Name

glTexParameter, glTextureParameter — set texture parameters

C Specification

void **glTexParameterf**(GLenum *target*,
GLenum *pname*,
GLfloat *param*);

void **glTexParameteri**(GLenum *target*,
GLenum *pname*,
GLint *param*);

void **glTextureParameterf**(GLuint *texture*,
GLenum *pname*,
GLfloat *param*);

void **glTextureParameteri**(GLuint *texture*,
GLenum *pname*,
GLint *param*);

void **glTexParameterfv**(GLenum *target*,
GLenum *pname*,
const GLfloat * *params*);

void **glTexParameteriv**(GLenum *target*,
GLenum *pname*,
const GLint * *params*);

void **glTexParameterIiv**(GLenum *target*,
GLenum *pname*,
const GLint * *params*);

void **glTexParameterIuiv**(GLenum *target*,
GLenum *pname*,
const GLuint * *params*);

void **glTextureParameterfv**(GLuint *texture*,

```
                                   GLenum pname,
                                   const GLfloat *params);

void glTextureParameteriv( GLuint texture,
                           GLenum pname,
                           const GLint *params);

void glTextureParameterIiv( GLuint texture,
                            GLenum pname,
                            const GLint *params);

void glTextureParameterIuiv( GLuint texture,
                             GLenum pname,
                             const GLuint *params);
```

Parameters

target

Specifies the target to which the texture is bound for **glTexParameter** functions. Must be one of `GL_TEXTURE_1D`, `GL_TEXTURE_1D_ARRAY`, `GL_TEXTURE_2D`, `GL_TEXTURE_2D_ARRAY`, `GL_TEXTURE_2D_MULTISAMPLE`, `GL_TEXTURE_2D_MULTISAMPLE_ARRAY`, `GL_TEXTURE_3D`, `GL_TEXTURE_CUBE_MAP`, `GL_TEXTURE_CUBE_MAP_ARRAY`, or `GL_TEXTURE_RECTANGLE`.

texture

Specifies the texture object name for **glTextureParameter** functions.

pname

Specifies the symbolic name of a single-valued texture parameter. *pname* can be one of the following: `GL_DEPTH_STENCIL_TEXTURE_MODE`, `GL_TEXTURE_BASE_LEVEL`, `GL_TEXTURE_COMPARE_FUNC`, `GL_TEXTURE_COMPARE_MODE`, `GL_TEXTURE_LOD_BIAS`, `GL_TEXTURE_MIN_FILTER`, `GL_TEXTURE_MAG_FILTER`, `GL_TEXTURE_MIN_LOD`, `GL_TEXTURE_MAX_LOD`, `GL_TEXTURE_MAX_LEVEL`, `GL_TEXTURE_SWIZZLE_R`, `GL_TEXTURE_SWIZZLE_G`, `GL_TEXTURE_SWIZZLE_B`, `GL_TEXTURE_SWIZZLE_A`, `GL_TEXTURE_WRAP_S`, `GL_TEXTURE_WRAP_T`, or `GL_TEXTURE_WRAP_R`.

For the vector commands (**glTexParameter*v**), *pname* can also be one of `GL_TEXTURE_BORDER_COLOR` or `GL_TEXTURE_SWIZZLE_RGBA`.

param

For the scalar commands, specifies the value of *pname*.

params

For the vector commands, specifies a pointer to an array where the value or values of *pname* are stored.

Description

glTexParameter and **glTextureParameter** assign the value or values in *params* to the texture parameter specified as *pname*. For **glTexParameter**, *target* defines the target texture, either GL_TEXTURE_1D, GL_TEXTURE_1D_ARRAY, GL_TEXTURE_2D, GL_TEXTURE_2D_ARRAY, GL_TEXTURE_2D_MULTISAMPLE, GL_TEXTURE_2D_MULTISAMPLE_ARRAY, GL_TEXTURE_3D, GL_TEXTURE_CUBE_MAP, GL_TEXTURE_CUBE_MAP_ARRAY, or GL_TEXTURE_RECTANGLE. The following symbols are accepted in *pname*:

GL_DEPTH_STENCIL_TEXTURE_MODE

Specifies the mode used to read from depth-stencil format textures. *params* must be one of GL_DEPTH_COMPONENT or GL_STENCIL_INDEX. If the depth stencil mode is GL_DEPTH_COMPONENT, then reads from depth-stencil format textures will return the depth component of the texel in R_t and the stencil component will be discarded. If the depth stencil mode is GL_STENCIL_INDEX then the stencil component is returned in R_t and the depth component is discarded. The initial value is GL_DEPTH_COMPONENT.

GL_TEXTURE_BASE_LEVEL

Specifies the index of the lowest defined mipmap level. This is an integer value. The initial value is 0.

GL_TEXTURE_BORDER_COLOR

The data in *params* specifies four values that define the border values that should be used for border texels. If a texel is sampled from the border of the texture, the values of GL_TEXTURE_BORDER_COLOR are interpreted as an RGBA color to match the texture's internal format and substituted for the non-existent texel data. If the texture contains depth components, the first component of GL_TEXTURE_BORDER_COLOR is interpreted as a depth value. The initial value is $(0.0, 0.0, 0.0, 0.0)$.

If the values for GL_TEXTURE_BORDER_COLOR are specified with **glTexParameterIiv** or **glTexParameterIuiv**, the values are stored unmodified with an internal data type of integer. If specified with **glTexParameteriv**, they are converted to floating point with the following equation: $f = \frac{2c+1}{2^b-1}$. If specified with **glTexParameterfv**, they are stored unmodified as floating-point values.

GL_TEXTURE_COMPARE_FUNC

Specifies the comparison operator used when GL_TEXTURE_COMPARE_MODE is set to GL_COMPARE_REF_TO_TEXTURE. Permissible values are:

Texture Comparison Function	**Computed result**
GL_LEQUAL	$result = \begin{cases} 1.0 & r <= D_t \\ 0.0 & r > D_t \end{cases}$
GL_GEQUAL	$result = \begin{cases} 1.0 & r >= D_t \\ 0.0 & r < D_t \end{cases}$
GL_LESS	$result = \begin{cases} 1.0 & r < D_t \\ 0.0 & r >= D_t \end{cases}$
GL_GREATER	$result = \begin{cases} 1.0 & r > D_t \\ 0.0 & r <= D_t \end{cases}$
GL_EQUAL	$result = \begin{cases} 1.0 & r = D_t \\ 0.0 & r \neq D_t \end{cases}$
GL_NOTEQUAL	$result = \begin{cases} 1.0 & r \neq D_t \\ 0.0 & r = D_t \end{cases}$
GL_ALWAYS	$result = 1.0$
GL_NEVER	$result = 0.0$

where r is the current interpolated texture coordinate, and D_t is the depth texture value sampled from the currently bound depth texture. $result$ is assigned to the red channel.

`GL_TEXTURE_COMPARE_MODE`

Specifies the texture comparison mode for currently bound depth textures. That is, a texture whose internal format is `GL_DEPTH_COMPONENT_*`; see glTexImage2D) Permissible values are:

`GL_COMPARE_REF_TO_TEXTURE`

Specifies that the interpolated and clamped r texture coordinate should be compared to the value in the currently bound depth texture. See the discussion of `GL_TEXTURE_COMPARE_FUNC` for details of how the comparison is evaluated. The result of the comparison is assigned to the red channel.

`GL_NONE`

Specifies that the red channel should be assigned the appropriate value from the currently bound depth texture.

`GL_TEXTURE_LOD_BIAS`

params specifies a fixed bias value that is to be added to the level-of-detail parameter for the texture before texture sampling. The specified value is added to the shader-supplied bias value (if any) and subsequently clamped into the implementation-defined range $[-bias_{max}, bias_{max}]$, where $bias_{max}$ is the value of the implementation defined constant `GL_MAX_TEXTURE_LOD_BIAS`. The initial value is 0.0.

`GL_TEXTURE_MIN_FILTER`

The texture minifying function is used whenever the level-of-detail function used when sampling from the texture determines that the texture should be minified. There are six defined minifying functions. Two of them use either the nearest texture elements or a weighted average of multiple texture elements to compute the texture value. The other four use mipmaps.

A mipmap is an ordered set of arrays representing the same image at progressively lower resolutions. If the texture has dimensions $2^n \times 2^m$, there are $max\left(n, m\right) + 1$ mipmaps. The first mipmap is the original texture, with dimensions $2^n \times 2^m$. Each subsequent mipmap has dimensions $2^{k-1} \times 2^{l-1}$, where $2^k \times 2^l$ are the dimensions of the previous mipmap, until either $k = 0$ or $l = 0$. At that point, subsequent mipmaps have dimension $1 \times 2^{l-1}$ or $2^{k-1} \times 1$ until the final mipmap, which has dimension 1×1. To define the mipmaps, call glTexImage1D, glTexImage2D, glTexImage3D, glCopyTexImage1D, or glCopyTexImage2D with the *level* argument indicating the order of the mipmaps. Level 0 is the original texture; level $max\left(n, m\right)$ is the final 1×1 mipmap.

params supplies a function for minifying the texture as one of the following:

`GL_NEAREST`

Returns the value of the texture element that is nearest (in Manhattan distance) to the specified texture coordinates.

`GL_LINEAR`

Returns the weighted average of the four texture elements that are closest to the specified texture coordinates. These can include items wrapped or repeated from other parts of a texture, depending on the values of `GL_TEXTURE_WRAP_S` and `GL_TEXTURE_WRAP_T`, and on the exact mapping.

`GL_NEAREST_MIPMAP_NEAREST`

Chooses the mipmap that most closely matches the size of the pixel being textured and uses the `GL_NEAREST` criterion (the texture element closest to the specified texture coordinates) to produce a texture value.

`GL_LINEAR_MIPMAP_NEAREST`

Chooses the mipmap that most closely matches the size of the pixel being textured and uses the `GL_LINEAR` criterion (a weighted average of the four texture elements that are closest to the specified texture coordinates) to produce a texture value.

`GL_NEAREST_MIPMAP_LINEAR`

Chooses the two mipmaps that most closely match the size of the pixel being textured and uses the `GL_NEAREST` criterion (the texture element closest to the specified texture coordinates) to produce a texture value from each mipmap. The final texture value is a weighted average of those two values.

`GL_LINEAR_MIPMAP_LINEAR`

Chooses the two mipmaps that most closely match the size of the pixel being textured and uses the `GL_LINEAR` criterion (a weighted average of the texture elements that are closest to the specified texture coordinates) to produce a texture value from each mipmap. The final texture value is a weighted average of those two values.

As more texture elements are sampled in the minification process, fewer aliasing artifacts will be apparent. While the `GL_NEAREST` and `GL_LINEAR` minification functions can be faster than the other four, they sample only one or multiple texture elements to determine the texture value of the pixel being rendered and can produce moire patterns or ragged transitions. The initial value of `GL_TEXTURE_MIN_FILTER` is `GL_NEAREST_MIPMAP_LINEAR`.

`GL_TEXTURE_MAG_FILTER`

The texture magnification function is used whenever the level-of-detail function used when sampling from the texture determines that the texture should be magified. It sets the texture magnification function to either `GL_NEAREST` or `GL_LINEAR` (see below). `GL_NEAREST` is generally faster than `GL_LINEAR`, but it can produce textured images with sharper edges because the transition between texture elements is not as smooth. The initial value of `GL_TEXTURE_MAG_FILTER` is `GL_LINEAR`.

`GL_NEAREST`

Returns the value of the texture element that is nearest (in Manhattan distance) to the specified texture coordinates.

`GL_LINEAR`

Returns the weighted average of the texture elements that are closest to the specified texture coordinates. These can include items wrapped or repeated from other parts of a texture, depending on the values of `GL_TEXTURE_WRAP_S` and `GL_TEXTURE_WRAP_T`, and on the exact mapping.

`GL_TEXTURE_MIN_LOD`

Sets the minimum level-of-detail parameter. This floating-point value limits the selection of highest resolution mipmap (lowest mipmap level). The initial value is -1000.

`GL_TEXTURE_MAX_LOD`

Sets the maximum level-of-detail parameter. This floating-point value limits the selection of the lowest resolution mipmap (highest mipmap level). The initial value is 1000.

`GL_TEXTURE_MAX_LEVEL`

Sets the index of the highest defined mipmap level. This is an integer value. The initial value is 1000.

`GL_TEXTURE_SWIZZLE_R`

Sets the swizzle that will be applied to the r component of a texel before it is returned to the shader. Valid values for *param* are `GL_RED`, `GL_GREEN`, `GL_BLUE`, `GL_ALPHA`, `GL_ZERO` and `GL_ONE`. If `GL_TEXTURE_SWIZZLE_R` is `GL_RED`, the value for r will be taken from the first channel of the fetched texel. If `GL_TEXTURE_SWIZZLE_R` is `GL_GREEN`, the value for r will be taken from the second channel of the fetched texel. If `GL_TEXTURE_SWIZZLE_R` is `GL_BLUE`, the value for r will be taken from the third channel of the fetched texel. If `GL_TEXTURE_SWIZZLE_R` is `GL_ALPHA`, the value for r will be taken from the fourth channel of the fetched texel. If `GL_TEXTURE_SWIZZLE_R` is `GL_ZERO`, the value for r will be subtituted with 0.0. If `GL_TEXTURE_SWIZZLE_R` is `GL_ONE`, the value for r will be subtituted with 1.0. The initial value is `GL_RED`.

`GL_TEXTURE_SWIZZLE_G`

Sets the swizzle that will be applied to the g component of a texel before it is returned to the shader. Valid values for *param* and their effects are similar to those of `GL_TEXTURE_SWIZZLE_R`. The initial value is `GL_GREEN`.

`GL_TEXTURE_SWIZZLE_B`

Sets the swizzle that will be applied to the b component of a texel before it is returned to the shader. Valid values for *param* and their effects are similar to those of `GL_TEXTURE_SWIZZLE_R`. The initial value is `GL_BLUE`.

`GL_TEXTURE_SWIZZLE_A`

Sets the swizzle that will be applied to the a component of a texel before it is returned to the shader. Valid values for *param* and their effects are similar to those of `GL_TEXTURE_SWIZZLE_R`. The initial value is `GL_ALPHA`.

`GL_TEXTURE_SWIZZLE_RGBA`

Sets the swizzles that will be applied to the r, g, b, and a components of a texel before they are returned to the shader. Valid values for *params* and their effects are similar to those of `GL_TEXTURE_SWIZZLE_R`, except that all channels are specified simultaneously. Setting the value of `GL_TEXTURE_SWIZZLE_RGBA` is equivalent (assuming no errors are generated) to setting the parameters of each of `GL_TEXTURE_SWIZZLE_R`, `GL_TEXTURE_SWIZZLE_G`, `GL_TEXTURE_SWIZZLE_B`, and `GL_TEXTURE_SWIZZLE_A` successively.

`GL_TEXTURE_WRAP_S`

Sets the wrap parameter for texture coordinate s to either `GL_CLAMP_TO_EDGE`, `GL_CLAMP_TO_BORDER`, `GL_MIRRORED_REPEAT`, `GL_REPEAT`, or `GL_MIRROR_CLAMP_TO_EDGE`. `GL_CLAMP_TO_EDGE` causes s coordinates to be clamped to the range $\left[\frac{1}{2N}, 1 - \frac{1}{2N}\right]$, where N is the size of the texture in the direction of clamping. `GL_CLAMP_TO_BORDER` evaluates s coordinates in a similar manner to GL_CLAMP_TO_EDGE. However, in cases where clamping would have occurred in GL_CLAMP_TO_EDGE mode, the fetched texel data is substituted with the values specified by GL_TEXTURE_BORDER_COLOR. GL_REPEAT causes the integer part of the s coordinate to be ignored; the GL uses only the fractional part, thereby creating a repeating pattern. GL_MIRRORED_REPEAT causes the s coordinate to be set to the fractional part of the texture coordinate if the integer part of s is even; if the integer part of s is odd, then the s texture coordinate is set to $1 - frac(s)$, where $frac(s)$ represents the fractional part of s. GL_MIRROR_CLAMP_TO_EDGE causes the s coordinate to be repeated as for GL_MIRRORED_REPEAT for one repetition of the texture, at which point the coordinate to be clamped as in GL_CLAMP_TO_EDGE. Initially, GL_TEXTURE_WRAP_S is set to GL_REPEAT.

GL_TEXTURE_WRAP_T

Sets the wrap parameter for texture coordinate t to either GL_CLAMP_TO_EDGE, GL_CLAMP_TO_BORDER, GL_MIRRORED_REPEAT, GL_REPEAT, or GL_MIRROR_CLAMP_TO_EDGE. See the discussion under GL_TEXTURE_WRAP_S. Initially, GL_TEXTURE_WRAP_T is set to GL_REPEAT.

GL_TEXTURE_WRAP_R

Sets the wrap parameter for texture coordinate r to either GL_CLAMP_TO_EDGE, GL_CLAMP_TO_BORDER, GL_MIRRORED_REPEAT, GL_REPEAT, or GL_MIRROR_CLAMP_TO_EDGE. See the discussion under GL_TEXTURE_WRAP_S. Initially, GL_TEXTURE_WRAP_R is set to GL_REPEAT.

Notes

Suppose that a program attempts to sample from a texture and has set GL_TEXTURE_MIN_FILTER to one of the functions that requires a mipmap. If either the dimensions of the texture images currently defined (with previous calls to glTexImage1D, glTexImage2D, glTexImage3D, glCopyTexImage1D, or glCopyTexImage2D) do not follow the proper sequence for mipmaps (described above), or there are fewer texture images defined than are needed, or the set of texture images have differing numbers of texture components, then the texture is considered *incomplete*.

Linear filtering accesses the four nearest texture elements only in 2D textures. In 1D textures, linear filtering accesses the two nearest texture elements. In 3D textures, linear filtering accesses the eight nearest texture elements.

glTexParameter specifies the texture parameters for the active texture unit, specified by calling glActiveTexture. **glTextureParameter** specifies the texture parameters for the texture object with ID *texture*.

GL_DEPTH_STENCIL_TEXTURE_MODE is available only if the GL version is 4.3 or greater.

GL_MIRROR_CLAMP_TO_EDGE is available only if the GL version is 4.4 or greater.

Errors

`GL_INVALID_ENUM` is generated by **glTexParameter** if *target* is not one of the accepted defined values.

`GL_INVALID_ENUM` is generated if *pname* is not one of the accepted defined values.

`GL_INVALID_ENUM` is generated if *params* should have a defined constant value (based on the value of *pname*) and does not.

`GL_INVALID_ENUM` is generated if **glTexParameter{if}** or **glTextureParameter{if}** is called for a non-scalar parameter (pname `GL_TEXTURE_BORDER_COLOR` or `GL_TEXTURE_SWIZZLE_RGBA`).

`GL_INVALID_ENUM` is generated if the effective target is either `GL_TEXTURE_2D_MULTISAMPLE` or `GL_TEXTURE_2D_MULTISAMPLE_ARRAY`, and *pname* is any of the sampler states.

`GL_INVALID_ENUM` is generated if the effective target is `GL_TEXTURE_RECTANGLE` and either of pnames `GL_TEXTURE_WRAP_S` or `GL_TEXTURE_WRAP_T` is set to either `GL_MIRROR_CLAMP_TO_EDGE`, `GL_MIRRORED_REPEAT` or `GL_REPEAT`.

`GL_INVALID_ENUM` is generated if the effective target is `GL_TEXTURE_RECTANGLE` and pname `GL_TEXTURE_MIN_FILTER` is set to a value other than `GL_NEAREST` or `GL_LINEAR` (no mipmap filtering is permitted).

`GL_INVALID_OPERATION` is generated if the effective target is either `GL_TEXTURE_2D_MULTISAMPLE` or `GL_TEXTURE_2D_MULTISAMPLE_ARRAY`, and pname `GL_TEXTURE_BASE_LEVEL` is set to a value other than zero.

`GL_INVALID_OPERATION` is generated by **glTextureParameter** if texture is not the name of an existing texture object.

`GL_INVALID_OPERATION` is generated if the effective target is `GL_TEXTURE_RECTANGLE` and pname `GL_TEXTURE_BASE_LEVEL` is set to any value other than zero.

`GL_INVALID_VALUE` is generated if *pname* is `GL_TEXTURE_BASE_LEVEL` or `GL_TEXTURE_MAX_LEVEL`, and *param* or *params* is negative.

Associated Gets

glGetTexParameter

glGetTextureParameter

glGetTexLevelParameter

glGetTextureLevelParameter

Version Support

Function / Feature Name	OpenGL Version											
	2.0	2.1	3.0	3.1	3.2	3.3	4.0	4.1	4.2	4.3	4.4	4.5
glTexParameterIiv	-	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
glTexParameterIuiv	-	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
glTexParameterf	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

	OpenGL Version											
Function / Feature Name	2.0	2.1	3.0	3.1	3.2	3.3	4.0	4.1	4.2	4.3	4.4	4.5
glTexParameterfv	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
glTexParameteri	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
glTexParameteriv	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
glTextureParameterIiv	-	-	-	-	-	-	-	-	-	-	-	✓
glTextureParameterIuiv	-	-	-	-	-	-	-	-	-	-	-	✓
glTextureParameterf	-	-	-	-	-	-	-	-	-	-	-	✓
glTextureParameterfv	-	-	-	-	-	-	-	-	-	-	-	✓
glTextureParameteri	-	-	-	-	-	-	-	-	-	-	-	✓
glTextureParameteriv	-	-	-	-	-	-	-	-	-	-	-	✓

See Also

glActiveTexture, glBindTexture, glCopyTexImage1D, glCopyTexImage2D, glCopyTexSubImage1D, glCopyTexSubImage2D, glCopyTexSubImage3D, glPixelStore, glSamplerParameter, glTexImage1D, glTexImage2D, glTexImage3D, glTexSubImage1D, glTexSubImage2D, glTexSubImage3D

Copyright