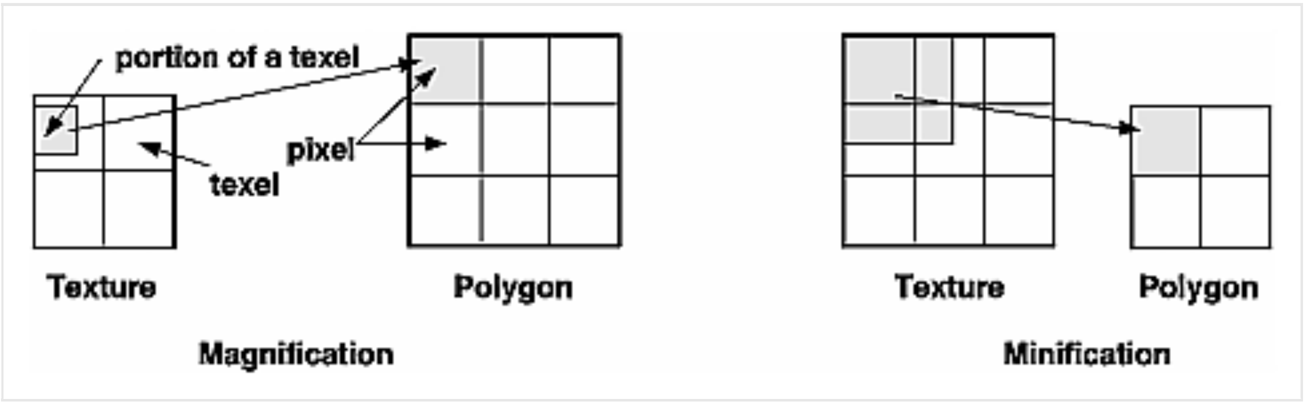
Texture Parameter and Filtering

Texture maps are square or rectangular, but after being mapped to a polygon or surface and transformed into screen coordinates, the individual texels of a texture rarely correspond to individual pixels of the final screen image.



Depending on the transformations used and the texture mapping applied, a single pixel on the screen can correspond to anything from a tiny portion of a texel (**magnification**) to a large collection of texels (**minification**).

In either case, it is unclear exactly which texel values should be used and how they should be averaged or interpolated.

In some cases, it is not obvious whether magnification or minification is called for. If the texture needs to be stretched (or shrunk) in both the *x* and *y* directions, then magnification (or minification) is needed. If the mipmap needs to be stretched in one direction and shrunk in the other, OpenGL makes a choice between magnification and minification that in most cases gives the best result possible. It's best to try to avoid these situations by using texture coordinates that map without such distortion.

OpenGL allows you to specify any of several filtering options to determine these calculations. The options provide different trade-offs between speed and image quality.

The following lines are examples of how to use **glTexParameter*()** to specify the magnification and minification filtering methods:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
                GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
                GL_NEAREST);
```

The first argument to **glTexParameter*()** is either GL_TEXTURE_2D or GL_TEXTURE_1D, the second argument is either GL_TEXTURE_MAG_FILTER or GL_TEXTURE_MIN_FILTER to indicate whether you're specifying the filtering method for magnification or minification. The third argument specifies the filtering method.

Filtering Methods for Magnification and Minification

Parameter	Values
GL_TEXTURE_MAG_FILTER	GL_NEAREST GL_LINEAR
GL_TEXTURE_MIN_FILTER	GL_NEAREST GL_LINEAR GL_NEAREST_MIPMAP_NEAREST GL_NEAREST_MIPMAP_LINEAR GL_LINEAR_MIPMAP_NEAREST GL_LINEAR_MIPMAP_LINEAR

If you choose GL_NEAREST, the texel with coordinates nearest the center of the pixel is used for both magnification and minification. This can result in aliasing artifacts (sometimes severe).

If you choose GL_LINEAR, a weighted linear average of the 2x2 array of texels that lie nearest to the center of the pixel is used, again for both magnification and minification. When the texture coordinates are near the edge of the texture map, the nearest 2x2 array of texels might include some that are outside the texture map. In these cases, the texel values used depend on whether GL_REPEAT or GL_CLAMP is in effect and whether you've assigned a border for the texture.

GL_NEAREST requires less computation than GL_LINEAR and therefore might execute more quickly, but GL_LINEAR provides smoother results.

With magnification, even if you've supplied mipmaps, the largest texture map (*level* = 0) is always used. With minification, you can choose a filtering method that uses the most appropriate one or two mipmaps, as described in the next paragraph. (If GL_NEAREST or GL_LINEAR is specified with minification, the largest texture map is used.)

As shown in the Table, four additional filtering choices are available when minifying with mipmaps. Within an individual mipmap, you can choose the nearest texel value with GL_NEAREST_MIPMAP_NEAREST, or you can interpolate linearly by specifying GL_LINEAR_MIPMAP_NEAREST. Using the nearest texels is faster but yields less desirable results. The particular mipmap chosen is a function of the amount of minification required, and there is a cutoff point from the use of one particular mipmap to the next. To avoid a sudden transition, use GL_NEAREST_MIPMAP_LINEAR or GL_LINEAR_MIPMAP_LINEAR to linearly interpolate texel values from the two nearest best choices of mipmaps. GL_NEAREST_MIPMAP_LINEAR selects the nearest texel in each of the two maps and then interpolates linearly between these two values. GL_LINEAR_MIPMAP_LINEAR uses linear interpolation to compute the value in each of two maps and then interpolates linearly between these two values. As you might expect, GL_LINEAR_MIPMAP_LINEAR generally produces the smoothest results, but it requires the most computation and therefore might be the slowest.