ((i + skip\_pixels) \bmod 8), & lsb\_first = \texttt{False} \end{array} \right.$$

for all

$$0 \le i \le width, 0 \le j \le height$$

The contents of all other bits in the encoding are undefined.

### **A.3** Pixel Data in Replies

This section describes the encoding for images of pixel data in the replies for the query commands in Section 2.2.2 Pixel data in the replies to **glGetTexImage3D** is sometimes formatted differently; this case is described separately in the section "Encoding of Three-Dimensional Images" below.

Unlike the rendering commands, there is no containing image and subimage for these commands. The image is simply the image returned by the GL, padded to 4 bytes per row. The encoding of the image is described by these attributes, which are given in the description of the encoding for each of the commands listed above:

Attribute	Description
width	width of the image, in groups
height	height of the image, in groups
$depth^5$	depth of the image, in groups
format	enumerant that specifies the format of groups
type	enumerant that specifies the type of elements
$swap\_bytes$	applicable for pixel types other than GL_BITMAP
$lsb\_first$	applicable only for pixel type GL_BITMAP

Table A.4: Encoding For Pixel Types Other Than GL\_BITMAP.

<sup>&</sup>lt;sup>5</sup> *depth* is used only for 3D images.

### A.3.1 Encoding For Pixel Types Other Than GL\_BITMAP

Let:

```
nbytes = number of bytes in an element (see Table A.1)

nelements = number of elements in a group (see Table A.2)

k = number of bytes in a row
```

Then:

$$k = \left\{ \begin{array}{ll} nbytes \cdot nelements \cdot width, & nbytes \ge 4 \\ 4 \cdot \lceil \frac{nbytes \cdot nelements \cdot width}{4} \rceil, & nbytes < 4 \end{array} \right.$$

The  $i^{th}$  group of the  $j^{th}$  row of the image begins at byte

$$(j \cdot k) + (i \cdot nlements \cdot nbytes)$$

of the encoding, and occupies  $(nelements \cdot nbytes)$  bytes, for all

$$0 \le i < width, 0 \le j < height$$

The contents of all other bytes in the encoding are undefined.

Each element has a byte order that is determined by  $swap\_bytes$ : if  $swap\_bytes$  is False, the byte order is the same as the client's native byte order; if True, it is the opposite of the client's native byte order.

#### **Encoding of Three-Dimensional Images**

When the *target* parameter of the query command **glGetTexImage** is GL\_TEXTURE\_-3D, the reply is a three-dimensional image and is formatted differently than the images described above.

A three-dimensional image is arranged as a sequence of *depth* adjacent rectangles. Each rectangle is a 2-dimensional image, whose structure is as described above.

### A.4 Encoding For Pixel Type GL\_BITMAP

GL\_BITMAP is used only with the pixel formats GL\_COLOR\_INDEX or GL\_STENCIL\_-INDEX.

Let:

k = number of bytes in a row

Then:

$$k = 4 \cdot \lceil \frac{width}{32} \rceil$$

For pixel type GL\_BITMAP, each group contains 1 element, a single bit. In this discussion the least significant bit of a byte is numbered bit 0, and the most significant bit is numbered bit 7.

The  $i^{th}$  bit of the  $j^{th}$  row of the image is in byte

$$(j \cdot k) + \lfloor \frac{i}{8} \rfloor$$

of the encoding, and is the  $h^{th}$  bit of that byte, where

$$h = \left\{ \begin{array}{ll} i \bmod 8, & lsb\_first = \mathtt{True} \\ 7 - (i \bmod 8), & lsb\_first = \mathtt{False} \end{array} \right.$$

for all

$$0 \leq i < width, 0 \leq j < height$$

The contents of all other bits in the encoding are undefined.

# Appendix B

# **GLX Versions**

New requests and commands have been added to GLX in versions 1.1, 1.2, and 1.3. Note that GLX 1.3 supports OpenGL versions up to 1.2.1, including the ARB\_multitexture extension if supported by the underlying OpenGL implementation. GLX 1.2 supports OpenGL versions up to 1.1. GLX 1.0 and GLX 1.1 support OpenGL version 1.0.

## **B.1** Requests for GLX commands

The following GLX requests are only available in GLX versions 1.3 and later:

```
glXCreatePbuffer
glXDestroyPbuffer
glXCreatePixmap
glXDestroyPixmap
glXCreateWindow
glXDestroyWindow
glXMakeContextCurrent
glXCreateNewContext
glXGetFBConfigs
glXQueryContext
glXGetDrawableAttributes
glXChangeDrawableAttributes
```

The following GLX requests are only available in GLX versions 1.1 and later:

glXQueryServerString glXClientInfo

### **B.2** Requests for OpenGL Non-rendering Commands

The following OpenGL non-rendering commands are only available in GLX versions 1.3 and later:

GetColorTableParameterfv
GetColorTableParameteriv
GetColorTable
GetConvolutionFilter
GetConvolutionParameterfv
GetConvolutionParameteriv
GetHistogramParameteriv
GetHistogramParameteriv
GetHistogram
GetMinmaxParameteriv
GetMinmaxParameteriv
GetMinmax
GetSeparableFilter

The following OpenGL non-rendering commands are only available in GLX versions 1.2 and later:

AreTexturesResident DeleteTextures GenTextures IsTexture

## **B.3** Protocol for OpenGL rendering commands

The following OpenGL rendering commands are only available in GLX versions 1.3 and later:

ActiveTextureARB

BlendColor

BlendEquation

ColorSubTable

ColorTableParameterfy

ColorTableParameteriv

ColorTable

ConvolutionFilter1D

ConvolutionFilter2D

ConvolutionParameterfy

ConvolutionParameterf

ConvolutionParameteriv

ConvolutionParameteri

Copy Color Sub Table

CopyColorTable

CopyConvolutionFilter1D

CopyConvolutionFilter2D

CopyTexSubImage3D

Histogram

Minmax

MultiTexCoord[1234][sifd]ARB

MultiTexCoord[1234][sifd]vARB

ResetHistogram

ResetMinmax

SeparableFilter2D

TexImage3D

TexSubImage3D

The following OpenGL rendering commands are only available in GLX versions 1.2 and later:

**BindTexture** 

CopyTexImage1D

CopyTexImage2D

CopyTexSubImage1D

CopyTexSubImage2D

**DrawArrays** 

PolygonOffset

**PrioritizeTextures** 

TexSubImage1D

TexSubImage2D

# **Appendix C**

# References

- The OpenGL <sup>®</sup> Graphics System: A Specification, Version 1.3, Segal, Mark, and Akeley, Kurt.
- $\bullet$  OpenGL  $^{\circledR}$  Graphics with the X Window System  $^{\circledR}$  , Version 1.3, Karlton, Phil.

# **Index**

Accum, 78	Color3sv, 82
ActiveTextureARB, 78, 156	Color3ubv, 82
AlphaFunc, 78	Color3uiv, 82
AreTexturesResident, 41, 156	Color3usv, 82
	Color4by, 83
BadAccess, 16, 18, 19, 34	Color4dv, 83
BadAlloc, 14-17, 24, 30-32, 34-36, 39,	Color4fv, 83
40, 76	Color4iv, 83
BadDrawable, 38	Color4sv, 84
BadFont, 23	Color4uby, 84
BadLength, 74, 76	Color4uiv, 84
BadMatch, 14–16, 18, 19, 24, 30–32, 34–	Color4usy, 84
36, 39, 40	COLOR_ARRAY, 128
BadPixmap, 24, 30, 31	ColorMask, 85
BadValue, 13–15, 18, 24, 26, 27, 29, 32,	ColorMaterial, 85
38, 39	ColorSubTable, 75, 136, 147, 157
BadWindow, 39	ColorTable, 75, 135, 147, 157
Begin, 78	ColorTableParameterfy, 85, 157
BindTexture, 79, 157	ColorTableParameteriv, 85, 157
Bitmap, 23, 75, 134, 147	ConvolutionFilter1D, 137, 157
BlendColor, 79, 157	ConvolutionFilter2D, 137, 157
BlendEquation, 79, 157	ConvolutionFilterxD, 75, 147
BlendFunc, 79	ConvolutionParameterf, 86, 157
	ConvolutionParameterfy, 86, 157
CallList, 79	ConvolutionParameteri, 86, 157
CallLists, 75, 126	ConvolutionParameteriv, 86, 157
Clear, 79	CopyColorSubTable, 87, 157
ClearAccum, 80	CopyColorTable, 87, 157
ClearColor, 80	CopyConvolutionFilter1D, 87, 157
ClearDepth, 80	CopyConvolutionFilter2D, 88, 157
ClearIndex, 80	CopyPixels, 88
ClearStencil, 80	CopyTexImage1D, 157
ClipPlane, 81	CopyTexImage2D, 88, 157
Color3bv, 81	CopyTexSubImage1D, 88, 157
Color3dv, 81	CopyTexSubImage2D, 89, 157
Color3fv, 81	CopyTexSubImage3D, 89, 157
Color3iv, 82	CullFace, 89
	C

DeleteLists, 41	GetError, 48
DeleteTextures, 42, 156	GetFloaty, 48
DepthFunc, 90	GetHistogram, 69, 147, 156
DepthMask, 90	GetHistogramParameterfy, 49, 156
DepthRange, 90	GetHistogramParameteriv, 50, 156
DOUBLE, 128	GetIntegery, 50
DrawArrays, 75, 127, 157	GetLightfy, 51
DrawBuffer, 90	GetLightiv, 52
DrawPixels, 75, 139, 147	GetMapdy, 52
	GetMapfy, 53
EDGE_FLAG_ARRAY, 128	GetMapiv, 54
EdgeFlagv, 90	GetMaterialfy, 54
End, 91	GetMaterialiv, 55
EndList, 42	GetMinmax, 70, 147, 156
ENUM, 3	GetMinmaxParameterfy, 56, 156
EvalCoord1dv, 91	GetMinmaxParameteriy, 56, 156
EvalCoord1fv, 91	GetPixelMapfy, 57
EvalCoord2dv, 91	GetPixelMapuiv, 58
EvalCoord2fv, 91	GetPixelMapusy, 58
EvalMesh1, 91	GetPolygonStipple, 71, 147
EvalMesh2, 92	GetSeparableFilter, 71, 147, 156
EvalPoint1, 92	GetString, 14, 59
EvalPoint2, 92	GetTexEnvfv, 59
	GetTexEnviv, 60
FeedbackBuffer, 42	GetTexGendy, 61
Finish, 42	GetTexGenfy, 61
FLOAT, 128	GetTexGeniv, 62
Flush, 43	GetTexImage, 72, 147, 149
Fogf, 92	GetTexLevelParameterfy, 62
Fogfv, 92	GetTexLevelParameteriv, 63
Fogi, 93	GetTexParameterfy, 64
Fogiv, 93	GetTexParameteriv, 64
FrontFace, 93	GL_2_BYTES, 127
Frustum, 94	GL_3_BYTES, 127
~	GL_4_BYTES, 127
GenLists, 43	GL_ACCUM_BUFFER_BIT, 18
GenTextures, 43, 156	GL_ALL_ATTRIB_BITS, 19
GetBooleany, 44	GL_ALPHA, 149
GetClipPlane, 44	GL_AMBIENT, 96, 101
GetColorTable, 68, 147, 156	GL_AMBIENT_AND_DIFFUSE, 101
GetColorTableParameterfy, 45, 156	GL_BGR, 149
GetColorTableParameteriv, 46, 156	GL_BGRA, 149
GetConvolutionFilter, 69, 147, 156	GL_BITMAP, 71, 135, 140, 148, 149,
GetConvolutionParameterfy, 46, 156	151–154
GetConvolutionParameteriv, 47, 156	GL_BLUE, 149
GetDoublev, 48	GL_BYTE, 127, 128, 148
	, , , , -

```
GL_COLOR_BUFFER_BIT, 18
GL_COLOR_INDEX, 71, 135, 140, 148,
       149, 151, 153
GL_COLOR_INDEXES, 101
GL_COLOR_TABLE_BIAS, 85
GL_COLOR_TABLE_SCALE, 85
GL_CONSTANT_ATTENUATION, 96,
GL_CONVOLUTION_BORDER_-
       MODE, 86, 87
GL_CONVOLUTION_FILTER_BIAS,
       86, 87
GL_CONVO-
       LUTION_FILTER_SCALE, 86,
       87
GL_CONVOLUTION_FORMAT, 86, 87
GL_CONVOLUTION_HEIGHT, 86, 87
GL_CONVOLUTION_WIDTH, 86, 87
GL_CURRENT_BIT, 18
GL_DEPTH_BUFFER_BIT, 18
GL_DEPTH_COMPONENT, 149
GL_DIFFUSE, 96, 97, 101
GL_EMISSION, 101
GL_ENABLE_BIT, 18
GL_EVAL_BIT, 19
GL_EYE_PLANE, 120, 121
GL_FEEDBACK, 35, 67
GL_FLOAT, 127, 148
GL_FOG_BIT, 18
GL_FOG_COLOR, 93
GL_FOG_DENSITY, 93
GL_FOG_END, 93
GL_FOG_INDEX, 93
GL_{-}
```

FOG

GL_SPOT_EXPONENT, 96, 97	GLXBadCurrentDrawable, 34, 35
GL_STENCIL_BUFFER_BIT, 18	GLXBadCurrentWindow, 16–19, 21–23
GL_STENCIL_INDEX, 148, 149, 151,	GLXBadDrawable, 16, 22, 34, 35, 38, 39
153	GLXBadFBConfig, 29–32, 36, 39, 40
GL_TEXTURE_3D, 153	GLXBadLargeRequest, 76
GL_TEXTURE_BIT, 19	GLXBadPbuffer, 37
GL_TEXTURE_BORDER_COLOR, 122	GLXBadPixmap, 25, 31
GL_TEXTURE_ENV_COLOR, 119, 120	GLXBadRenderRequest, 74
GL_TEXTURE_ENV_MODE, 119, 120	GLXBadWindow, 34, 35, 40
GL_TEXTURE_GEN_MODE, 120, 121	glXChangeDrawableAttributes, 20, 38,
GL_TEXTURE_MAG_FILTER, 122	155
GL_TEXTURE_MIN_FILTER, 122	glXChooseFBConfig, 29
GL_TEXTURE_WRAP_S, 122	glXClientInfo, 13, 156
GL_TEXTURE_WRAP_T, 122	glXCopyContext, 10, 18, 20, 21
GL_TRANSFORM_BIT, 18	glXCreateContext, 14, 20
GL_UNSIGNED_BYTE, 127, 148	glXCreateGLXPixmap, 20, 24, 31
GL_UNSIGNED_INT, 127, 148	gIXCreateNewContext, 20, 32, 155
GL_UNSIGNED_SHORT, 127, 148	glXCreatePbuffer, 20, 36, 155
GL_VIEWPORT_BIT, 18	glXCreatePixmap, 20, 30, 31, 155
glGetTexImage, 153	glXCreateWindow, 20, 39, 155
glGetTexImage3D, 152	glXDestroyContext, 15, 20
glPixelStoref, 148	glXDestroyGLXPixmap, 20, 25, 31
glPixelStorei, 148	gIXDestroyPbuffer, 20, 37, 155
glPushAttrib, 19	glXDestroyPixmap, 20, 31, 155
glTexImage3D, 147, 150	glXDestroyWindow, 20, 40, 155
glTexSubImage3D, 147, 150	GLXFBConfig, 36
GLX_DAMAGED, 9	glXGetDrawableAttributes, 20, 37, 155
GLX_EVENT_MASK, 38, 39	glXGetFBConfigs, 20, 29, 155
GLX_EXTENSIONS, 13	glXGetSelectedEvent, 37
GLX_FBCONFIG_ID, 33, 38	glXGetVisualConfigs, 20, 25
GLX_HEIGHT, 38	glXIsDirect, 17, 20
GLX_LARGEST_PBUFFER, 38	glXMakeContextCurrent, 4, 10, 20, 34,
GLX_PBUFFER, 9	155
GLX_PBUFFER_CLOBER_MASK, 39	glXMakeCurrent, 4, 10, 16, 20
GLX_PRESERVED_CONTENTS, 38	GLXPbuffer, 35, 36
GLX_RENDER_TYPE, 33	GLXPixmap, 35
GLX_SAVED, 9	glXQueryContext, 20, 33, 155
GLX_SCREEN, 33	glXQueryDrawable, 37
GLX_VENDOR, 13	glXQueryExtensionsString, 20
	glXQueryServerString, 12, 20, 156
GLX_VERSION, 13 GLX_WIDTH, 38	• - •
	glXQueryVersion, 11, 20
GLX_WINDOW, 9 GLYBodContext 14 10 32 34	glXRender, 6, 10, 11, 20, 41, 73–78, 126,
GLXBadContext, 14–19, 32–34 GLXBadContextState, 16, 23, 34, 35	131, 134
	glXRenderLarge, 6, 10, 11, 20, 73, 75–77,
GLXBadContextTag, 10, 16, 18, 19, 21–	126, 127, 129–146
23, 41, 74, 76	glXSelectEvent, 38

glXSwapBuffers, 10, 20, 22, 24, 30, 36	MapGrid1f, 99
GLXUnsupportedPrivateRequest, 27, 28	MapGrid2d, 100
glXUseXFont, 4, 10, 21, 23	MapGrid2f, 100
glXVendorPrivate, 27	Materialf, 100
glXVendorPrivateWithReply, 28	Materialfy, 100
glXWaitGL, 10, 19–21	Materiali, 101
glXWaitX, 10, 20, 21	Materialiv, 101
GLXWindow, 35	MatrixMode, 101
0211 ( mgc	Minmax, 102, 157
Hint, 94	MultiTexCoord1dvARB, 102
Histogram, 94, 157	MultiTexCoord1fvARB, 102
111010814111, 71, 107	
INDEX_ARRAY, 128	MultiTexCoord1ivARB, 102
Indexdy, 94	MultiTexCoord1svARB, 102
Indexfev, 94	MultiTexCoord2dvARB, 103
Indexiv, 95	MultiTexCoord2fvARB, 103
	MultiTexCoord2ivARB, 103
IndexMask, 95	MultiTexCoord2svARB, 103
Indexsv, 95	MultiTexCoord3dvARB, 103
Indexuby, 95	MultiTexCoord3fvARB, 104
InitNames, 95	MultiTexCoord3ivARB, 104
INT, 128	MultiTexCoord3svARB, 104
IsList, 65	MultiTexCoord4dvARB, 104
IsTexture, 65, 156	MultiTexCoord4fvARB, 105
	MultiTexCoord4ivARB, 105
Lightf, 95	MultiTexCoord4svARB, 105
Lightfy, 96	MultiTexCoord[1234][sifd]ARB, 157
Lighti, 96	MultiTexCoord[1234][sifd]vARB, 157
Lightiv, 96	MultMatrixd, 105
LightModelf, 97	
LightModelfy, 97	MultMatrixf, 106
LightModeli, 97	NewList, 66
LightModeliv, 97	
LineStipple, 98	None, 35
LineWidth, 98	Normal3by, 106
ListBase, 98	Normal3dv, 106
LoadIdentity, 98	Normal3fv, 106
LoadMatrixd, 98	Normal3iv, 106
	Normal3sv, 107
LoadMatrixf, 99	NORMAL_ARRAY, 128
LoadName, 99	
LogicOp, 99	Ortho, 107
Map1d, 75, 131	PassThrough, 107
Map1f, 75, 131, 132	PixelMapfy, 75, 129
Map2d, 75, 131–133	PixelMapuiv, 75, 130
Map2f, 75, 131–133	•
MapGrid1d, 99	PixelMapusy, 130
141apO11u1u, 77	PixelStoref, 66

PixelStorei, 66	SelectBuffer, 67
PixelTransferf, 107	SeparableFilter2D, 75, 138, 147, 157
PixelTransferi, 108	ShadeModel, 115
PixelZoom, 108	SHORT, 128
PixMapusy, 75	StencilFunc, 115
•	
PointSize, 108	Stencil Mask, 115
PolygonMode, 108	StencilOp, 115
PolygonOffset, 108, 157	TexCoord1dv, 116
PolygonStipple, 75, 140, 147	TexCoord1fv, 116
PopMatrix, 100	TexCoord1iv, 116
PopMatrix, 109	TexCoord1sv, 116
PopName, 109	TexCoord2dv, 116
PrioritizeTextures, 75, 109, 131, 157	TexCoord2fv, 117
PushAttrib, 109	TexCoord2iv, 117
PushName 100	TexCoord2sv, 117
PushName, 109	TexCoord3dv, 117
QueryExtension, 3	TexCoord3fv, 117
Query Entension, 5	TexCoord3iv, 118
RasterPos2dv, 110	TexCoord3sv, 118
RasterPos2fv, 110	TexCoord4dv, 118
RasterPos2iv, 110	TexCoord4fv, 118
RasterPos2sv, 110	TexCoord4iv, 118
RasterPos3dv, 110	TexCoord4sv, 119
RasterPos3fv, 111	TexEnvf, 119
RasterPos3iv, 111	TexEnvfv, 119
RasterPos3sv, 111	TexEnvi, 119
RasterPos4dv, 111	TexEnviv, 120
RasterPos4fv, 112	TexGend, 120
RasterPos4iv, 112	TexGendy, 120
RasterPos4sv, 112	TexGenf, 120
ReadBuffer, 112	TexGenfv, 121
ReadPixels, 73, 147	TexGeni, 121
Rectdy, 112	TexGeniv, 121
Rectfy, 113	TexImage1D, 140
Rectiv, 113	TexImage2D, 141
Rectsv, 113	TexImage3D, 142, 157
RenderMode, 67	TexImage xD, 75, 147, 149
ResetHistogram, 113, 157	TexParameterf, 121
ResetMinmax, 114, 157	TexParameterfy, 122
Rotated, 114	TexParameteri, 122
Rotatef, 114	TexParameteriv, 122
C1-J 114	TexSubImage1D, 143, 157
Scaled, 114	TexSubImage2D, 144, 157
Scalef, 114	TexSubImage3D, 145, 157
Scissor, 115	TexSubImage <i>X</i> D, 75, 147, 149

```
TEXTURE_COORD_ARRAY, 128
Translated, 123
Translatef, 123
UNSIGNED_BYTE, 128
UNSIGNED_BYTE_2_3_3_REV, 148
UNSIGNED_BYTE_3_3_2, 148
UNSIGNED_INT, 128
UNSIGNED_INT_10_10_10_2, 148
UNSIGNED_INT_2_10_10_10_REV, 148
UNSIGNED_INT_8_8_8_8, 148
UNSIGNED_INT_8_8_8_8_REV, 148
UNSIGNED_SHORT, 128
UNSIGNED_SHORT_1_5_5_5_REV, 148
UNSIGNED_SHORT_4_4_4_4, 148
UNSIGNED_SHORT_4_4_4_4_REV, 148
UNSIGNED_SHORT_5_5_5_1, 148
UNSIGNED_SHORT_5_6_5, 148
UNSIGNED_SHORT_5_6_5_REV, 148
Vertex2dv, 123
Vertex2fv, 123
Vertex2iv, 123
Vertex2sv, 124
Vertex3dv, 124
Vertex3fv, 124
Vertex3iv, 124
Vertex3sv, 124
Vertex4dv, 125
Vertex4fv, 125
Vertex4iv, 125
Vertex4sv, 125
VERTEX_ARRAY, 128
Viewport, 126
Visual, 2
xImage1D, 126
```