*Appendix B*

# OpenGL State Variables

This appendix lists the queryable OpenGL state variables, their default values, and the commands for obtaining the values of these variables. It also describes OpenGL's error handling facility, and how to save and restore sets of state variables. The *OpenGL Reference Manual* contains detailed information on all the commands and constants discussed in this appendix. This appendix has these major sections:

- "**The Query Commands**"
- "**Error Handling**"
- "**Saving and Restoring Sets of State Variables**"
- "**OpenGL State Variables**"

## The Query Commands

There are four commands for obtaining simple state variables, and one for determining whether a particular state is enabled or disabled.

void *glGetBooleanv*(GLenum *pname*, GLboolean **params*);
void *glGetIntegerv*(GLenum *pname*, GLint **params*);
void *glGetFloatv*(GLenum *pname*, GLfloat **params*);
void *glGetDoublev*(GLenum *pname*, GLdouble **params*);

Obtains Boolean, integer, floating−point, or double−precision state variables. The*pname* argument is a symbolic constant indicating the state variable to return, and *params* is a pointer to an array of the indicated type in which to place the returned data. The possible values for *pname* are listed in the tables in "**OpenGL State Variables.**" A type conversion is performed if necessary to return the desired variable as the requested data type.

GLboolean *glIsEnabled*(GLenum *cap*)

Returns GL_TRUE if the mode specified by *cap* is enabled; otherwise, returns GL_FALSE. The possible values for *cap* are listed in the tables in "**OpenGL State Variables**"

Other specialized commands return specific state variables. The prototypes for these commands are listed here; to find out when you need to use these commands, use the tables in "**OpenGL State Variables**" Also see the *OpenGL Reference Manual*. OpenGL's error handling facility and the *glGetError()* command are described in more detail in the next section.

void *glGetClipPlane*(GLenum *plane*, GLdouble **equation*);
GLenum *glGetError*(void);
void *glGetLight*{if}*v*(GLenum *light*, GLenum *pname*, *TYPE *params*);
void *glGetMap*{ifd}*v*(GLenum *target*, GLenum *query*, *TYPE *v*);
void *glGetMaterial*{if}*v*(GLenum *face*, GLenum *pname*, *TYPE *params*);
void *glGetPixelMap*{f ui us}*v*(GLenum *map*, *TYPE *values*);
void *glGetPolygonStipple*(GLubyte **mask*);
const GLubyte * *glGetString*(GLenum *name*);
void *glGetTexEnv*{if}*v*(GLenum *target*, GLenum *pname*, *TYPE *params*);
void *glGetTexGen*{ifd}*v*(GLenum *coord*, GLenum *pname*, *TYPE *params*);
void *glGetTexImage*(GLenum *target*, GLint *level*, GLenum *format*, GLenum *type*, GLvoid **pixels*);
void *glGetTexLevelParameter*{if}*v*(GLenum *target*, GLint *level*, GLenum *pname*, *TYPE *params*);
void *glGetTexParameter*{if}*v*(GLenum *target*, GLenum *pname*, *TYPE *params*);

## Error Handling

When OpenGL detects an error, it records a current error code. The command that caused the error is ignored, so it has no effect on OpenGL state or on the framebuffer contents. (If the error recorded was GL_OUT_OF_MEMORY, however, the results of the command are undefined.) Once recorded, the current error code isn't cleared—that is, additional errors aren't recorded—until you call the query command *glGetError()*, which returns the current error code. In distributed implementations of OpenGL, there might be multiple current error codes, each of which remains set until queried. Once you've queried all the current error codes, or if there's no error, *glGetError()* returns GL_NO_ERROR. Thus, if you obtain an error code, it's good practice to continue to call *glGetError()* until GL_NO_ERROR is returned to be sure you've discovered all the errors. **Table B−1** lists the defined OpenGL error codes.

You can use the GLU routine *gluErrorString()* to obtain a descriptive string corresponding to the error code passed in. This routine is described in more detail in "**Describing Errors.**" Note that GLU routines often return error values if an error is detected. Also, the GLU defines the error codes GLU_INVALID_ENUM, GLU_INVALID_VALUE, and GLU_OUT_OF_MEMORY, which have the same meaning as the related OpenGL codes.

| Error Code | Description |
|---|---|
| GL_INVALID_ENUM | GLenum argument out of range |
| GL_INVALID_VALUE | Numeric argument out of range |
| GL_INVALID_OPERATION | Operation illegal in current state |
| GL_STACK_OVERFLOW | Command would cause a stack overflow |
| GL_STACK_UNDERFLOW | Command would cause a stack underflow |
| GL_OUT_OF_MEMORY | Not enough memory left to execute command |

**Table B−1** OpenGL Error Codes

## Saving and Restoring Sets of State Variables

You can save and restore the values of a collection of state variables on an attribute stack with the commands *glPushAttrib()* and *glPopAttrib()*. The attribute stack has a depth of at least 16, and the actual depth can be obtained using GL_MAX_ATTRIB_STACK_DEPTH with *glGetIntegerv()*. Pushing a full stack or popping an empty one generates an error.

In general, it's faster to use *glPushAttrib()* and *glPopAttrib()* than to get and restore the values yourself. Some values might be pushed and popped in the hardware, and saving and restoring them might be expensive. Also, if you're operating on a remote client, all the attribute data has to be transferred across the network connection and back as it's saved and restored. However, your OpenGL implementation keeps the attribute stack on the server, avoiding unnecessary network delays.

---

void *glPushAttrib*(GLbitfield *mask*);

---

Saves all the attributes indicated by bits in *mask* by pushing them onto the attribute stack. **Table B−2** lists the possible mask bits that can be logically ORed together to save any combination of attributes. Each bit corresponds to a collection of individual state variables. For example, GL_LIGHTING_BIT refers to all the state variables related to lighting, which include the current material color, the ambient, diffuse, specular, and emitted light, a list of the lights that are enabled, and the directions of the spotlights. When *glPopAttrib()* is called, all those variables are restored. To find out exactly which attributes are saved for particular mask values, see the tables in "OpenGL State Variables."

| Mask Bit | Attribute Group |
|---|---|
| GL_ACCUM_BUFFER_BIT | accum−buffer |
| GL_ALL_ATTRIB_BITS | −− |
| GL_COLOR_BUFFER_BIT | color−buffer |
| GL_CURRENT_BIT | current |

| | |
|---|---|
| GL_DEPTH_BUFFER_BIT | depth−buffer |
| GL_ENABLE_BIT | enable |
| GL_EVAL_BIT | eval |
| GL_FOG_BIT | fog |
| GL_HINT_BIT | hint |
| GL_LIGHTING_BIT | lighting |
| GL_LINE_BIT | line |
| GL_LIST_BIT | list |
| GL_PIXEL_MODE_BIT | pixel |
| GL_POINT_BIT | point |
| GL_POLYGON_BIT | polygon |
| GL_POLYGON_STIPPLE_BIT | polygon−stipple |
| GL_SCISSOR_BIT | scissor |
| GL_STENCIL_BUFFER_BIT | stencil−buffer |
| GL_TEXTURE_BIT | texture |
| GL_TRANSFORM_BIT | transform |
| GL_VIEWPORT_BIT | viewport |

**Table B−2** Attribute Groups

void *glPopAttrib*(void);

Restores the values of those state variables that were saved with the last *glPushAttrib()*.

# OpenGL State Variables

The following pages contain tables that list the names of queryable state variables. For each variable, the tables list a description of it, its attribute group, its initial or minimum value, and the suggested *glGet*()* command to use for obtaining it. State variables that can be obtained using *glGetBooleanv(), glGetIntegerv(), glGetFloatv(),* or *glGetDoublev()* are listed with just one of these commands—the one that's most appropriate given the type of data to be returned. These state variables can't be obtained using *glIsEnabled()*. However, state variables for which *glIsEnabled()* is listed as the query command can also be obtained using *glGetBooleanv(), glGetIntegerv(), glGetFloatv(),* and *glGetDoublev()*. State variables for which any other command is listed as the query command can be obtained only by using that command. If no attribute group is listed, the variable doesn't belong to any group. All queryable state variables except the implementation−dependent ones have initial values. If no initial value is listed, you need to consult the section where that variable is discussed (or the *OpenGL Reference Manual*) to determine its initial value.

## Current Values and Associated Data

| State Variable | Description | Attribute Group |
|---|---|---|
| GL_CURRENT_COLOR | Current color | current |
| GL_CURRENT_INDEX | Current color index | current |
| GL_CURRENT_TEXTURE_COORDS | Current texture coordinates | current |
| GL_CURRENT_NORMAL | Current normal | current |
| GL_CURRENT_RASTER_POSITION | Current raster position | current |
| GL_CURRENT_RASTER_DISTANCE | Current raster distance | current |
| GL_CURRENT_RASTER_COLOR | Color associated with raster position | current |
| GL_CURRENT_RASTER_INDEX | Color index associated with raster position | current |
| GL_CURRENT_RASTER_TEXTURE_ | Texture coordinates | current |

| | | |
|---|---|---|
| COORDS | associated with raster position | |
| GL_CURRENT_RASTER_POSITION_ VALID | Raster position valid bit | current |
| GL_EDGE_FLAG | Edge flag | current |

**Table B−3** State Variables for Current Values and Associated Data

## Transformation

| State Variable | Description | Attribute Group |
|---|---|---|
| GL_MODELVIEW_MATRIX | Modelview matrix stack | −− |
| GL_PROJECTION_MATRIX | Projection matrix stack | −− |
| GL_TEXTURE_MATRIX | Texture matrix stack | −− |
| GL_VIEWPORT | Viewport origin and extent | viewport |
| GL_DEPTH_RANGE | Depth range near and far | viewport |
| GL_MODELVIEW_STACK_DEPTH | Modelview matrix stack pointer | −− |
| GL_PROJECTION_STACK_DEPTH | Projection matrix stack pointer | −− |
| GL_TEXTURE_STACK_DEPTH | Texture matrix stack pointer | −− |
| GL_MATRIX_MODE | Current matrix mode | transform |
| GL_NORMALIZE | Current normal normalization on/off | transform/ enable |
| GL_CLIP_PLANE*i* | User clipping plane coefficients | transform |
| GL_CLIP_PLANE*i* | *i*th user clipping plane enabled | transform/ enable |

**Table B−4** Transformation State Variables

## Coloring

| State Variable | Description | Attribute Group | Initial Value |
|---|---|---|---|
| GL_FOG_COLOR | Fog color | fog | 0, 0, 0, 0 |
| GL_FOG_INDEX | Fog index | fog | 0 |
| GL_FOG_DENSITY | Exponential fog density | fog | 1.0 |
| GL_FOG_START | Linear fog start | fog | 0.0 |
| GL_FOG_END | Linear fog end | fog | 1.0 |
| GL_FOG_MODE | Fog mode | fog | GL_EXP |
| GL_FOG | True if fog enabled | fog/enable | GL_FAL SE |
| GL_SHADE_MODEL | glShadeModel() setting | lighting | GL_SMO OTH |

**Table B−5** Coloring State Variables

## Lighting

See also **Table 6−1** and **Table 6−2** for initial values.

| State Variable | Description | Attribute Group |
|---|---|---|
| GL_LIGHTING | True if lighting is enabled | lighting /enable |
| GL_COLOR_MATERIAL | True if color tracking is enabled | lighting |
| GL_COLOR_MATERIAL_PARAME TER | Material properties tracking | lighting |

| | | |
|---|---|---|
| | current color | |
| GL_COLOR_MATERIAL_FACE | Face(s) affected by color tracking | lighting |
| GL_AMBIENT | Ambient material color | lighting |
| GL_DIFFUSE | Diffuse material color | lighting |
| GL_SPECULAR | Specular material color | lighting |
| GL_EMISSION | Emissive material color | lighting |
| GL_SHININESS | Specular exponent of material | lighting |
| GL_LIGHT_MODEL_AMBIENT | Ambient scene color | lighting |
| GL_LIGHT_MODEL_LOCAL_VIE WER | Viewer is local | lighting |
| GL_LIGHT_MODEL_TWO_SIDE | Use two−sided lighting | lighting |
| GL_AMBIENT | Ambient intensity of light *i* | lighting |
| GL_DIFFUSE | Diffuse intensity of light *i* | lighting |
| GL_SPECULAR | Specular intensity of light *i* | lighting |
| GL_POSITION | Position of light *i* | lighting |
| GL_CONSTANT_ATTENUATION | Constant attenuation factor | lighting |
| GL_LINEAR_ATTENUATION | Linear attenuation factor | lighting |
| GL_QUADRATIC_ATTENUATION | Quadratic attenuation factor | lighting |
| GL_SPOT_DIRECTION | Spotlight direction of light *i* | lighting |
| GL_SPOT_EXPONENT | Spotlight exponent of light *i* | lighting |
| GL_SPOT_CUTOFF | Spotlight angle of light *i* | lighting |
| GL_LIGHT*i* | True if light *i* enabled | lighting /enable |
| GL_COLOR_INDEXES | ca, cd, and cs for color−index lighting | lighting /enable |

**Table B−6** Lighting State Variables

## Rasterization

| State Variable | Description | Attribute Group |
|---|---|---|
| GL_POINT_SIZE | Point size | point |
| GL_POINT_SMOOTH | Point antialiasing on | point/enable |
| GL_LINE_WIDTH | Line width | line |
| GL_LINE_SMOOTH | Line antialiasing on | line/enable |
| GL_LINE_STIPPLE_PATTERN | Line stipple | line |
| GL_LINE_STIPPLE_REPEAT | Line stipple repeat | line |
| GL_LINE_STIPPLE | Line stipple enable | line/enable |
| GL_CULL_FACE | Polygon culling enabled | polygon/enable |
| GL_CULL_FACE_MODE | Cull front−/back−facing polygons | polygon |
| GL_FRONT_FACE | Polygon front−face CW/CCW indicator | polygon |
| GL_POLYGON_SMOOTH | Polygon antialiasing on | polygon/enable |
| GL_POLYGON_MODE | Polygon rasterization mode (front and back) | polygon |
| GL_POLYGON_STIPPLE | Polygon stipple enable | polygon/enable |
| −− | Polygon stipple pattern | polygon−stipple |

**Table B−7** Rasterization State Variables

## Texturing

| State Variable | Description | Attribute Group | Initial Value |
|---|---|---|---|
| GL_TEXTURE_*x* | True if *x*−D texturing enabled (*x* is 1D or 2D) | texture/ enable | GL_F ALSE |

| | | | |
|---|---|---|---|
| GL_TEXTURE | *x*−D texture image at level of detail *i* | −− | −− |
| GL_TEXTURE_WIDTH | *x*−D texture image *i*'s width | −− | 0 |
| GL_TEXTURE_HEIGHT | *x*−D texture image *i*'s height | −− | 0 |
| GL_TEXTURE_BORDER | *x*−D texture image *i*'s border width | −− | 0 |
| GL_TEXTURE_COMPONENTS | Texture image components | −− | 1 |
| GL_TEXTURE_BORDER_COL OR | Texture border color | texture | 0, 0, 0, 0 |
| GL_TEXTURE_MIN_FILTER | Texture minification function | texture | GL_NEAREST_MIPMAP_LINEAR |
| GL_TEXTURE_MAG_FILTER | Texture magnification function | texture | GL_LINEAR |
| GL_TEXTURE_WRAP_*x* | Texture wrap mode (*x* is S or T) | texture | GL_REPEAT |
| GL_TEXTURE_ENV_MODE | Texture application function | texture | GL_MODULATE |
| GL_TEXTURE_ENV_COLOR | Texture environment color | texture | 0, 0, 0, 0 |
| GL_TEXTURE_GEN_*x* | Texgen enabled (*x* is S, T, R, or Q) | texture/ enable | GL_FALSE |
| GL_EYE_LINEAR | Texgen plane equation coefficients | texture | −− |
| GL_OBJECT_LINEAR | Texgen object linear coefficients | texture | −− |
| GL_TEXTURE_GEN_MODE | Function used for texgen | texture | GL_EYE_LINEAR |

**Table B−8** Texturing State Variables

## Pixel Operations

| State Variable | Description | Attribute Group |
|---|---|---|
| GL_SCISSOR_TEST | Scissoring enabled | scissor/enable |
| GL_SCISSOR_BOX | Scissor box | scissor |
| GL_STENCIL_TEST | Stenciling enabled | stencil−buffer/ enable |
| GL_STENCIL_FUNC | Stencil function | stencil−buffer |
| GL_STENCIL_VALUE_MASK | Stencil mask | stencil−buffer |
| GL_STENCIL_REF | Stencil reference value | stencil−buffer |
| GL_STENCIL_FAIL | Stencil fail action | stencil−buffer |
| GL_STENCIL_PASS_DEPTH_FAI L | Stencil depth buffer fail action | stencil−buffer |
| GL_STENCIL_PASS_DEPTH_PAS S | Stencil depth buffer pass action | stencil−buffer |
| GL_ALPHA_TEST | Alpha test enabled | color−buffer/enable |
| GL_ALPHA_TEST_FUNC | Alpha test function | color−buffer |

| | | |
|---|---|---|
| GL_ALPHA_TEST_REF | Alpha test reference value | color−buffer |
| GL_DEPTH_TEST | Depth buffer enabled | depth−buffer/ enable |
| GL_DEPTH_FUNC | Depth buffer test function | depth−buffer |
| GL_BLEND | Blending enabled | color−buffer/e nable |
| GL_BLEND_SRC | Blending source function | color−buffer |
| GL_BLEND_DST | Blending destination function | color−buffer |
| GL_LOGIC_OP | Logical operation enabled | color−buffer/e nable |
| GL_LOGIC_OP_MODE | Logical operation function | color−buffer |
| GL_DITHER | Dithering enabled | color−buffer/e nable |

**Table B−9** Pixel Operations

## Framebuffer Control

| State Variable | Description | Attribute Group |
|---|---|---|
| GL_DRAW_BUFFER | Buffers selected for drawing | color−buffer |
| GL_INDEX_WRITEMASK | Color−index writemask | color−buffer |
| GL_COLOR_WRITEMASK | Color write enables; R, G, B, or A | color−buffer |
| GL_DEPTH_WRITEMASK | Depth buffer enabled for writing | depth−buffer |
| GL_STENCIL_WRITEMASK | Stencil−buffer writemask | stencil−buffer |
| GL_COLOR_CLEAR_VALUE | Color−buffer clear value (RGBA mode) | color−buffer |
| GL_INDEX_CLEAR_VALUE | Color−buffer clear value (color−index mode) | color−buffer |
| GL_DEPTH_CLEAR_VALUE | Depth−buffer clear value | depth−buffer |
| GL_STENCIL_CLEAR_VALUE | Stencil−buffer clear value | stencil−buffer |
| GL_ACCUM_CLEAR_VALUE | Accumulation−buffer clear value | accum−buffer |

**Table B−10** Framebuffer Control State Variables

## Pixels

| State Variable | Description | Attribute Group |
|---|---|---|
| GL_UNPACK_SWAP_BYTES | Value of GL_UNPACK_SWAP_BYTES | −− |
| GL_UNPACK_LSB_FIRST | Value of GL_UNPACK_LSB_FIRST | −− |
| GL_UNPACK_ROW_LENGTH | Value of GL_UNPACK_ROW_LENGT H | −− |
| GL_UNPACK_SKIP_ROWS | Value of GL_UNPACK_SKIP_ROWS | −− |
| GL_UNPACK_SKIP_PIXELS | Value of GL_UNPACK_SKIP_PIXELS | −− |
| GL_UNPACK_ALIGNMENT | Value of GL_UNPACK_ALIGNMENT | −− |
| GL_PACK_SWAP_BYTES | Value of GL_PACK_SWAP_BYTES | −− |
| GL_PACK_LSB_FIRST | Value of GL_PACK_LSB_FIRST | −− |
| GL_PACK_ROW_LENGTH | Value of GL_PACK_ROW_LENGTH | −− |
| GL_PACK_SKIP_ROWS | Value of GL_PACK_SKIP_ROWS | −− |

| | | |
|---|---|---|
| GL_PACK_SKIP_PIXELS | Value of GL_PACK_SKIP_PIXELS | −− |
| GL_PACK_ALIGNMENT | Value of GL_PACK_ALIGNMENT | −− |
| GL_MAP_COLOR | True if colors are mapped | pixel |
| GL_MAP_STENCIL | True if stencil values are mapped | pixel |
| GL_INDEX_SHIFT | Value of GL_INDEX_SHIFT | pixel |
| GL_INDEX_OFFSET | Value of GL_INDEX_OFFSET | pixel |
| GL_*x*_SCALE | Value of GL_*x*_SCALE; *x* is GL_RED, GL_GREEN, GL_BLUE, GL_ALPHA, or GL_DEPTH | pixel |
| GL_*x*_BIAS | Value of GL_*x*_BIAS; *x* is one of GL_RED, GL_GREEN, GL_BLUE, GL_ALPHA, or GL_DEPTH | pixel |
| GL_ZOOM_X | *x* zoom factor | pixel |
| GL_ZOOM_Y | *y* zoom factor | pixel |
| GL_x | glPixelMap() translation tables; *x* is a map name from **Table 8−5** | pixel |
| GL_*x*_SIZE | Size of table *x* | pixel |
| GL_READ_BUFFER | Read source buffer | pixel |

**Table B−11** Pixel State Variables

## Evaluators

| State Variable | Description | Attribute Group | Initial Value |
|---|---|---|---|
| GL_ORDER | 1D map order | −− | 1 |
| GL_ORDER | 2D map orders | −− | 1, 1 |
| GL_COEFF | 1D control points | −− | −− |
| GL_COEFF | 2D control points | −− | −− |
| GL_DOMAIN | 1D domain endpoints | −− | −− |
| GL_DOMAIN | 2D domain endpoints | −− | −− |
| GL_MAP1_*x* | 1D map enables: *x* is map type | eval/enable | GL_FALSE |
| GL_MAP2_*x* | 2D map enables: *x* is map type | eval/enable | GL_FALSE |
| GL_MAP1_GRID_DOMAIN | 1D grid endpoints | eval | 0, 1 |
| GL_MAP2_GRID_DOMAIN | 2D grid endpoints | eval | 0, 1; 0, 1 |
| GL_MAP1_GRID_SEGMENTS | 1D grid divisions | eval | 1 |
| GL_MAP2_GRID_SEGMENTS | 2D grid divisions | eval | 1,1 |
| GL_AUTO_NORMAL | True if automatic normal generation enabled | eval | GL_FALSE |

**Table B−12** Evaluator State Variables

## Hints

| State Variable | Description | Attribute Group | Initial Value |
|---|---|---|---|

| | | | |
|---|---|---|---|
| GL_PERSPECTIVE_CORRECTION_ HINT | Perspective correction hint | hint | GL_DONT_CARE |
| GL_POINT_SMOOTH_HINT | Point smooth hint | hint | GL_DONT_CARE |
| GL_LINE_SMOOTH_HINT | Line smooth hint | hint | GL_DONT_CARE |
| GL_POLYGON_SMOOTH_HINT | Polygon smooth hint | hint | GL_DONT_CARE |
| GL_FOG_HINT | Fog hint | hint | GL_DONT_CARE |

**Table B−13** Hint State Variables

## Implementation−Dependent Values

| State Variable | Description | Attribute Group |
|---|---|---|
| GL_MAX_LIGHTS | Maximum number of lights | −− |
| GL_MAX_CLIP_PLANES | Maximum number of user clipping planes | −− |
| GL_MAX_MODELVIEW_STACK_DEPTH | Maximum modelview−matrix stack depth | −− |
| GL_MAX_PROJECTION_STACK_DEPTH | Maximum projection−matrix stack depth | −− |
| GL_MAX_TEXTURE_STACK_DE PTH | Maximum depth of texture matrix stack | −− |
| GL_SUBPIXEL_BITS | Number of bits of subpixel precision in x and y | −− |
| GL_MAX_TEXTURE_SIZE | Maximum height or width of a texture image (w/o borders) | −− |
| GL_MAX_PIXEL_MAP_TABLE | Maximum size of a glPixelMap() translation table | −− |
| GL_MAX_NAME_STACK_DEPT H | Maximum selection−name stack depth | −− |
| GL_MAX_LIST_NESTING | Maximum display−list call nesting | −− |
| GL_MAX_EVAL_ORDER | Maximum evaluator polynomial order | −− |
| GL_MAX_VIEWPORT_DIMS | Maximum viewport dimensions | −− |
| GL_MAX_ATTRIB_STACK_DEPT H | Maximum depth of the attribute stack | −− |
| GL_AUX_BUFFERS | Number of auxiliary buffers | −− |
| GL_RGBA_MODE | True if color buffers store RGBA | −− |
| GL_INDEX_MODE | True if color buffers store indices | −− |

| | | |
|---|---|---|
| GL_DOUBLEBUFFER | True if front & back buffers exist | −− |
| GL_STEREO | True if left & right buffers exist | −− |
| GL_POINT_SIZE_RANGE | Range (low to high) of antialiased point sizes | −− |
| GL_POINT_SIZE_GRANULARIT Y | Antialiased point−size granularity | −− |
| GL_LINE_WIDTH_RANGE | Range (low to high) of antialiased line widths | −− |
| GL_LINE_WIDTH_GRANULARI TY | Antialiased line−width granularity | −− |

**Table B−14** Implementation−Dependent State Variables

## Implementation−Dependent Pixel Depths

| State Variable | Description | Attribute Group |
|---|---|---|
| GL_RED_BITS | Number of bits per red component in color buffers | −− |
| GL_GREEN_BITS | Number of bits per green component in color buffers | −− |
| GL_BLUE_BITS | Number of bits per blue component in color buffers | −− |
| GL_ALPHA_BITS | Number of bits per alpha component in color buffers | −− |
| GL_INDEX_BITS | Number of bits per index in color buffers | −− |
| GL_DEPTH_BITS | Number of depth−buffer bitplanes | −− |
| GL_STENCIL_BITS | Number of stencil bitplanes | −− |
| GL_ACCUM_RED_BITS | Number of bits per red component in the accumulation buffer | −− |
| GL_ACCUM_GREEN_BITS | Number of bits per green component in the accumulation buffer | −− |
| GL_ACCUM_BLUE_BITS | Number of bits per blue component in the accumulation buffer | −− |
| GL_ACCUM_ALPHA_BITS | Number of bits per alpha component in the accumulation buffer | −− |

**Table B−15** Implementation−Dependent Pixel−Depth State Variables

## Miscellaneous

| State Variable | Description |
|---|---|
| GL_LIST_BASE | Setting of glListBase() |
| GL_LIST_INDEX | Number of display list under construction; 0 if none |
| GL_LIST_MODE | Mode of display list under construction; undefined if none |
| GL_ATTRIB_STACK_DEPTH | Attribute stack pointer |
| GL_NAME_STACK_DEPTH | Name stack depth |
| GL_RENDER_MODE | glRenderMode() setting |
| −− | Current error code(s) |

**Table B−16** Miscellaneous State Variables