NAME

glEnable, glDisable – enable or disable server-side GL capabilities

C SPECIFICATION

void **glEnable**(GLenum *cap*)

PARAMETERS

cap Specifies a symbolic constant indicating a GL capability.

C SPECIFICATION

void **glDisable**(GLenum *cap*)

PARAMETERS

cap Specifies a symbolic constant indicating a GL capability.

DESCRIPTION

glEnable and **glDisable** enable and disable various capabilities. Use **glIsEnabled** or **glGet** to determine the current setting of any capability. The initial value for each capability with the exception of **GL_DITHER** is **GL_FALSE**. The initial value for **GL_DITHER** is **GL_TRUE**.

Both **glEnable** and **glDisable** take a single argument, *cap*, which can assume one of the following values:

GL_ALPHA_TEST	If enabled, do alpha testing. See **glAlphaFunc**.
GL_AUTO_NORMAL	If enabled, generate normal vectors when either **GL_MAP2_VERTEX_3** or **GL_MAP2_VERTEX_4** is used to generate vertices. See **glMap2**.
GL_BLEND	If enabled, blend the incoming RGBA color values with the values in the color buffers. See **glBlendFunc**.
GL_CLIP_PLANE*i*	If enabled, clip geometry against user-defined clipping plane *i*. See **glClip-Plane**.
GL_COLOR_LOGIC_OP	If enabled, apply the currently selected logical operation to the incoming RGBA color and color buffer values. See **glLogicOp**.
GL_COLOR_MATERIAL	If enabled, have one or more material parameters track the current color. See **glColorMaterial**.
GL_CULL_FACE	If enabled, cull polygons based on their winding in window coordinates. See **glCullFace**.
GL_DEPTH_TEST	If enabled, do depth comparisons and update the depth buffer. Note that even if the depth buffer exists and the depth mask is non-zero, the depth buffer is not updated if the depth test is disabled. See **glDepthFunc** and **glDepthRange**.
GL_DITHER	If enabled, dither color components or indices before they are written to the color buffer.
GL_FOG	If enabled, blend a fog color into the posttexturing color. See **glFog**.
GL_INDEX_LOGIC_OP	If enabled, apply the currently selected logical operation to the incoming index and color buffer indices. See **glLogicOp**.
GL_LIGHT*i*	If enabled, include light *i* in the evaluation of the lighting equation. See **glLightModel** and **glLight**.
GL_LIGHTING	If enabled, use the current lighting parameters to compute the vertex color or index. Otherwise, simply associate the current color or index with each

vertex. See **glMaterial**, **glLightModel**, and **glLight**.

GL_LINE_SMOOTH If enabled, draw lines with correct filtering. Otherwise, draw aliased lines. See **glLineWidth**.

GL_LINE_STIPPLE If enabled, use the current line stipple pattern when drawing lines. See **glLineStipple**.

GL_MAP1_COLOR_4 If enabled, calls to **glEvalCoord1**, **glEvalMesh1**, and **glEvalPoint1** generate RGBA values. See **glMap1**.

GL_MAP1_INDEX If enabled, calls to **glEvalCoord1**, **glEvalMesh1**, and **glEvalPoint1** generate color indices. See **glMap1**.

GL_MAP1_NORMAL If enabled, calls to **glEvalCoord1**, **glEvalMesh1**, and **glEvalPoint1** generate normals. See **glMap1**.

GL_MAP1_TEXTURE_COORD_1
If enabled, calls to **glEvalCoord1**, **glEvalMesh1**, and **glEvalPoint1** generate *s* texture coordinates. See **glMap1**.

GL_MAP1_TEXTURE_COORD_2
If enabled, calls to **glEvalCoord1**, **glEvalMesh1**, and **glEvalPoint1** generate *s* and *t* texture coordinates. See **glMap1**.

GL_MAP1_TEXTURE_COORD_3
If enabled, calls to **glEvalCoord1**, **glEvalMesh1**, and **glEvalPoint1** generate *s*, *t*, and *r* texture coordinates. See **glMap1**.

GL_MAP1_TEXTURE_COORD_4
If enabled, calls to **glEvalCoord1**, **glEvalMesh1**, and **glEvalPoint1** generate *s*, *t*, *r*, and *q* texture coordinates. See **glMap1**.

GL_MAP1_VERTEX_3 If enabled, calls to **glEvalCoord1**, **glEvalMesh1**, and **glEvalPoint1** generate *x*, *y*, and *z* vertex coordinates. See **glMap1**.

GL_MAP1_VERTEX_4 If enabled, calls to **glEvalCoord1**, **glEvalMesh1**, and **glEvalPoint1** generate homogeneous *x*, *y*, *z*, and *w* vertex coordinates. See **glMap1**.

GL_MAP2_COLOR_4 If enabled, calls to **glEvalCoord2**, **glEvalMesh2**, and **glEvalPoint2** generate RGBA values. See **glMap2**.

GL_MAP2_INDEX If enabled, calls to **glEvalCoord2**, **glEvalMesh2**, and **glEvalPoint2** generate color indices. See **glMap2**.

GL_MAP2_NORMAL If enabled, calls to **glEvalCoord2**, **glEvalMesh2**, and **glEvalPoint2** generate normals. See **glMap2**.

GL_MAP2_TEXTURE_COORD_1
If enabled, calls to **glEvalCoord2**, **glEvalMesh2**, and **glEvalPoint2** generate *s* texture coordinates. See **glMap2**.

GL_MAP2_TEXTURE_COORD_2
If enabled, calls to **glEvalCoord2**, **glEvalMesh2**, and **glEvalPoint2** generate *s* and *t* texture coordinates. See **glMap2**.

GL_MAP2_TEXTURE_COORD_3
If enabled, calls to **glEvalCoord2**, **glEvalMesh2**, and **glEvalPoint2** generate *s*, *t*, and *r* texture coordinates. See **glMap2**.

GL_MAP2_TEXTURE_COORD_4
If enabled, calls to **glEvalCoord2**, **glEvalMesh2**, and **glEvalPoint2**

generate *s*, *t*, *r*, and *q* texture coordinates. See **glMap2**.

GL_MAP2_VERTEX_3 If enabled, calls to **glEvalCoord2**, **glEvalMesh2**, and **glEvalPoint2** generate *x*, *y*, and *z* vertex coordinates. See **glMap2**.

GL_MAP2_VERTEX_4 If enabled, calls to **glEvalCoord2**, **glEvalMesh2**, and **glEvalPoint2** generate homogeneous *x*, *y*, *z*, and *w* vertex coordinates. See **glMap2**.

GL_NORMALIZE If enabled, normal vectors specified with **glNormal** are scaled to unit length after transformation. See **glNormal**.

GL_POINT_SMOOTH If enabled, draw points with proper filtering. Otherwise, draw aliased points. See **glPointSize**.

GL_POLYGON_OFFSET_FILL
If enabled, and if the polygon is rendered in **GL_FILL** mode, an offset is added to depth values of a polygon's fragments before the depth comparison is performed. See **glPolygonOffset**.

GL_POLYGON_OFFSET_LINE
If enabled, and if the polygon is rendered in **GL_LINE** mode, an offset is added to depth values of a polygon's fragments before the depth comparison is performed. See **glPolygonOffset**.

GL_POLYGON_OFFSET_POINT
If enabled, an offset is added to depth values of a polygon's fragments before the depth comparison is performed, if the polygon is rendered in **GL_POINT** mode. See **glPolygonOffset**.

GL_POLYGON_SMOOTH If enabled, draw polygons with proper filtering. Otherwise, draw aliased polygons. For correct anti-aliased polygons, an alpha buffer is needed and the polygons must be sorted front to back.

GL_POLYGON_STIPPLE If enabled, use the current polygon stipple pattern when rendering polygons. See **glPolygonStipple**.

GL_SCISSOR_TEST If enabled, discard fragments that are outside the scissor rectangle. See **glScissor**.

GL_STENCIL_TEST If enabled, do stencil testing and update the stencil buffer. See **glStencilFunc** and **glStencilOp**.

GL_TEXTURE_1D If enabled, one-dimensional texturing is performed (unless two-dimensional texturing is also enabled). See **glTexImage1D**.

GL_TEXTURE_2D If enabled, two-dimensional texturing is performed. See **glTexImage2D**.

GL_TEXTURE_GEN_Q If enabled, the *q* texture coordinate is computed using the texture generation function defined with **glTexGen**. Otherwise, the current *q* texture coordinate is used. See **glTexGen**.

GL_TEXTURE_GEN_R If enabled, the *r* texture coordinate is computed using the texture generation function defined with **glTexGen**. Otherwise, the current *r* texture coordinate is used. See **glTexGen**.

GL_TEXTURE_GEN_S If enabled, the *s* texture coordinate is computed using the texture generation function defined with **glTexGen**. Otherwise, the current *s* texture coordinate is used. See **glTexGen**.

GL_TEXTURE_GEN_T If enabled, the *t* texture coordinate is computed using the texture generation function defined with **glTexGen**. Otherwise, the current *t* texture coordinate is used. See **glTexGen**.

NOTES

GL_POLYGON_OFFSET_FILL, **GL_POLYGON_OFFSET_LINE**, **GL_POLYGON_OFFSET_POINT**, **GL_COLOR_LOGIC_OP**, and **GL_INDEX_LOGIC_OP** are only available if the GL version is 1.1 or greater.

ERRORS

GL_INVALID_ENUM is generated if *cap* is not one of the values listed previously.

GL_INVALID_OPERATION is generated if **glEnable** or **glDisable** is executed between the execution of **glBegin** and the corresponding execution of **glEnd**.

SEE ALSO

glAlphaFunc, **glBlendFunc**, **glClipPlane**, **glColorMaterial**, **glCullFace**, **glDepthFunc**, **glDepthRange**, **glEnableClientState**, **glFog**, **glGet**, **glIsEnabled**, **glLight**, **glLightModel**, **glLineWidth**, **glLineStipple**, **glLogicOp**, **glMap1**, **glMap2**, **glMaterial**, **glNormal**, **glPointSize**, **glPolygonMode**, **glPolygonOffset**, **glPolygonStipple**, **glScissor**, **glStencilFunc**, **glStencilOp**, **glTexGen**, **glTexImage1D**, **glTexImage2D**