NAME

glBlendFunc – specify pixel arithmetic

C SPECIFICATION

void **glBlendFunc**(GLenum *sfactor*,
GLenum *dfactor*)

delim $$

PARAMETERS

sfactor Specifies how the red, green, blue, and alpha source blending factors are computed. Nine symbolic constants are accepted: **GL_ZERO**, **GL_ONE**, **GL_DST_COLOR**, **GL_ONE_MINUS_DST_COLOR**, **GL_SRC_ALPHA**, **GL_ONE_MINUS_SRC_ALPHA**, **GL_DST_ALPHA**, **GL_ONE_MINUS_DST_ALPHA**, and **GL_SRC_ALPHA_SATURATE**. The initial value is **GL_ONE**.

dfactor Specifies how the red, green, blue, and alpha destination blending factors are computed. Eight symbolic constants are accepted: **GL_ZERO**, **GL_ONE**, **GL_SRC_COLOR**, **GL_ONE_MINUS_SRC_COLOR**, **GL_SRC_ALPHA**, **GL_ONE_MINUS_SRC_ALPHA**, **GL_DST_ALPHA**, and **GL_ONE_MINUS_DST_ALPHA**. The initial value is **GL_ZERO**.

DESCRIPTION

In RGBA mode, pixels can be drawn using a function that blends the incoming (source) RGBA values with the RGBA values that are already in the frame buffer (the destination values). Blending is initially disabled. Use **glEnable** and **glDisable** with argument **GL_BLEND** to enable and disable blending.

glBlendFunc defines the operation of blending when it is enabled. *sfactor* specifies which of nine methods is used to scale the source color components. *dfactor* specifies which of eight methods is used to scale the destination color components. The eleven possible methods are described in the following table. Each method defines four scale factors, one each for red, green, blue, and alpha.

In the table and in subsequent equations, source and destination color components are referred to as $(R sub s , G sub s , B sub s , A sub s)$ and $(R sub d , G sub d , B sub d , A sub d)$. They are understood to have integer values between 0 and $(k sub R , k sub G , k sub B , k sub A)$, where

$k sub c ~=~ 2 sup m sub c - 1$

and $(m sub R , m sub G , m sub B , m sub A)$ is the number of red, green, blue, and alpha bitplanes.

Source and destination scale factors are referred to as $(s sub R , s sub G , s sub B , s sub A)$ and $(d sub R , d sub G , d sub B , d sub A)$. The scale factors described in the table, denoted $(f sub R , f sub G , f sub B , f sub A)$, represent either source or destination factors. All scale factors have range [0,1].

parameter	*$(f sub R , ~~ f sub G , ~~ f sub B , ~~ f sub A)$*
GL_ZERO	$(0, ~0, ~0, ~0)$
GL_ONE	$(1, ~1, ~1, ~1)$
GL_SRC_COLOR	$(R sub s / k sub R , ~G sub s / k sub G , ~B sub s / k sub B , ~A sub s / k sub A)$
GL_ONE_MINUS_SRC_COLOR	$(1, ~1, ~1, ~1) ~-~ (R sub s / k sub R , ~G sub s / k sub G , ~B sub s / k sub B , ~A sub s / k sub A)$
GL_DST_COLOR	$(R sub d / k sub R , ~G sub d / k sub G , ~B sub d / k sub B , ~A sub d / k sub A)$
GL_ONE_MINUS_DST_COLOR	$(1, ~1, ~1, ~1) ~-~ (R sub d / k sub R , ~G sub d / k sub G , ~B sub d / k sub B , ~A sub d / k sub A)$
GL_SRC_ALPHA	$(A sub s / k sub A , ~A sub s / k sub A , ~A sub s / k sub A , ~A sub s / k sub A)$
GL_ONE_MINUS_SRC_ALPHA	$(1, ~1, ~1, ~1) ~-~ (A sub s / k sub A , ~A sub s / k sub A , ~A sub s / k sub A , ~A sub s / k sub A)$
GL_DST_ALPHA	$(A sub d / k sub A , ~A sub d / k sub A , ~A sub d / k sub A , ~A sub d / k sub A)$
GL_ONE_MINUS_DST_ALPHA	$(1, ~1, ~1, ~1) ~-~ (A sub d / k sub A , ~A sub d / k sub A , ~A sub d / k sub A , ~A sub d / k sub A)$
GL_SRC_ALPHA_SATURATE	$(i, ~i, ~i, ~1)$

In the table,

$i ~=~ min (A sub s , ~k sub A - A sub d) ~/~ k sub A$

To determine the blended RGBA values of a pixel when drawing in RGBA mode, the system uses the following equations:

$R sub d ~=~ min (k sub R , ~~ R sub s s sub R + R sub d d sub R)$
$G sub d ~=~ min (k sub G , ~~ G sub s s sub G + G sub d d sub G)$
$B sub d ~=~ min (k sub B , ~~ B sub s s sub B + B sub d d sub B)$
$A sub d ~=~ min (k sub A , ~~ A sub s s sub A + A sub d d sub A)$

Despite the apparent precision of the above equations, blending arithmetic is not exactly specified, because blending operates with imprecise integer color values. However, a blend factor that should be equal to 1 is guaranteed not to modify its multiplicand, and a blend factor equal to 0 reduces its multiplicand to 0. For example, when *sfactor* is **GL_SRC_ALPHA**, *dfactor* is **GL_ONE_MINUS_SRC_ALPHA**, and $A sub s$ is equal to $k sub A$, the equations reduce to simple replacement:

$R sub d ~=~ R sub s$
$G sub d ~=~ G sub s$
$B sub d ~=~ B sub s$
$A sub d ~=~ A sub s$

EXAMPLES

Transparency is best implemented using blend function (**GL_SRC_ALPHA**, **GL_ONE_MINUS_SRC_ALPHA**) with primitives sorted from farthest to nearest. Note that this transparency calculation does not require the presence of alpha bitplanes in the frame buffer.

Blend function (**GL_SRC_ALPHA**, **GL_ONE_MINUS_SRC_ALPHA**) is also useful for rendering antialiased points and lines in arbitrary order.

Polygon antialiasing is optimized using blend function (**GL_SRC_ALPHA_SATURATE**, **GL_ONE**) with polygons sorted from nearest to farthest. (See the **glEnable**, **glDisable** reference page and the **GL_POLYGON_SMOOTH** argument for information on polygon antialiasing.) Destination alpha bitplanes, which must be present for this blend function to operate correctly, store the accumulated coverage.

NOTES

Incoming (source) alpha is correctly thought of as a material opacity, ranging from 1.0 ($K sub A$), representing complete opacity, to 0.0 (0), representing complete
transparency.

When more than one color buffer is enabled for drawing, the GL performs blending separately for each enabled buffer, using the contents of that buffer for destination color. (See **glDrawBuffer**.)

Blending affects only RGBA rendering. It is ignored by color index renderers.

ERRORS

GL_INVALID_ENUM is generated if either *sfactor* or *dfactor* is not an accepted value.

GL_INVALID_OPERATION is generated if **glBlendFunc** is executed between the execution of **glBegin** and the corresponding execution of **glEnd**.

ASSOCIATED GETS

glGet with argument **GL_BLEND_SRC**
glGet with argument **GL_BLEND_DST**
glIsEnabled with argument **GL_BLEND**

SEE ALSO

glAlphaFunc, **glClear**, **glDrawBuffer**, **glEnable**, **glLogicOp**, **glStencilFunc**