Contents

List of Figures

List of Tables

Chapter 1

Introduction

This document describes the OpenGL graphics system: what it is, how it acts, and

Letter	Corresponding

GL Type	Minimum Bit Width	Description
`boolean`	1	

back from the framebuffer or copied from one portion of the framebuffer to another.

Error	Description	Offending command ignored?
INVALID_ENUM	enum argument out of range	Yes
INVALID_VALUE		

images are mapped onto a primitive. The number of texture units supported is implementation dependent but must be at least two. The number of texture units supported can be queried with the state MAX_TEXTURE_UNITS

Figure 2.3. Primitive assembly and processing.

There is no limit on the number of vertices that may be specified between a **Begin** and an **End**.

Points. A series of individual points may be specified by calling **Begin** with an argument value of `POINTS`. No special state need be kept between **Begin** and **End** in this case, since each point is independent of previous and following points.

Line Strips. A series of one or more connected line segments is specified by enclosing a series of two or more endpoints within a **Begin**/**End** pair when **Begin** is called with `LINE_STRIP`. In this case, the first vertex specifies the first segment's start point while the second vertex specifies the first segment's endpoint and the second segment's start point 100.2 0 Td0(prdFeneral)-305(the)]TJ F44310.909 Tf 2903199 0 md[(is]TJ/F34 10.909

gins an edge. If the edge flag bit is

call to

the three coordinates of the current normal, one floating-point number to store the current fog coordinate, four floating-point values to store the current RGBA *void***VertexPointer**

2.8 Vertex Arrays

The vertex specification commands described in section 2.7 accept data in almost any format, but their use requires many command executions to specify even simple geometry. Vertex data may also be placed into arrays that are stored in the client's address space. Blocks of data in these arrays may then be used to specify multiple geometric primitives through the execution of a single GL command. The client may specify up to seven plus the value of `MAX_TEXTURE_UNITS` arrays: one each to store vertex coordinates, normals, colors, secondary colors, color indices, fog coordinates, one or more texture coordinate sets, and edge flags. The commands

`void` **VertexPointer**(`int` *size*, `enum` *type*, `sizei` *stride*,
`void` **pointer*);

`void` **NormalPointer**(`enum` *type*, `sizei` *stride*,
`void` **pointer*);

`void` **ColorPointer**(`int` *size*, `enum` *type*, `sizei` *stride*,
`void` **pointer*);

`void` **SecondaryA7 38(0 g 0** *size*, `enum` *type*, `sizei` *stride* [illegible]

Command	Sizes	Types
VertexPointer	2,3,4	`short`, `int`, `float`, `double`
NormalPointer	3	`byte`, `short`, `int`, `float`, `double`

DrawArrays (*mode*, *first*, *count*);

is the same as the effect of the command sequence

```
if (mode or count is invalid )
   generate appropriate error
else {
   int i;
   Begin(mode);
   for (i=0; i < count ; i++)
      ArrayElement(first+ i);
   End();
}
```

format

Name	Type	Initial Value	Legal Values

returns *n* previously unused buffer object names in *buffers*. These names are marked as used, for the purposes of **GenBuffers** only, but they acquire buffer state only when they are first bound, just as if they were unused.

While a buffer object is bound, any GL operations on that object affect any other bindings of that object. If a buffer object is delebct.

Name	

relinquished by calling

`boolean` **UnmapBuffer**(`enum` *target*)`;`

computed by subtracting a null pointer from the pointer value, where both pointers are treated as pointers to basic machine units.

It is acceptable for vertex, variant, or attrib arrays to be sourced from any combination of client memory and various buffer objects during a single rendering operation.

Attempts to source data from a currently mapped buffer object will generate an INVALID_

2.10 Rectangles

There is a set of GL commands to support efficient specification of rectangles as

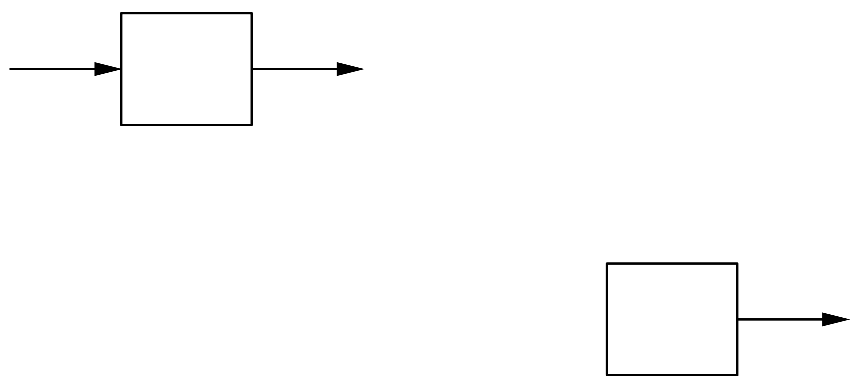

Figure 2.6. Vertex transformation sequence.

2.11.2 Matrices

The projection matrix and model-view matrix are set and modified with a variety of commands. The affected matrix is determined by the current matrix mode. The current matrix mode is set with

`void` **MatrixMode**(`enum` *mode*);

which takes one of the pre-defined constants

θ gives an angle of rotation in degrees; the coordinates of a vector

`void` **ActiveTexture**(`enum` *texture*);

a pointer to an array of values that specify texture generation parameters. *pname* must be one of the three symbolic constants TEXTURE_GEN_MODE, OBJECT_PLANE, or EYE_PLANE. If *pname* is TEXTURE_GEN_MODE, then either *params* points to or *param* is an integer that is one of the symbolic constants OBJECT_LINEAR, EYE_LINEAR, SPHERE_MAP, REFLECTION_MAP, or NORMAL_MAP.

If TEXTURE_GEN_MODE indicates OBJECT

s coordinate is $s = r$

This clipping produces a value, $0 \leq t \leq 1$

WindowPos3 takes three values indicating x, y and z, while **WindowPos2** takes two values indicating x and y with z implicitly set to 0. The current raster position, (x

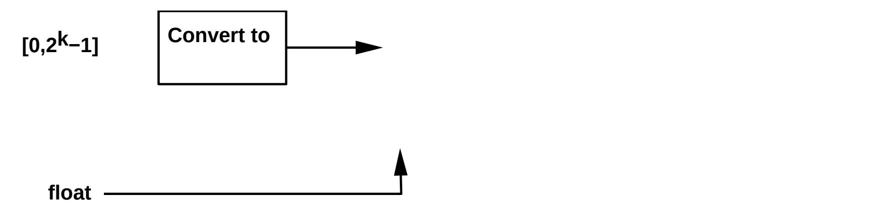

Figure 2.8. Processing of RGBA colors. The heavy dotted lines indicate both primary and secondary vertex colors, which are processed in the same fashion. See Table 2.9 for the interpretation of k.

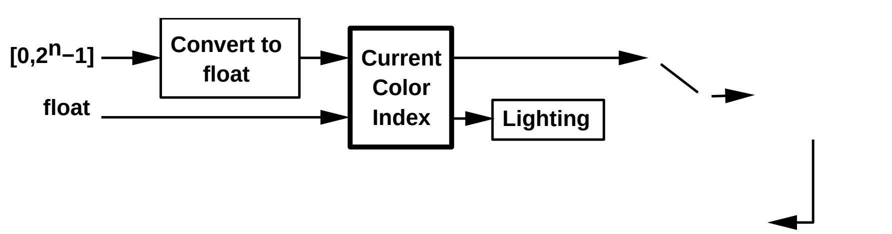

Figure 2.9. Processing of color indices. n is the number of bits in a color index.

h

```
void
```

The values

k $\{0, 1, \dots, 2^m - 1\}$, as k (e.g. 1.0 is represented in binary as a string of

Chapter 3

Rasterization

Rasterization is the process by which a primitive is converted to a two-dimensional

A GL implementation may use other methods to perform antialiasing, subject to the following conditions:

1. If f_1 and f_2 are two fragments, and the portion of f_1 covered by some primitive is a subset of the corresponding portion of f_2 t27.2727it J/F3applics tgfolg5imTd[(84p565dtioclud3dime5

base GL may result in a higher quality image. This mechanism is designed to allow multisample and smooth antialiasing techniques to be alternated during the rendering of a single scene.

If the value of SAMPLE

void **glPointParameter{if}v**(enum *pname*, const float **params*);

If *pname* is POINT

Figure 3.3. Rasterization of antialiased wide points. The black dot indicates the point to be rasterized. The shaded region has the specified width. The X marks indicate those fragment centers produced by rasterization. A fragment's computed coverage value is based on the portion of the shaded region that covers the corresponding fragment square. Solid lines lie on -34eg6hat

width is used instead. The range of supported widths and the width of evenly-

3.4.1 Basic Line Segment Rasterization

Line segment rasterization begins by characterizing the segment as either *x-major* or *y-major*. x-major line segments have slope in the closed interval [§

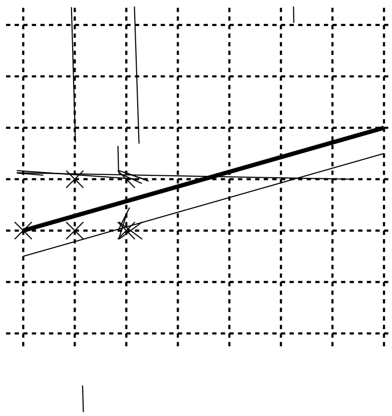

Figure 3.6. The region used in rasterizing and finding corresponding coverage values for an antialiased line segment (an x-major line segment is shown).

incident on the starting endpoint of the segment. Each of these rectangles is ei-

CULL_FACE. Front facing polygons are rasterized if either culling is disabled or the this may yield acceptable results for color values (it *must* be used for depth values), but will normally lead to unacceptable distortion effects if used for texture

Polygon stippling may be enabled or disabled with **Enable** or **Disable** using

Boolean state values POLYGON_OFFSET_POINT, POLYGON_OFFSET_LINE, and POLYGON_OFFSET_FILL determine whether o is applied during the rasterization of polygons in POINT, LINE, and FILL modes. These boolean state values are enabled and disabled as argument values to the commands **Enable** and **Disable**. If POLYGON_OFFSET_POINT

Table Name	Type
COLOR_TABLE	regular

RGBA, with zero-sized components). The initial value of the scale parameters is (1,1,1,1) and the initial value of the bias parameters is (0,0,0,0).

meanings, as the equivalent arguments of **ConvolutionFilter2D**. *format* is taken to be RGBA.

The command

type Parameter Token Name	GL Data Type	Number of Components	Matching Pixel Formats
UNSIGNED_			

Base Internal Format	

`POST_CONVOLUTION_COLOR_TABLE`. In all other respects, operation is identical to color table lookup, as defined earlier in section 3.6.5.

Histogram

This step applies only to RGBA component groups. Histogram operation is enabled or disabled by calling **Enable** or **Disable** with the symbolic constant `HISTOGRAM`.

If the width of the table is non-zero, then indices R_{iR}

group component values that are outside the representable range.

If the **Minmax** *sink* parameter is FALSE, minmax operation has no effect on the stream of pixel groups being processed. Otherwise, all RGBA pixel groups are discarded immediately after the minmax operation is completed. No pixel fragments are generated, no change is made to texture memory contents, and no pixel values are returned. However, texture object state is modified whether or not pixel groups are discarded.

3.6.6 Pixel Rectangle Multisample Rasterization

If MULTISAMPLE is enabled, and the value of SAMPLE

Sized Internal Format	Base Internal Format	R bits	G bits	B bits	A	L	I	D

Compressed Internal Format	Base Internal Format

The *level* argument to **TexImage3D** is an integer *level-of-detail*

1534

`void` **CopyTexImage1D**(`enum` *target*

$$x + w > w_s - b_s$$

$$y < -b_s$$

$$y + h > h_s - b_s$$

$$z < -b_s$$

$$z + d > d_s - b_s$$

(Recall that d

The *xoffset* argument of **TexSubImage1D** and **CopyTexSubImage1D** specifies the left texel coordinate of a *width*

```
void
```

and `TEXTURE_COMPRESSED_IMAGE_SIZE` for image level *level* in effect at the time of the **GetCompressedTexImage** call returning *data*.

This guarantee applies not just to images returned by **GetCompressedTexImage**,

target is the target, either `TEXTURE_1D`, `TEXTURE_2D`, `TEXTURE_3D`, or `TEXTURE_CUBE_MAP`. *pname* is a symbolic constant indicating the parameter to be set; the possible constants and corresponding parameters are summarized in table 3.19. In the first form of the command, *param* is a value to which to set a single-valued parameter; in the second form of the command, *params* is an array of parameters whose type depends on the parameter being set. If the values for `TEXTURE_BORDER_COLOR`

Name	Type	Legal Values
TEXTURE_WRAP_S	integer	CLAMP, CLAMP_TO_EDGE, REPEAT, CLAMP

$$\text{mirror}(f) =$$

Let $s(x, y)$ be the function that associates an s texture coordinate with each set of window coordinates (x, y) that lie within a primitive; define $t(x, y)$ and $r(x, y)$ analogously. Let u

Mipmapping

TEXTURE_MIN_FILTER values NEAREST_MIPMAP_NEAREST, NEAREST_MIPMAP_

3.8.9 Texture Magnification

When λ indicates magnification, the value assigned to TEXTURE_MAG_FILTER determines how the texture value is obtained. There are two possible values for TEXTURE_MAG_FILTER: NEAREST and LINEAR. NEAREST behaves exactly as NEAREST for TEXTURE_MIN_FILTER (equations

•

age2D is executed with the *target* field specified as

FALSE is returned, the error INVALID_VALUE is generated, and the contents of

If a texture unit is disabled or has an invalid or incomplete texture (as defined in section 3.8.10) bound to it, then blending is disabled for that texture unit. If the

If the fog source, as defined below, is FRAGMENT_DEPTH, then c is the eye-coordinate distance from the eye, $(0, 0, 0, 1)$

the same number of bits to the right of the binary point as i_r

Chapter 4

Figure 4.1. Per-fragment operations.

4.1 Per-Fragment Operations

4.1.2 Scissor Test

The scissor test determines if (x_w, y_w) lies within the scissor rectangle defined by four values. These values are set with

`void` **Scissor**(

Next, if SAMPLE_ALPHA_TO_ONE

The required state consists of the floating-point reference value, an eight-

4.1.10 Logical Operation

Finally, a logical operation is applied between the incoming fragment's color or index values and the color or index values stored at the corresponding location in the framebuffer. The result replaces the values in the framebuffer at the fragment's (x_w, y_w) coordinates. The logical operation on color indices is enabled or disabled with **Enable** or **Disable** using the symbolic constant INDEX_LOGIC

are written or not (a value of TRUE means that the corresponding value is written).

enabled for color writing, the depth buffer, the stencil buffer, and the accumulation buffer (see below), respectively. The value to which each buffer is cleared depends

The `MULT` operation multiplies each R, G, B, and A in the accumulation buffer by *value*

through `AUX`n. `FRONT` and `LEFT` refer to the front left buffer, `BACK` refers to the back left buffer, and `RIGHT` refers to the front right buffer. The other constants correspond directly to the buffers that they name. If the requested buffer is missing, then the error `INVALID_OPERATION` is generated. The initial setting for **ReadBuffer** is `FRONT` if there is no back buffer and

4.3.3 Copying Pixels

CopyPixels transfers a rectangle of pixel values from one region of the framebuffer

Figure 5.1. Map Evaluation.

`void` **Map2**

void **MapGrid1{fd}**(int

If *mode*

consists of one or two orders, an appropriately sized array of control points, and a generates

Hit records are placed in the selection array by maintaining a pointer into that array. When selection mode is entered, the pointer is initialized to the beginning of the array. Each time a hit record is copied, the pointer is updated to point at the array element after the one into which the topmost element of the name stack

Type	coordinates	

that points to an array of offsets. Each offset is constructed as determined by *lists*

5.5 Flush and Finish

The command

```
void Flush( void );
```

indicates that all commands that have previously been sent to the GL must complete in finite time.

The command

```
void Finish( void );
```

forces all previous GL commands to complete. **Finish** does not return until all effects from previously issued commands on GL client and server state and the framebuffer are fully realized.

5.6 Hints

Certain aspects of GL behavior, when there is room for variation, may be controlled with hints. A hint is specified using

```
void Hint( enum
```

be used if the compression results are to be retrieved by **GetCompressedTexImage** (section 6.1.4) for reuse.

The interpretation of hints is implementation dependent. An implementation may ignore them entirely.

The initial value of all hints is `DONT_CARE`.

Chapter 6

State and State Requests

The state required to describe the GL machine is enumerated in section 6.2. Most

6.1.2 Data Conversions

If a **Get** command is issued that returns value types different from the type of the value being obtained, a type conversion is performed. If **GetBooleanv** is called, a floating-point or integer value converts to FALSE if and only if it is zero (otherwise it converts to TRUE). If **GetIntegerv** (or any of the **Get** commands below) is called, a boolean value is interpreted as either 1 or 0, and a floating-point value is rounded to the nearest integer, unless the value is an RGBA color component, a **DepthRange** value, a Ae

6.1.QUERYING GL STATE

TEXTURE

For **GetPixelMap**, the *map* must be a map name from Table 3.3. For **GetMap**, *map* must be one of the map types described in section 5.1, and *value* must be one of `ORDER`, `COEFF`, or `DOMAIN`.

6.1.4 Texture

6.1.9 Histogram Query

The current contents of the histogram table are queried using

void **GetHistogram**(enum *target*, boolean *reset*,
enum *format*, enum *type*, void* *values*);

target must be HISTOGRAM. *type* and *format* accept the same values as do the corresponding parameters of **GetTexImage** *target* must be `MINMAX`. *type*

returns a pointer to a static string describing some aspect of the current GL connection. The possible values for *name* are VENDOR, RENDERER, VERSION, and EXTENSIONSRENDERER

An error is generated if **GetBufferSubData** is executed for a buffer object that is currently mapped.

reset the values of those state variables that were saved with the last corresponding **PushAttrib** or

6.2 State Tables

The tables on the following pages indicate which state variables are obtained with what commands. State variables that can be obtained using any of **GetBooleanv**,

Get value	Type	Get Cmnd	Initial Value	Description	Sec.	Attribute
–	Z_{11}	–	0	When = 0, indicates **begin/end** object	2.6.1	–
–	V	–	–	Previous vertex in **Begin/End line**	2.6.1	–
–	B	–	–	Indicates if *line-vertex* is the first	2.6.1	–
–	V	–	–	First vertex of a **Begin/End line loop**	2.6.1	

Get value	Type	Get Cmnd	Initial Value	Description	Sec.	Attribute

Get value	Type	Get Cmnd	Initial Value	Description	Sec.	Attribute
TEXTURE_COORD_ARRAY	$2 \times B$	**IsEnabled**	*False*	Texture coordinate array enable	2.8	vertex-array
TEXTURE_COORD_ARRAY_SIZE	$2 \times Z^+$	**GetIntegerv**	4	Coordinates per element	2.8	vertex-array
TEXTURE_COORD_ARRAY_TYPE	$2 \times Z_4$	**GetIntegerv**	FLOAT	Type of texture coordinates	2.8	

Get value	Type	Get Cmnd	Initial Value	Description	Sec.	Attribute

Get value	Type	Get Cmnd	Initial Value	Description	Sec.	Attribute
AMBIENT	$8 \times C$	**GetLightfv**	(0.0,0.0,0.0,1.0)	Ambient intensity of light i		

Get value	Type	Get Cmnd	Initial Value	Description	Sec.	Attribute

Get value	Type	Get Cmnd	Initial Value	Description	Sec.	Attribute

Get value	Type	Get Cmnd	Initial Value	Description	Sec.	Attribute

Get value

Type

Type

Get value

Get value	Type	Get Cmnd	Minimum Value	Description	Sec.	Attribute
MAX_LIGHTS	Z^+	**GetIntegerv**	8			

Get value	Type	Get Cmnd

Get value	Type	Get Cmnd	Initial Value	Description	Sec.	Attribute
x_BITS	Z^+					

Get value	Type	Get Cmnd	Initial Value	Description	Sec.	Attribute

Appendix A

Invariance

The OpenGL specification is not pixel exact. It therefore does not guarantee an exact match between images produced by different GL implementations. However, the specification does specify exact matches, in some cases, for images produced by the same implementation. The purpose of this appendix is to identify and provide justification for those cases that require exact matches.

A.1 Repeatability

The obvious and most fundamental case is repeated issuance of a series of GL commands. For any given GL and framebuffer state *vector*, and for any GL command, the resulting GL and framebuffer state must be identical whenever the command is executed on that initial GL and framebuffer state.

One purpose of repeatability is avoidance of visual artifacts when a double-buffered scene is redrawn. If rendering is not repeatable, swaderapping between twadero

A.2 Multi-pass Algorithms

Invariance is necessary for a whole set of useful multi-pass algorithms. Such algorithms render multiple times, each time with a different GL mode vector, to eventually produce a result in the framebuffer. Examples of these algorithms include:

- "Erasing" a primitive from the framebuffer by redrawing it, either in a different color or using the XOR logical operation.
- Using stencil operations to compute capping planes.

- *Scissor parameters (other than enable)*

8. Polygon shading is completed before the polygon mode is interpreted. If the

17. (No pixel dropouts or duplicates.) Let two polygons share an identical edge (that is, there exist vertices A and B of an edge of one polygon, and vertices C and D of an edge of the other polygon, and the coordinates of vertex A (resp. B) are identical to those of vertex C (resp. D), and the state of the the coordinate transfomations is identical when A, B, C, and D are specified). Then, when the fragments produced by rasterization of both polygons are taken together, each fragment intersecting the interior of the shared edge is produced exactly once.

18. OpenGL state continues to be modified in `FEEDBACK` mode and in `SELECT` mode50re349 s

Appendix C

Version 1.1

Appendix D

Version 1.2

OpenGL version 1.2, released on March 16, 1998, is the second revision since the original version 1.0. Version 1.2 is upward compatible with version 1.1, meaning that any program that runs with a 1.1 GL implementation will also run unchanged with a 1.2 GL implementation.

Several additions were made to the GL, especially to texture mapping capa-

D.3 Packed Pixel Formats

Packed pixels in host memory are represented entirely by one unsigned byte, one unsigned short, or one unsigned integer. The fields with the packed pixel are not proper machine types, but the pixel as a whole is. Thus the pixel storage modes and their unpacking counterparts all work correctly with packed pixels.

The additions match those of the

The additions match those of the `SGIS_texture_edge_clamp` extension.

D.7 Texture Level of Detail Control

Two constraints related to the texture level of detail parameter λ are added. One constraint clamps λ to a specified floating point range. The other limits the se-

D.9.4 Pixel Pipeline Statistics

Pixel operations that count occurences of specific color component values (histogram) and that track the minimum and maximum color component values (minmax) are performed at the end of the pixel transfer pipeline. An optional mode allows pixel data to be discarded after the histogram and/or minmax operations are

Phil Lacroute, Silicon Graphics

Henri Warren, Digital Equipment / Megatek
Paula Womack, Silicon Graphics
Steve Wright, Microsoft
David Yu, Silicon Graphics
Randy Zhao, S3

Appendix F

Version 1.3

for the next texture environment. Changes to texture client state and texture server image, the color returned is derived only from border texels. This behavior mirrors the behavior of the texture edge clamp mode introduced by OpenGL 1.2.

Texture border clamp was promoted from the `GL_ARB_texture_border_clamp` extension.

F.9 Transpose Matrix

New functions and tokens are added allowing application matrices stored in row major order rather than column major order to be transferred to the implementation. This allows an application to use standard C-language 2-dimensional arrays and have the array indices match the expected
These arrays are referred to as transpose rices they are the transpose of
the standard matrices passed to OpenGL.

Transpose adds interface transfering to from OpenGL
pipeline. It does not change any OpenGL processing or imply any changes in state representation.

Transpose as the

Tim Kelley, Real 3D
Tom Frisinger, ATI
Victor Vedovato, Micron
Vikram Simha, MERL
Yanjun Zhang, Sun
Zahid Hussain, TI

Appendix G

Version 1.4

Blend squaring was promoted from the `GL_NV`

Kurt Akeley, NVIDIA
Allen Akin
Bill Armstrong, Evans & Sutherland
Ben Ashbaugh, Intel
Chris Bentley, ATI
Bob Beretta, Apple
Daniel Brokenshire, IBM
Pat Brown, NVIDIA
Bill Clifford, Intel
Graham Connor, Videologic
Matt Craighead, NVIDIA
Suzy Deffeyes, IBM
Jean-Luc Dery, Discreet
Kenneth Dyke, Apple
Cass Everitt, NVIDIA
Allen Gallotta, ATI
Lee Gross, IBM
Evan Hart, ATI
Chris Hecker, Definition 6
Alan Heirich, Compaq / HP
Gareth Hughes, VA Linux
Michael I Gold, NVIDIA
Rich Johnson, HP
Mark Kilgard, NVIDIA
Dale Kirkland, 3Dlabs
David Kirk, NVIDIA
Christian Laforte, Alias—Wavefront
Luc Leblanc, Discreet
Jon Leech, SGI
Bill Licea-Kane, ATI
Barthold Lichtenbelt, 3Dlabs
Jack Middleton, Sun
Howard Miller, Apple
Jeremy Morris, 3Dlabs
Jon Paul Schelter, Matrox
Brian Paul, VA Linux / Tungsten Graphics
Bimal Poddar, Intel
Thomas Roell, Xi Graphics
Randi Rost, 3Dlabs
Jeremy Sandmel, ATI

Appendix H

Version 1.5

H.2 Occlusion Queries

An occlusion query is a mechanism whereby an application can query the number of pixels (or, more precisely, samples) drawn by a primitive or group of primitives. The primary purpose of occlusion queries is to determine the visibility of an object.

Occlusion query was promoted from the `ARB_occlusion_query` extension.

H.3 Shadow Functions

Texture comparison functions are generalized to support all eight binary functions rather than just `LEQUAL` and `GEQUAL`.

Texture comparison functions were promoted from the `EXT_shadow_funcs` extension.

H.4 Changed Tokens

To achieve consistency with the syntax guidelines for OpenGL function and token names, new token names are introduced to be used in place of old, inconsistent names. However, thebesuppored(,)-379(for)-.96back(w)10(rdns)]TJ 0 -13.549 Td[(comptsibility)-325(with)-3i-25de
and thetheyre(plac,y)-20((are)-20[(sHo)25(an)-250ian)-250tableT

Appendix I

ARB Extensions

OpenGL extensions that have been approved by the OpenGL Architectural Review Board (ARB) are described in this chapter. These extensions are not required to be supported by a conformant OpenGL implementation, but are expected to be widely available; they define functionality that is likely to move into the required feature set in a future revision of the specification.

In order not to compromise the readability of the core specification, ARB extensions are not integrated into the core language; instead, they are made available online in the *OpenGL Extension Registry* (as0 i.be noRch217.(nlar)18(g-289(.(nnumb-289(.(non)-25888(

- All enumerants defined by the extension will have names of the form *NAME*_ARB.

I.2 Promoting Extensions to Core Features

ARB extensions can be *promoted*

I.20 Window Raster Position

The name string for window raster position is `GL_ARB`

I.30 Point Sprites