NAME

Parc - Draws a circular arc

SPECIFICATION

void Parc(Fcoord x, Fcoord y, Fcoord z, Fcoord radius, Angle start, Angle end)

ARGUMENTS

x	The x coordinate of the center of the arc
y	The y coordinate of the center of the arc
z	The z coordinate of the center of the arc
radius	The radius of the arc
start	The starting angle of the arc, 0 is along the x axis
end	The ending angle of the arc

RETURNS

Nothing

DESCRIPTION

Parc draws a circular arc where *x*, *y*, *z* is the origin of the arc, *start*, *end* define the starting and ending position of the arc, and *radius* defines the radius of the arc. The color is taken from the current line color attribute.

Positive angles are measured clockwise from the positive x axis, negative angles are measured counterclockwise. If any angle is greater than 360, then an effective angle of *a = a mod 360* is used.

The number of line segments used to draw the arc is set with **Pset_precision**.

NOTE

An arc is a line primitive.

SEE ALSO

Parc, Pcircle, Pcircle_fill, Pline_color, Pset_precision

NAME

Parc_fill - Draws a filled circular arc

SPECIFICATION

void Parc_fill(Fcoord x, Fcoord y, Fcoord z, Fcoord radius, Angle start, Angle end)

ARGUMENTS

x	The x coordinate of the center of the arc
y	The y coordinate of the center of the arc
z	The z coordinate of the center of the arc
radius	The radius of the arc
start	The starting angle of the arc, 0 is along the x axis
end	The ending angle of the arc

RETURNS

Nothing

DESCRIPTION

Parc_fill draws a filled circular arc where *x*, *y, z* is the origin of the arc, *start*, *end* define the starting and ending position of the arc, and *radius* defines the radius of the arc. The color is taken from the current color attribute.
Positive angles are measured clockwise from the positive x axis, negative angles are measured counterclockwise. If any angle is greater than 360, then an effective angle of *a = a mod 360* is used.
The number of line segments used to draw the arc is set with **Pset_precision**.

NOTE

An filled arc is a polygon primitive.

SEE ALSO

Parc, Pcircle, Pcircle_fill, Pcolor, Pset_precision

NAME

Pbackface - Turns backfacing polygon removal on or off

SPECIFICATION

void Pbackface(int mode)

ARGUMENTS

mode	A flag to turn the backface/frontface culling on or off.

RETURNS

Nothing

DESCRIPTION

Pbackface sets the backface culling mode. Valid modes are:

PEXtk_CULL_BACKFACE
PEXtk_CULL_FRONTFACE
PEXtk_CULL_NONE

If *mode* is set to PEXtk_CULL_BACKFACE, then polygon backface removal is activated. In this mode, any polygon that has a polygon normal pointing away from the eye/viewing position is not drawn. If *mode* is set to PEXtk_CULL_FRONTFACE, then any polygon that has a polygon normal pointing towards the eye/viewing position is not drawn. If *mode* is set to PEXtk_CULL_NONE, then polygon backface/frontface removal is deactivated. The default mode is PEXtk_CULL_BACKFACE.

SEE ALSO

Pget_backface, Pzbuffer

NAME

Pbegin_line - Initiates the use of vertex calls as polylines

SPECIFICATION

void Pbegin_line(void)

ARGUMENTS

None

RETURNS

Nothing

DESCRIPTION

Pbegin_line puts the library into a mode where any subsequent **Pvertex** calls are interpreted as defining a polyline. The call **Pend_line** ends this mode.

SEE ALSO

Pend_line, Pvertex, Pvertex_color

NAME

Pbegin_poly - Initiates the use of the vertex calls as polygon vertices

SPECIFICATION

void Pbegin_poly(int style)

ARGUMENTS

style	The fill style to use for the polygon.

RETURNS

Nothing

DESCRIPTION

Pbegin_poly puts the library into a mode where any subsequent **Pvertex** calls are interpreted as defining a polygon. The call **Pend_poly** ends this mode.
The *style* may be set to PEXtk_FILL_SOLID to create a filled polygon, or set to PEXtk_FILL_HOLLOW to create a non-filled polygon.

SEE ALSO

Pend_poly, Ppoly_normal, Pvertex, Pvertex_color, Pvertex_normal

NAME

Pbegin_tmesh - Initiates the use of vertex calls as triangle mesh vertices

SPECIFICATION

void Pbegin_tmesh(int style)

ARGUMENTS

style	The fill style to use for the triangle mesh.

RETURNS

Nothing

DESCRIPTION

Pbegin_tmesh puts the library into a mode where any subsequent **Pvertex** calls are interpreted as defining a triangle mesh. After the first two vertices, subsequent calls to **Pvertex** define a triangle whose first two vertices are the preceding two vertices. The call **Pend_tmesh** ends this mode.
The triangle mesh will be composed of *n*-2 triangles, where *n* is the number of vertices. The normal for the triangle, if not specified using **Ppoly_normal**, will be calculated the same as for a polygon for the odd-numbered triangles. For the even numbered triangles, the vertex order is reversed, and the normal will be calculated the same as for a polygon.
The *style* may be set to PEXtk_FILL_SOLID to create a filled triangle mesh, or set to PEXtk_FILL_HOLLOW to create a non-filled triangle mesh

SEE ALSO

Pend_tmesh, Ppoly_normal, Pvertex, Pvertex_color, Pvertex_normal

NAME

Pborder - Sets the geometry outlining attribute

SPECIFICATION

void Pborder(int type)

ARGUMENTS

type	The type of border to use for all subsequent polygons.

RETURNS

Nothing

DESCRIPTION

Pborder sets the *type* of outline for a polygon. The possible border types are:

PEXtk_BORDER_NONE
PEXtk_BORDER_SOLID
PEXtk_BORDER_DASHED
PEXtk_BORDER_DOTTED
PEXtk_BORDER_DOTDASH

The default border type is PEXtk_BORDER_NONE.

NOTE

Some border types may not be available on all platforms.

SEE ALSO

Pline_type, Pget_border, Ppoly, Ppoly_fill, Ppoly_shade

NAME

Pbuffer_mode - Set single or double buffer mode

SPECIFICATION

void Pbuffer_mode(int mode)

ARGUMENTS

mode	An integer representing the buffering mode

RETURNS

Nothing

DESCRIPTION

Pbuffer_mode puts the system into single or double buffer mode. To switch to single buffer mode, set *mode* to PEXtk_SINGLE_BUFFER, and to switch to double buffer mode set *mode* to PEXtk_DOUBLE_BUFFER.
If a back buffer does not already exist when **Pbuffer_mode** is called with *mode* set to PEXtk_DOUBLE_BUFFER, then PEXtk will create a back buffer. The default setting of creating the back buffer using the Multi-Buffering Extension will be used if it is available, or by creating a Pixmap if it is not. If the default has been overridden by calling **Pset_function** with a type of PEXtk_FUNC_CREATE_BUFFER, then that routine will be called instead of using the default routine.

NOTE

Single buffer mode allows updates to the window to be seen almost instantly, but animation looks jumpy. Double buffer mode makes animation look smooth, but updates to the window will not be seen until a **Pswap_buffers** command is given.

SEE ALSO

Pset_display_buffer, Pset_function, Pset_update_buffer, Pswap_buffers

NAME

Pcall_display_list - Execute a display list and save the state

SPECIFICATION

void Pcall_display_list(int name)

ARGUMENTS

name	An integer representing the display list name

RETURNS

Nothing

DESCRIPTION

Pcall_display_list executes the commands stored in the display list *name*.
A **Pcall_display_list** may be inserted into a display list before the display list *name* has been defined. However, the display list must be defined when the parent display list is traversed, otherwise an error is generated.
If this is called within another display list, the command is just inserted into that display list, thus building a hierarchy.
If this is called outside of a display list it will cause the named display list and all the display lists it calls (i.e., its children), to be traversed. The current update buffer will receive the resulting output. If the system is in double buffer mode, no changes to the window will be seen until **Pswap_buffers** is called.

SEE ALSO

Pclose_display_list, Pcreate_display_list, Pdelete_display_list

NAME

Pcall_struct - Execute a structure and save state
Pexec_struct - Execute a structure

SPECIFICATION

void Pcall_struct(int name)
void Pexec_struct(int name)

ARGUMENTS

name	An integer representing the structure name

RETURNS

Nothing

DESCRIPTION

Pcall_struct saves the state information, executes the commands stored in the structure *name*, and then restores the state after it returns.

A **Pcall_struct** may be inserted into a structure before the structure *name* has been defined. However, the structure must be defined when the parent structure is traversed, otherwise an error is generated.

If this is called within another structure, the command is just inserted into that structure, thus building a hierarchy.

If this is called outside of a structure it will cause the named structure and all the structures it calls (i.e., its children), to be traversed. The current update buffer will receive the resulting output. If the system is in double buffer mode, no changes to the window will be seen until **Pswap_buffers** is called.

Pexec_struct is identical to **Pcall_struct** except that no state information is saved before the call. The **Ppush_matrix** and **Ppop_matrix** routines should be used to limit the scope of any attribute or matrix changes within the structure being called by **Pexec_struct**.

NOTE

Pexec_struct is not available for PEXtk Version 1.0.

SEE ALSO

Pcreate_struct, Pclose_struct, Ppush_attributes, Ppop_attributes, Ppush_viewport, Ppop_viewport, Ppush_matrix, Ppop_matrix, Pswap_buffers, Pset_buffer_mode

NAME

Pcircle - Draws a circle

SPECIFICATION

```
void Pcircle(Fcoord x, Fcoord y, Fcoord z, Fcoord radius)
```

ARGUMENTS

x	The x coordinate of the center of the circle
y	The y coordinate of the center of the circle
z	The z coordinate of the center of the circle
radius	The radius of the circle

RETURNS

Nothing

DESCRIPTION

Pcircle draws an outlined circle where *x*, *y*, *z* is the center of the circle, and the radius is defined by *radius*. The color is taken from the current line color attribute.
The number of line segments used to draw the circle is set with **Pset_precision**.

NOTE

A circle is a line primitive.

SEE ALSO

Parc, Parc_fill, Pcircle_fill, Pline_color, Pset_precision

NAME

Pcircle_fill - Draws a filled circle

SPECIFICATION

```
void Pcircle_fill(Fcoord x, Fcoord y, Fcoord z, Fcoord radius)
```

ARGUMENTS

x	The x coordinate of the center of the circle
y	The y coordinate of the center of the circle
z	The z coordinate of the center of the circle
radius	The radius of the circle

RETURNS

Nothing

DESCRIPTION

Pcircle_fill draws a filled circle where *x*, *y*, *z* is the center of the circle, and the radius is defined by *radius*. The color is taken from the current color attribute.
The number of polygon segments used to draw the circle is set with **Pset_precision**.

NOTE

A filled circle is a polygon primitive.

SEE ALSO

Parc, Parc_fill, Pcircle, Pcolor, Pset_precision

NAME

Pclear - Clear current viewport to color
Pclear_ind - Clear current viewport to color index
Pclear_hsv - Clear current viewport to HSV color
Pclear_rgb - Clear current viewport to RGB color

SPECIFICATION

```
void Pclear(PCOLOR *col)
void Pclear_ind(int colind)
void Pclear_hsv(Fdata hue, Fdata saturation, Fdata value)
void Pclear_rgb(Fdata red, Fdata green, Fdata blue)
```

ARGUMENTS

**col*	A PCOLOR structure containing the RGB, HSV or CIE values
colind	An integer representing the color index
hue	A float representing the color hue
saturation	A float representing the color saturation
value	A float representing the color value
red,green,blue	Floats representing the RGB colors

RETURNS

Nothing

DESCRIPTION

Pclear clears the current viewport to the color specified by the color *col*.
Pclear_ind clears the current viewport to the color specified by the color index *colind*.
Pclear_hsv clears the current viewport to the HSV color specified by *hue*, *saturation*, *value* where *hue* is between 0.0 and 360.0, *saturation* is between 0.0 and 1.0 and *value* is between 0.0 and 1.0.
Pclear_rgb clears the current viewport to the RGB color specified by *red*, *green*, *blue* with values ranging between 0.0 and 1.0.

NOTE

The color table must be set with **Pset_color_table** before the indices are referenced by the **Pclear_ind** routine.
If zbuffer mode is on, then the z buffer is also cleared in the current viewport.

SEE ALSO

Pset_color_table

NAME

Pclip_rectangle - Specify the clipping rectangle

SPECIFICATION

void Pclip_rectangle(Screencoord left, Screencoord right, Screencoord bottom, Screencoord top)

ARGUMENTS

left	The clip rectangle lower left x coordinate
right	The clip rectangle upper right x coordinate
bottom	The clip rectangle bottom left y coordinate
top	The clip rectangle top right y coordinate

RETURNS

Nothing

DESCRIPTION

Pclip_rectangle sets the clipping rectangle to *left*, *right*, *bottom*, *top* which are in X window relative coordinates, and must be a subregion of the current viewport to have any effect. All geometry written to the window will be clipped by this rectangle. The clip rectangle is usually set automatically by a call to **Pviewport** which sets the clip rectangle to match the viewport.

NOTE

Since the clip rectangle must be a a subregion of the current viewport, the **Pviewport** call must precede the **Pclip_rectangle** call.

SEE ALSO

Pviewport

NAME

Pclose_display_list - Close currently open display list

SPECIFICATION

void Pclose_display_list(void)

ARGUMENTS

None

RETURNS

Nothing

DESCRIPTION

Pclose_display_list closes the currently open display list. All routines other than geometry routines called following a **Pcreate_display_list** and before **Pclose_display_list** become part of the display list that was named as the argument passed to the **Pcreate_display_list** routine. The commands are only executed when that display list is called by **Pcall_display_list**.
An error will be generated if no display list is currently open.

NOTE

This routine can only be used inside a display list.
Only one display list may be open at a time.
A display list can not currently be edited.

SEE ALSO

Pcall_display_list, Pcreate_display_list

NAME

Pclose_struct - Close currently open structure

SPECIFICATION

void Pclose_struct(void)

ARGUMENTS

None

RETURNS

Nothing

DESCRIPTION

Pclose_struct closes the currently open structure. All geometry routines called following a **Pcreate_struct** and before **Pclose_struct** become part of the structure that was named as the argument passed to the **Pcreate_struct** routine. The commands are only executed when that structure is called by **Pexec_struct** or **Pcall_struct**.
Pclose_struct also closes a structure that has been opened for editing by the **Pedit_struct** routine.
An error will be generated if no structure is currently open.

NOTE

This routine can only be used inside a structure.
Only one structure may be open at a time.
Any changes made to the structure will not be reflected on the display until the structure is traversed again.

SEE ALSO

Pcall_struct, Pcreate_struct, Pedit_struct, Preplace_struct

NAME

Pcolor - Set current filled polygon color
Pcolor_ind - Set current filled polygon to index
Pcolor_hsv - Set current filled polygon color to HSV
Pcolor_rgb - Set current filled polygon color to RGB

SPECIFICATION

```
void Pcolor(PCOLOR *col)
void Pcolor_ind(int colind)
void Pcolor_hsv(Fdata hue, Fdata saturation, Fdata value)
void Pcolor_rgb(Fdata red, Fdata green, Fdata blue)
```

ARGUMENTS

**col*	Pointer to a PCOLOR structure containing the RGB, HSV, or CIE values
colind	An integer representing the color index
hue	A float representing the color hue
saturation	A float representing the color saturation
value	A float representing the color value
red,green,blue	Floats representing the RGB colors

RETURNS

Nothing

DESCRIPTION

Pcolor sets the polygon color to *col* for all polygons that follow.
Pcolor_ind sets the polygon color to the index *colind*.
Pcolor_hsv sets the polygon color specified by *hue*, *saturation* and *value* where *hue* is between 0.0 and 360.0, *saturation* is between 0.0 and 1.0 and *value* is between 0.0 and 1.0.
Pcolor_rgb sets the polygon color specified by *red*, *green*, *blue* with values ranging between 0.0 and 1.0.

NOTE

This affects all types of polygons that follow, excluding Gouraud shaded.
The color table must be set with **Pset_color_table** before the indices are referenced by the **Pcolor_ind** routine.
The current polygon color is an attribute and is saved by **Ppush_attributes** and restored with **Ppop_attributes**.

SEE ALSO

Pline_color, Pset_color_table

NAME

Pcreate_display_list - Create a display list

SPECIFICATION

int Pcreate_display_list(int name)

ARGUMENTS

name	An integer representing the display list name

RETURNS

0 if successful, non-zero if an error occurred

DESCRIPTION

Pcreate_display_list starts a display list named *name*. All routines other than geometry routines called before a **Pclose_display_list** are stored in memory for later execution. A display list may be referenced before it has been created, so long as it exists when the display list is traversed.
Unless otherwise noted most non-geometry commands may appear in a display list.
An error is generated if the display list already exists, or a display list is already open.

NOTE

This routine cannot be used inside a structure or display list. The display list name may be any value other than 0.
A display list can not currently be edited.

SEE ALSO

Pcall_display_list, Pclose_display_list, Pdelete_display_list, Pdisplay_list_used

NAME

Pcreate_struct - Create a structure

SPECIFICATION

```
int Pcreate_struct(int name)
```

ARGUMENTS

name	An integer representing the structure name

RETURNS

0 if successful, non-zero if an error occurred

DESCRIPTION

Pcreate_struct starts a structure named *name*. All geometry routines called before a **Pclose_struct** are stored in memory for later execution. A structure may be referenced before it has been created, so long as it exists when the structure is traversed.
Unless otherwise, noted most geometry commands may appear in a structure.
An error is generated if the structure already exists, or a structure is already open.

NOTE

This routine cannot be used inside a structure or display list. The structure *name* may be any name other than -1.

SEE ALSO

Pcall_struct, Pclose_struct, Pedit_struct, Preplace_struct, Pstruct_used

NAME

Pcurve_approx_method - Sets curve approximation method and tolerance

SPECIFICATION

void Pcurve_approx_method(int meth, Fdata tol)

ARGUMENTS

meth	The curve approximation method to be used when tessellating curve primitives
tol	The curve tolerance to be used when tessellating curve

RETURNS

Nothing

DESCRIPTION

Pcurve_approx_method sets the curve approximation method and approximation tolerance criteria to be used when tessellating **Pnurb_curve** primitives. Available methods are:

PEXtk_APPROX_IMP_DEP	This technique will vary from one platform to another, but it is always available. It is usually the most efficient method.
PEXtk_APPROX_CONSTANT_KNOTS	Curve is tessellated with equal parametric increments between successive pairs of knots. Only the integer portion of the tolerance is used. If *tol* <= 0, the curve is evaluated at the parameter limits and at those knots within the parameter limits.If *tol* > 0, the curve is evaluated at the parameter limits, at the knots within the parameter limits, and at the number of intervals specified by *tol* between each pair of knots.
PEXtk_APPROX_WC_CHORD_LEN	Curve is tessellated until the length of each line segment in world coordinates is < *tol*.
PEXtk_APPROX_NC_CHORD_LEN	Curve is tessellated until the length of each line segment in normalized projection coordinates is < *tol*.
PEXtk_APPROX_DC_CHORD_LEN	Curve is tessellated until the length of each line segment in device coordinates is < *tol*.
PEXtk_APPROX_WC_CHORD_DEV	Curve is tessellated until the maximum deviation in world coordinates is < *tol*.
PEXtk_APPROX_NC_CHORD_DEV	Curve is tessellated until the maximum deviation in normalized projection coordinates is < *tol*.
PEXtk_APPROX_DC_CHORD_DEV	Curve is tessellated until the maximum deviation in device coordinates is < *tol*.
PEXtk_APPROX_WC_RELATIVE	Curve is tessellated using a relative level of quality in world coordinates.
PEXtk_APPROX_NC_RELATIVE	Curve is tessellated using a relative level of quality in normalized projection coordinates.
PEXtk_APPROX_DC_RELATIVE	Curve is tessellated using a relative level of quality in device coordinates.

The current curve approximation method and tolerance is an attribute and is saved by **Ppush_attributes** and restored with **Ppop_attributes**.

SEE ALSO

Pnurb_curve, Pnurb_surface, Psurface_approx_method

NAME

Pdelete_display_list - Deletes a display list

SPECIFICATION

int Pdelete_display_list(int name)

ARGUMENTS

name	An integer representing the display list name

RETURNS

0 if successful, 1 if an error occurred

DESCRIPTION

Pdelete_display_list deletes the display list *name*, freeing all the memory it occupied.
The *name* is also freed to be used by a new display list.
An error will be generated if the display list does not exist.

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pcall_display_list, Pclose_display_list, Pdelete_display_list, Pdisplay_list_used

NAME

Pdelete_struct - Deletes a structure

SPECIFICATION

int Pdelete_struct(int name)

ARGUMENTS

name	An integer representing the structure name

RETURNS

0 if successful, 1 if an error occurred

DESCRIPTION

Pdelete_struct deletes the structure *name*, freeing all the memory it occupied.
The *name* is also freed to be used by a new structure.
An error will be generated if the structure does not exist.

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pedit_struct, Pcreate_struct, Preplace_struct

NAME

Pdepth - Set the near and far z depths

SPECIFICATION

void Pdepth(Fcoord near, Fcoord far)

ARGUMENTS

near	A float representing the near clipping plane
far	A float representing the far clipping plane

RETURNS

Nothing

DESCRIPTION

Pdepth sets the minimum and maximum z of the current scene to *near* and *far*.

SEE ALSO

Portho_view, Ppersp_view, Pwindow_view

NAME

Pdepth_cue - Turns depth cue on or off and sets min and max range

SPECIFICATION

void Pdepth_cue(int flag, Fcoord rmin, Fcoord rmax)

ARGUMENTS

flag	A flag to turn depth cueing on or off
rmin	A value specifying the minimum depth cue range
rmax	A value specifying the maximum depth cue range

RETURNS

Nothing

DESCRIPTION

Pdepth_cue turns depth cueing on or off. If *flag* is FALSE, then depth cueing is turned off. If *flag* is TRUE, then depth cueing is turned on, and the minimum and maximum depth to be used in depth cueing is set to *rmin* and *rmax*. The range for *rmin* and *rmax* is 0.0 to 1.0.

SEE ALSO

Portho_view, Ppersp_view, Pwindow_view

NAME

Pdither - Turns dithering on or off

SPECIFICATION

void Pdither(int mode, int pattern)

ARGUMENTS

mode	A flag to turn dithering on or off
pattern	An integer representing the dithering pattern to use

RETURNS

Nothing

DESCRIPTION

Pdither turns dithering on or off. The use of **Pdither** will help reduce the effect of “Mach banding”. If *mode* is 0, then dithering is turned off. If *mode* is greater than 0, then dithering is turned on. The value for *mode* represents the size of the dither pattern. The *pattern* specifies which dithering pattern to use. For example, if *mode* is three and *pattern* is two, then pattern type two for 3x3 dithering is used.

NOTE

Support for dithering and the dithering pattern is platform dependent.
This routine cannot be used inside a structure.

SEE ALSO

Ppoly, Ppoly_shade, Ppoly_shade_normal, Ppoly_shade_texture, Ppoly_texture

NAME

Pdisplay_list_used - Indicates whether a display list exists

SPECIFICATION

Boolean Pdisplay_list_used(int name)

ARGUMENTS

name	An integer representing the display list name

RETURNS

0 if display list name is not in use, 1 if the display list name is in use

DESCRIPTION

Pdisplay_list_used returns TRUE (1) if the display list *name* is currently in use. Otherwise it returns FALSE (0).

NOTE

This routine cannot be used inside a structure or display list.This routine cannot be used inside a structure or display list.

SEE ALSO

Pcreate_display_list

NAME

Pdraw_2d - Draw a 2D line

SPECIFICATION

void Pdraw_2d(Fcoord x, Fcoord y)

ARGUMENTS

x	An Fcoord representing the x position
y	An Fcoord representing the y position

RETURNS

Nothing

DESCRIPTION

Pdraw_2d draws a line from the current draw position to *x* and *y*, and updates the current draw position. The color of the line is taken from the current line color attribute set by **Pline_color**.

SEE ALSO

Pline_color, Pmove_2d, Ppoint_2d, Portho_2d_view

NAME

Pdraw_3d - Draw a 3D line

SPECIFICATION

void Pdraw_3d(Fcoord x, Fcoord y, Fcoord z)

ARGUMENTS

x	A Fcoord representing the x position
y	A Fcoord representing the y position
z	A Fcoord representing the z position

RETURNS

Nothing

DESCRIPTION

Pdraw_3d draws a line from the current draw position to *x*, *y* and *z*, and updates the current draw position. The color of the line is taken from the current line color attribute set by **Pline_color**.

SEE ALSO

Pline_color, Pmove_3d, Ppoint_3d

NAME

Pedit_multiply_matrix - Multiplies a matrix in a structure

SPECIFICATION

int Pedit_multiply_matrix(int name, int label, Matrix mat, int postflag)

ARGUMENTS

name	The structure name the matrix is to be applied to
label	An integer representing the label to be applied
mat	The matrix to be applied
postflag	Flag set to TRUE to perform a postconcatenation

RETURNS

0 if successful, non-zero if in incorrect mode

DESCRIPTION

Pedit_multiply_matrix premultiplies the matrix *mat* with the matrix at position *label* in structure *name*. The matrix in the structure is replaced with the result of this concatenation. The command at *label* must be a **Pmultiply_matrix**, only the data portion of the command is replaced. If *postflag* is TRUE, then a postconcatenation is performed.
Both the structure *name* and the *label* must exist otherwise an error is generated.

NOTE

This routine cannot be used inside a structure or display list.
For example, if a structure was created with the following commands:

```
        Pcreate_struct(name);
...
        Pmake_label(label);
        Pmultiply_matrix(mat,postflag);
...
        Pclose_struct();
```

The **Pedit_multiply_matrix** call is equivalent to the following sequence of commands:

```
        Pget_structmatrix(name, label, omat);
        PEXtk_matrix_mult(mat, omat, rmat);
        Pedit_struct(name);
        Preplace_struct(label);
        Pmultiply_matrix(rmat,postflag);
        Pclose_struct();
```

Any changes made to the structure will not be reflected on the display until the structure is traversed again.

SEE ALSO

Pcreate_struct, Pmake_label, Pmultiply_matrix

NAME

Pedit_struct - Opens a structure for editing

SPECIFICATION

int Pedit_struct(int name)

ARGUMENTS

name	An integer representing the name of the structure to be edited

RETURNS

0 if successful, non-zero if in incorrect mode

DESCRIPTION

Pedit_struct opens structure *name* for editing. The structure *name* must have been created using the **Pcreate_struct** routine, otherwise an error is generated. Any geometry commands following this routine will be added to the end of the structure until a **Pclose_struct** is executed, or until the insertion pointer is changed with **Pinsert_struct** or **Preplace_struct**.

NOTE

This routine cannot be used inside a structure or display list.
Any changes made to the structure will not be reflected on the display until the structure is traversed again.

SEE ALSO

Pcreate_struct, Pinsert_struct, Preplace_struct

NAME

Pend_line - Ends use of vertices as polyline vertices

SPECIFICATION

int Pend_line(void)

ARGUMENTS

None

RETURNS

0 if successful, non-zero if an error occurred

DESCRIPTION

Pend_line terminates the polyline generation mode.

NOTE

The library maintains a buffer which is used to build the polyline, and when the **Pend_line** call is made, the data is sent to the server.

SEE ALSO

Pbegin_line

NAME

Pend_poly -Ends use of vertices as polygon vertices

SPECIFICATION

int Pend_poly(void)

ARGUMENTS

None

RETURNS

0 if successful, non-zero if an error occurred

DESCRIPTION

Pend_poly terminates the polygon generation mode.

NOTE

The library maintains a buffer which is used to build the polygon, and when the **Pend_poly** call is made, the data is sent to the server.

SEE ALSO

Pbegin_poly

NAME

Pend_tmesh -Ends use of vertices as triangle mesh vertices

SPECIFICATION

int Pend_tmesh(void)

ARGUMENTS

None

RETURNS

0 if successful, non-zero if an error occurred

DESCRIPTION

Pend_tmesh terminates the triangle mesh generation mode.

NOTE

The library maintains a buffer which is used to build the triangle mesh, and when the **Pend_tmesh** call is made, the data is sent to the server.

SEE ALSO

Pbegin_tmesh

NAME

Pexit - Exit PEXtk

SPECIFICATION

```
void Pexit(void)
```

ARGUMENTS

None

RETURNS

Nothing

DESCRIPTION

Pexit cleans up the graphics hardware in preparation for the application to exit. **Pinit** must be called before any more graphics can be drawn.

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pinit

NAME

Pflush - Flushes the graphics pipeline

SPECIFICATION

void Pflush(int flag)

ARGUMENTS

flag	Flag set to True to perform a full flush.

RETURNS

Nothing

DESCRIPTION

Pflush will flush the graphics pipeline and buffers. It will not perform a flush on any **Pbegin**... routine (i.e., it will not execute an associated **Pend_**... function). If *flag* is set to True, then a PEX End/Begin Rendering sequence will be performed to force a full flush of the PEX rendering pipeline. If *flag* is set to False, then the X and PEX buffers will be flushed.

NOTE

This routine would generally be used in single buffer mode to force the graphics primitives to be displayed.

SEE ALSO

Pswap_buffers

NAME

Pget_backbuffer - Returns the current backbuffer identifier

SPECIFICATION

unsigned long Pget_backbuffer(void)

ARGUMENTS

None

RETURNS

An unsigned long integer representing the backbuffer identifier

DESCRIPTION

Pget_backbuffer returns the backbuffer identifier for the current drawable. The interpretation of the identifier is hardware dependent. In most cases it will represent a drawable identifier (Window, Pixmap, or Multi-Buffer).

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pbuffer_mode, Pget_buffer_mode, Pset_drawable

NAME

Pget_backface - Returns current backfacing mode

SPECIFICATION

int Pget_backface(void)

ARGUMENTS

None

RETURNS

An integer representing the current backfacing mode

DESCRIPTION

Pget_backface returns the current backface culling mode. Possible returned values are PEXtk_CULL_BACKFACE (backface removal is on), PEXtk_CULL_FRONTFACE (front-face removal is on), and PEXtk_CULL_NONE (backfacing and frontfacing polygons are drawn).

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pbackface

NAME

Pget_border - Returns the geometry outlining attribute

SPECIFICATION

int Pget_border(void)

ARGUMENTS

None

RETURNS

An integer representing the current polygon border attribute

DESCRIPTION

Pborder returns the current polygon outline attribute.
Possible return values are:

PEXtk_BORDER_NONE
PEXtk_BORDER_SOLID
PEXtk_BORDER_DASHED
PEXtk_BORDER_DOTTED
PEXtk_BORDER_DOTDASH

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pborder, Ppoly, Ppoly_fill, Ppoly_shade

NAME

Pget_buffer_mode - Returns current buffering mode

SPECIFICATION

int Pget_buffer_mode(void)

ARGUMENTS

None

RETURNS

An integer representing the current buffering mode

DESCRIPTION

Pget_buffer_mode returns the current buffering mode.
Possible return values are PEXtk_DOUBLE_BUFFER or PEXtk_SINGLE_BUFFER.

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pbuffer_mode

NAME

Pget_clip_rectangle - Returns the current clipping rectangle

SPECIFICATION

void Pget_clip_rectangle(Screencoord *left, Screencoord *right, Screencoord *bottom, Screencoord *top)

ARGUMENTS

left	Pointer to the clip rectangle lower left x coordinate
right	Pointer to the clip rectangle upper right x coordinate
bottom	Pointer to the clip rectangle bottom left y coordinate
top	Pointer to the clip rectangle top right y coordinate

RETURNS

Nothing

DESCRIPTION

Pget_clip_rectangle returns the current clipping rectangle in *left*, *right*, *bottom*, *top* which are in X window relative coordinates. Note that 0,0 designates the bottom left hand corner of the rectangle.

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pclip_rectangle

NAME

Pget_color - Returns current polygon color

SPECIFICATION

```
void Pget_color(PCOLOR *col)
void Pget_light_color(PCOLOR *col)
void Pget_line_color(PCOLOR *col)
void Pget_marker_color(PCOLOR *col)
void Pget_specular_color(PCOLOR *col)
void Pget_text_color(PCOLOR *col)
```

ARGUMENTS

**col*	Pointer to a PCOLOR structure

RETURNS

Nothing

DESCRIPTION

Pget_color returns the current color set by **Pcolor**, **Pcolor_hsv**, **Pcolor_ind**, or **Pcolor_rgb**.
Pget_light_color returns the current color set by **Plight_color**, **Plight_color_hsv**, **Plight_color_ind**, or **Plight_color_rgb**.
Pget_line_color returns the current color set by **Pline_color**, **Pline_color_hsv**, **Pline_color_ind**, or **Pline_color_rgb**.
Pget_marker_color returns the current color set by **Pmarker_color**, **Pmarker_color_hsv**, **Pmarker_color_ind**, or **Pmarker_color_rgb**.
Pget_specular_color returns the current color set by **Pspecular_color**, **Pspecular_color_hsv**, **Pspecular_color_ind**, or **Pspecular_color_rgb**.
Pget_text_color returns the current color set by **Ptext_color**, **Ptext_color_hsv**, **Ptext_color_ind**, or **Ptext_color_rgb**.

NOTE

These routines cannot be used inside a structure or display list.

SEE ALSO

Pcolor, Plight_color, Pline_color, Pmarker_color, Pspecular_color, Ptext_color

NAME

Pget_depth_cue - Returns current depth cue mode

SPECIFICATION

int Pget_depth_cue(Fcoord *rmin, Fcoord *rmax)

ARGUMENTS

**rmin*	Pointer to a Fcoord specifying the minimum depth cue range
**rmax*	Pointer to a Fcoord specifying the maximum depth cue range

RETURNS

An integer representing the current depth cueing mode

DESCRIPTION

Pget_depth_cue returns the current depth cuing mode set by **Pdepth_cue**.
Possible return values are TRUE if depth cueing is enabled, or FALSE if depth cueing is disabled. If the return value is TRUE, then *rmin* and *rmax* contain the minimum and maximum depth used for depth cueing.

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pdepth_cue

NAME

Pget_display_buffer - Returns the current display buffer identifier

SPECIFICATION

unsigned long Pget_display_buffer(void)

ARGUMENTS

None

RETURNS

An unsigned long integer representing the display buffer identifier

DESCRIPTION

Pget_display_buffer returns the front buffer identifier (or drawable) if the current buffering mode is PEXtk_SINGLE_BUFFER, or the back buffer identifier if the mode is PEXtk_DOUBLE_BUFFER. The interpretation of the identifier is hardware dependent. In most cases it will represent a drawable identifier (Window, Pixmap, or Multi-Buffer).

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pbuffer_mode, Pget_backbuffer, Pget_buffer_mode, Pget_frontbuffer, Pset_drawable

NAME

Pget_frontbuffer - Returns the current front buffer identifier

SPECIFICATION

unsigned long Pget_frontbuffer(void)

ARGUMENTS

None

RETURNS

An unsigned long integer representing the display buffer identifier

DESCRIPTION

Pget_frontbuffer returns the front buffer identifier (or drawable). The interpretation of the identifier is hardware dependent. In most cases it will represent a drawable identifier (Window, Pixmap, or Multi-Buffer).

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pbuffer_mode, Pget_backbuffer, Pget_buffer_mode, Pget_display_buffer, Pset_drawable

NAME

Pget_identity_matrix - Returns 4x4 identity matrix

SPECIFICATION

void Pget_identity_matrix(Matrix mat)

ARGUMENTS

mat	Returns the 4x4 identity transformation matrix

RETURNS

Nothing

DESCRIPTION

Pget_identity_matrix copies the 4x4 identity matrix into *mat*.

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pget_matrix, Pget_inverse_matrix, Pload_matrix

NAME

Pget_inverse_matrix - Returns the inverse of the current transformation matrix

SPECIFICATION

void Pget_inverse_matrix(Matrix mat)

ARGUMENTS

mat	Returns the 4x4 inverse matrix

RETURNS

Nothing

DESCRIPTION

Pget_inverse_matrix copies the inverse of the current transformation matrix into *mat*.

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pget_matrix, Pload_matrix

NAME

Pget_line_type - Returns current line type

SPECIFICATION

int Pget_line_type(void)

ARGUMENTS

None

RETURNS

An integer representing the current line type

DESCRIPTION

Pget_line_type returns the current line type set by **Pline_type**. Possible line types are:

PEXtk_LINE_SOLID
PEXtk_LINE_DASHED
PEXtk_LINE_DOTTED
PEXtk_LINE_DOTDASH

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pline_type

NAME

Pget_line_width - Returns current line width

SPECIFICATION

```
Fdata Pget_line_width(void)
```

ARGUMENTS

None

RETURNS

An float representing the current line width scale factor

DESCRIPTION

Pget_line_width returns the current line width set by **Pline_width**.

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pline_width

NAME

Pget_marker_type - Returns current marker type

SPECIFICATION

int Pget_marker_type(void)

ARGUMENTS

None

RETURNS

An integer representing the current marker type

DESCRIPTION

Pget_line_type returns the current marker type set by **Pmarker_type**. Possible marker types are:

PEXtk_MRKR_POINT
PEXtk_MRKR_CIRCLE
PEXtk_MRKR_CROSS
PEXtk_MRKR_X
PEXtk_MRKR_ASTERISK
PEXtk_MRKR_BOX

NOTE

This routine cannot be used inside a structure or display list.
The marker type PEXtk_MRKR_BOX is only available on platforms supporting marker glyphs.

SEE ALSO

Pmarker_type

NAME

Pget_matrix - Returns the current transformation matrix

SPECIFICATION

void Pget_matrix(Matrix mat)

ARGUMENTS

mat	Returns the current 4x4 modeling transformation matrix

RETURNS

Nothing

DESCRIPTION

Pget_matrix copies the current transformation matrix into *mat*.

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pget_inverse_matrix, Pload_matrix

NAME

Pget_shade_mode - Returns current shade mode

SPECIFICATION

int Pget_shade_mode(void)

ARGUMENTS

None

RETURNS

An integer bitmask representing the current shade mode

DESCRIPTION

Pget_shade_mode returns the current shading mode.
Possible bitmask values are

PEXtk_SHADE_NONE	No attributes are defined
PEXtk_SHADE_WIRE	Edges only, polygon is not filled
PEXtk_SHADE_FLAT	no color interpolation
PEXtk_SHADE_COLORS	color interpolation
PEXtk_SHADE_DOTPRODUCT	Dot product interpolation
PEXtk_SHADE_NORMALS	Normal interpolation
PEXtk_SHADE_SPECULAR	Specular highlights
PEXtk_SHADE_HIDDENLINE	Hiddenline mode
PEXtk_SHADE_ANTIALIAS	Line antialiasing mode
PEXtk_SHADE_LIGHTS	Lighting mode

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pshade_mode

NAME

Pget_structmatrix - Returns a matrix in a structure

SPECIFICATION

int Pget_structmatrix(int name, int label, Matrix mat)

ARGUMENTS

name	The structure name the matrix is located in
label	An integer representing the label at which the matrix is located
mat	The returned 4x4 structure matrix

RETURNS

0 if successful, 1 if error

DESCRIPTION

Pget_structmatrix copies the matrix in the structure *name* at position *label* into *mat*.
The command at *label* must be a **Pmultiply_matrix** or a **Pload_matrix**. Note that only the data portion of the command is returned.
Both the structure *name* and the *label* must exist otherwise an error is generated.

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pedit_multiply_matrix, Pmultiply_matrix

NAME

Pget_structure - Returns the name of the currently open structure

SPECIFICATION

int Pget_structure(void)

ARGUMENTS

None

RETURNS

An integer representing the name of the currently open structure

DESCRIPTION

Pget_structure returns the name of the currently open structure. If no structure is open, then PEXtk_NO_STRUCT_OPEN is returned.

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pcreate_struct

NAME

Pget_support_info - Returns PEXtk support data

SPECIFICATION

int Pget_support_info(int type, int **data)

ARGUMENTS

type	An integer representing the type of information to return
***data*	Returns a pointer to an array of data containing the support data

RETURNS

0 if successful, 1 if error

DESCRIPTION

Pget_support_info returns information on PEXtk dependent constants and PEX server dependent constants. The memory for the data is allocated in **Pget_support_info**, and must be freed by the application using **free**.
Possible values for *type* are:

PEXtk_INFO_MAX_LIGHTS	Maximum number of enabled lights allowed
PEXtk_INFO_MAX_VIEWS	Maximum number of views allowed
PEXtk_INFO_ZBUFFER	Integer value is true if zbuffer is supported
PEXtk_INFO_ANTIALIAS	Integer value is true if antialiasing is supported
PEXtk_INFO_DITHER	Integer value is true if dithering is supported
PEXtk_INFO_TRANPARENCY	Integer value is true if transparency is supported
PEXtk_INFO_MAX_NAMES	Maximum number of names for name stack
PEXtk_INFO_LIGHT_TYPES	Returns the light types that are supported. The first value returned is the number of items in the list, followed by the light types supported.
PEXtk_INFO_BORDER_TYPES	Returns the border types that are supported. The first value returned is the number of items in the list, followed by the border types supported.
PEXtk_INFO_LINE_TYPES	Returns the line types that are supported. The first value returned is the number of items in the list, followed by the line types supported.
PEXtk_INFO_SHADE_MODES	Returns the shade mode that are supported. The first value returned is the number of items in the list, followed by the shade modes supported.

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Plight, Pshade_mode, Pzbuffer

NAME

Pget_view - Returns the current window viewing values

SPECIFICATION

```
int Pget_view(int *type, Fcoord *left, Fcoord *right, Fcoord *bottom, Fcoord *top,
          Fcoord *fov, Fcoord *aspect, Fcoord *near, Fcoord *far)
```

ARGUMENTS

**type*	Pointer to value to receive the viewing type
**left*	Pointer to value to receive the viewing lower left x coordinate
**right*	Pointer to value to receive the viewing upper right x coordinate
**bottom*	Pointer to value to receive the viewing bottom left y coordinate
**top*	Pointer to value to receive the viewing top right y coordinate
**fov*	Pointer to value to receive the near clipping plane value
**aspect*	Pointer to value to receive the far clipping plane value
**near*	Pointer to value to receive the near clipping plane value
**far*	Pointer to value to receive the far clipping plane value

RETURNS

The current view index id

DESCRIPTION

Pget_view returns the current window viewing values. The current viewing type is returned in *type*. If type is PEXtk_VIEW_ORTHO, then the orthographic values are returned in *left*, *right*, *bottom*, *top*, which correspond to the values passed to **Portho_view**. If type is PEXtk_VIEW_PERSP, the field of view is returned in *fov*, the aspect ratio is returned in *aspect*, and the near and far clipping plane values are returned in *near* and *far*, respectively. If the type is PEXtk_VIEW_WINDOW, then the values are returned in *left*, *right*, *bottom*, *top*, *near*, *far*.

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Portho_view, Portho_2d_view, Ppersp_view, Pwindow_view

NAME

Pget_viewport - Returns the current viewport

SPECIFICATION

int Pget_viewport(Screencoord *left, Screencoord *right, Screencoord *bottom,
    Screencoord *top)

ARGUMENTS

left	Pointer to the left screen coordinate
right	Pointer to the right screen coordinate
bottom	Pointer to the bottom screen coordinate
top	Pointer to the top screen coordinate

RETURNS

The current viewport stack location

DESCRIPTION

Pget_viewport loads the given parameters with the values on top of the viewport stack. Note that 0,0 represents the bottom left hand corner of the viewport.

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Ppop_viewport, Ppush_viewport, Pviewport

NAME

Pget_zbuffer - Returns current zbuffer mode

SPECIFICATION

int Pget_zbuffer(void)

ARGUMENTS

None

RETURNS

An integer representing whether the zbuffer is on or off

DESCRIPTION

Pget_zbuffer returns the current zbuffer mode.
Possible return values are PEXtk_ZBUFFER_ON (zbuffering is on), or PEXtk_ZBUFFER_OFF (zbuffering is off).

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pzbuffer

NAME

Pinit - Initialize PEXtk

SPECIFICATION

int Pinit(Display *display)

ARGUMENTS

display	Pointer to the display structure returned by XOpenDisplay

RETURNS

0 if successful, non-zero if error

DESCRIPTION

Pinit initializes the PEXtk environment and resets the hardware in preparation for use. This call must be made before any other PEXtk call. All PEXtk commands are sent to the server via the connection specified by *display*.

NOTE

This routine cannot be used inside a structure or display list, and should only be called once at the start of the application.

SEE ALSO

Pexit, Pset_drawable

NAME

Pinsert_struct - Positions the edit position to a label within a structure

SPECIFICATION

int Pinsert_struct(int label)

ARGUMENTS

label	An integer representing the structure label

RETURNS

0 is successful, non-zero if an error occurs

DESCRIPTION

Pinsert_struct inserts the commands after position *label* in the structure that has been opened for editing by **Pedit_struct**. An error will be generated if the structure is not open for editing or the *label* does not exist.

NOTE

Any changes made to the structure will not be reflected on the display until the structure is traversed again.

SEE ALSO

Pclose_struct, Pedit_struct, Pmake_label, Preplace_struct

NAME

Plabel_used - Indicates whether a label is in use.

SPECIFICATION

Boolean Plabel_used(int name, int label)

ARGUMENTS

name	An integer indicating which structure name to search
label	An integer representing the label to search for

RETURNS

0 if label is not in use, 1 if the label is in use

DESCRIPTION

Plabel_used returns TRUE (1) if *label* is in use in the structure *name*.
It returns FALSE (0) if either the structure *name* is not in use or *label* is not being used within that structure.

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pstruct_used, Pmake_label

NAME

Plight - Load a light source

SPECIFICATION

void Plight(int n, PEXtk_LIGHT *lgt)

ARGUMENTS

n	An integer representing the light number
**lgt*	A pointer to the structure containing the light type, direction, position, attenuation factors, concentration, and angle of influence.

RETURNS

Nothing

DESCRIPTION

Plight sets the n^{th} light to the light type specified by *lgt->type*, the direction specified by *lgt->dir.x*, *lgt->dir.*y, *lgt->dir.z*, and the position specified by *lgt->pos.x*, *lgt->pos.y*, *lgt->pos.z*. If **lgt* is NULL, then the n^{th} light is turned off. The light number *n* must be between 1 and the maximum number of lights supported.
The light type may be one of:

PEXtk_LIGHT_POINT
PEXtk_LIGHT_INFINITE
PEXtk_LIGHT_SPOT

A PEXtk_LIGHT_INFINITE light uses only the *lgt->dir* values. A PEXtk_LIGHT_POINT light uses the *lgt->pos*, *lgt->dir, lgt->atten*1, and *lgt->atten2* values. A PEXtk_LIGHT_SPOT light uses the *lgt->pos*, *lgt->dir*, *lgt->atten*1, *lgt->atten2*, *lgt->conc,* and *lgt->angle* values. In order for a light source to take effect, the **Pshade_mode** must be set to include PEXtk_SHADE_LIGHTS. The light source takes on the current light color attribute set by **Plight_color**.
If a light number of 0 is specified, then all lights are deactivated.
The *atten1* (L_{a1}) and *atten2* (L_{a2}) values are used to calculate the light attenuation factor L_{att}

$$L_{att} = \frac{1}{L_{a1} + L_{a2} (\| O_{pos} - L_{pos} \|)}$$

The default value for L_{a1} is.50, and for L_{a2} the default value is 0.0. If both *lgt->atten1* and *lgt->atten2* are zero, then the default values are used.

NOTE

Different hardware will support different list types. If a light type is requested that is not supported on the platform, it is ignored. This routine cannot be used inside a structure.

SEE ALSO

Pget_support_info, Plight_ambient, Plight_color, Pshade_mode

NAME

Plight_ambient - Set ambient light intensity and color

SPECIFICATION

void Plight_ambient(Fdata amb)

ARGUMENTS

amb	A floating point value representing the ambient light value

RETURNS

Nothing

DESCRIPTION

Plight_ambient sets the ambient light intensity to *amb*, which is between 0.0 and 1.0. The light color is set to the current light color set with **Plight_color**. An intensity value of 0.0 turns the ambient light off. Note that only one ambient light may be used.

NOTE

This routine cannot be used inside a structure.

SEE ALSO

Plight, Plight_color

NAME

Plight_color - Set light color
Plight_color_ind - Set light color to index
Plight_color_hsv - Set light color to hsv
Plight_color_rgb - Set light color to rgb

SPECIFICATION

void Plight_color(PCOLOR *col)
void Plight_color_ind(int colind)
void Plight_color_hsv(Fdata hue, Fdata saturation, Fdata value)
void Plight_color_rgb(Fdata red, Fdata green, Fdata blue)

ARGUMENTS

**col*	Pointer to a PCOLOR structure containing the RGB, HSV, or CIE values
colind	An integer representing the color index
hue	A float representing the color hue
saturation	A float representing the color saturation
value	A float representing the color value
red,green,blue	Floats representing the RGB colors

RETURNS

Nothing

DESCRIPTION

Plight_color sets the light source color to *co*l.
Plight_color_ind sets the light source color to the index *ind*.
Plight_color_hsv sets the light source color to the HSV color specified by *hue*, *saturation*, *value* where *hue* is between 0.0 and 360.0, *saturation* is between 0.0 and 1.0 and *value* is between 0.0 and 1.0.
Plight_color_rgb sets the light source color to the RGB color specified by *red*, *green*, *blue* with values ranging between 0.0 and 1.0.

NOTE

The color table must be set with **Pset_color_table** before the indices are referenced by the **Plight_color_ind** routine.
The current light color is an attribute and is saved by **Ppush_attributes** and restored with **Ppop_attributes**.

SEE ALSO

Plight, Plight_ambient, Pshade_mode

NAME

Pline_color - Set current line color
Pline_color_ind - Set current line color to index
Pline_color_hsv - Set current line color to HSV
Pline_color_rgb - Set current line color to RGB

SPECIFICATION

```
void Pline_color(PCOLOR *col)
void Pline_color_ind(int colind)
void Pline_color_hsv(Fdata hue, Fdata saturation, Fdata value)
void Pline_color_rgb(Fdata red, Fdata green, Fdata blue)
```

ARGUMENTS

**col*	Pointer to a PCOLOR structure containing the RGB, HSV, or CIE values
colind	An integer representing the color index
hue	A float representing the color hue
saturation	A float representing the color saturation
value	A float representing the color value
red,green,blue	Floats representing the RGB colors

RETURNS

Nothing

DESCRIPTION

Pline_color sets the line color to *col* for all lines and polylines that follow.
Pline_color_ind sets the line color to the index *colind*.
Pline_color_hsv sets the line color specified by *hue*, *saturation* and *value* where *hue* is between 0.0 and 360.0, *saturation* is between 0.0 and 1.0 and *value* is between 0.0 and 1.0.
Pline_color_rgb sets the line color specified by *red*, *green*, *blue* with values ranging between 0.0 and 1.0.

NOTE

The color table must be set with **Pset_color_table** before the index is referenced by the **Pline_color_ind** routine.
The current line color is an attribute and is saved by **Ppush_attributes** and restored with **Ppop_attributes**.

SEE ALSO

Pcolor, Pset_color_table

NAME

Pline_type - Sets the line type to be used for polylines

SPECIFICATION

void Pline_type(int type)

ARGUMENTS

type	The line type to be used when drawing line and curve primitives

RETURNS

Nothing

DESCRIPTION

Pline_type sets the type of all lines and polylines that follow. Possible line types are:

PEXtk_LINE_SOLID
PEXtk_LINE_DASHED
PEXtk_LINE_DOTTED
PEXtk_LINE_DOTDASH

The default line type is PEXtk_LINE_SOLID.
The current line type is an attribute and is saved by **Ppush_attributes** and restored with **Ppop_attributes**.

SEE ALSO

Pbegin_line, Pend_line, Pget_line_type, Ppoly_line

NAME

Pline_width - Sets the width of lines

SPECIFICATION

void Pline_width(Fdata width)

ARGUMENTS

width	The line width scale factor to be used when drawing line primitives

RETURNS

Nothing

DESCRIPTION

Pline_width sets the width of all lines and polylines that follow to *width*.
The current line width is an attribute and is saved by **Ppush_attributes** and restored with **Ppop_attributes**.

SEE ALSO

Pget_line_width, Ppoly_line

NAME

Pload_attribute - loads an attribute

SPECIFICATION

void Pload_attribute(int attribute, void *valueptr)

ARGUMENTS

attribute	An integer representing the attribute type to load
**valueptr*	A pointer to the address stack

RETURNS

Nothing

DESCRIPTION

Pload_attribute changes the attribute of *attribute* to the location at address *valueptr*. This replaces the current value of that attribute on top of the attribute stack.
The previous value may be saved with **Ppush_attributes**.

NOTE

All attributes are set by their own call. This routine would be used for extensions to PEXtk.

SEE ALSO

Ppush_attributes, Ppop_attributes

NAME

Pload_identity - Load identity matrix on top of stack

SPECIFICATION

void Pload_identity(void)

ARGUMENTS

None

RETURNS

Nothing

DESCRIPTION

Pload_identity replaces the top of the matrix stack with the identity matrix.
The matrix should be pushed first with **Ppush_matrix** if the current matrix needs to be preserved.

NOTE

This is equivalent to calling **Pload_matrix**(*unitymat*) where *unitymat* contains the identity matrix.

SEE ALSO

Pload_matrix, Ppop_matrix, Ppush_matrix

NAME

Pload_matrix - Replace the current transformation matrix

SPECIFICATION

void Pload_matrix(Matrix mat)

ARGUMENTS

mat	The 4x4 transformation matrix to load

RETURNS

Nothing

DESCRIPTION

Pload_matrix replaces the top of the matrix stack with *mat*.
The previous transformation matrix may be saved with **Ppush_matrix** and restored with **Ppop_matrix**.

NOTE

Two matrix stacks are maintained, the modeling and viewing matrix. The projection matrix uses the same stack pointer as the viewing matrix.
This routine cannot be used inside a structure for matrix mode PEXtk_MATRIX_PROJ.

SEE ALSO

Pget_matrix, Pget_inverse_matrix, Pmultiply_matrix, Ppop_matrix, Ppush_matrix.

NAME

Plogical_function - Sets the bit function for display updates.

SPECIFICATION

void Plogical_function(unsigned long func)

ARGUMENTS

func	The logical function to be performed when updating the display.

RETURNS

Nothing

DESCRIPTION

Plogical_function sets *func* as the logical function (or operation) to be used when updating the display drawable. This function controls how the source pixel values are to be combined with the destination pixel values already on the display drawable.

NOTE

Support for logical function is platform dependent.

SEE ALSO

Pplane_mask, Pset_display_buffer, Pset_drawable, Pset_display_buffer

NAME

Plookat_view - Define a view point and a reference point

SPECIFICATION

void Plookat_view(Fcoord vx, Fcoord vy, Fcoord vz, Fcoord px, Fcoord py, Fcoord pz, Angle twist)

ARGUMENTS

vx	The x viewpoint coordinate
vy	The y viewpoint coordinate
vz	The z viewpoint coordinate
px	The x reference coordinate
py	The y reference coordinate
pz	The z reference coordinate
twist	The twist angle

RETURNS

Nothing

DESCRIPTION

Plookat_view defines a viewing orientation transformation matrix. This routine is usually used in conjunction with either the **Portho_view**, **Ppersp_view**, or **Pwindow_view** routine.The transform is generated from the supplied data, where *vx*, *vy*, and *vz* defines the viewpoint or *from* location, and *px*, *py*, and *pz* defines the reference point or *to* location. The *from* and *to* points defines a line of sight, *twist* is the rotation around the z axis of that line of sight.
The resulting transformation matrix replaces the top of the matrix stack.

NOTE

This routine cannot be used inside a structure.

SEE ALSO

Portho_view, Ppersp_view, Ppolar_view, Pwindow_view

NAME

Pmake_label - labels the current position in the structure

SPECIFICATION

int Pmake_label(int label)

ARGUMENTS

label	An integer representing the structure label

RETURNS

0 if successful, non-zero if an error occurs

RETURNS

Nothing

DESCRIPTION

Pmake_label marks the current location in a structure with *label*. A structure must be currently open using **Pcreate_struct** or **Pedit_struct**.
This label may then be used by the various structure editing commands, and will reference the command inserted after the label.
The same label may be used in different structures.
An error is generated if a structure is not currently open, or the label is already being used in this structure.

NOTE

The name *label* may be any name other than -1.

SEE ALSO

Pedit_struct, Plabel_used, Pedit_multiply_matrix, Pget_structmatrix, Pcreate_struct, Pinsert_struct, Preplace_struct

NAME

Pmarker - Draw a set of polymarkers

SPECIFICATION

void Pmarker(int n, Fcoord3D points[])

ARGUMENTS

n	The number of vertices in the *points* array
points[]	Array of 3 Fcoords that define the coordinates of each polymarker

RETURNS

Nothing

DESCRIPTION

Pmarker will draw a set of marker primitives. This routine will draw *n* markers at the locations defined in the *points* array. The color used is the current marker color set with **Pmarker_color**, the type used is the current marker type set with **Pmarker_type**, and the size of the marker is scaled using the scale factor set with **Pmarker_scale**.

SEE ALSO

Pmarker_color, Pmarker_type, Pmarker_scale

NAME

Pmarker_color - Set current marker color
Pmarker_color_ind - Set current marker color to index
Pmarker_color_hsv - Set current marker color to HSV
Pmarker_color_rgb - Set current marker color to RGB

SPECIFICATION

```
void Pmarker_color(PCOLOR *col)
void Pmarker_color_ind(int colind)
void Pmarker_color_hsv(Fdata hue, Fdata saturation, Fdata value)
void Pmarker_color_rgb(Fdata red, Fdata green, Fdata blue)
```

ARGUMENTS

**col*	Pointer to a PCOLOR structure containing the RGB, HSV or CIE values
colind	An integer representing the color index
hue	A float representing the color hue
saturation	A float representing the color saturation
value	A float representing the color value
red,green,blue	Floats representing the RGB colors

RETURNS

Nothing

DESCRIPTION

Pmarker_color sets the marker color to *col* for all **Pmarker** calls that follow.
Pmarker_color_ind sets the marker color to the index *colind*.
Pmarker_color_hsv sets the marker color specified by *hue*, *saturation* and *value* where *hue* is between 0.0 and 360.0, *saturation* is between 0.0 and 1.0 and *value* is between 0.0 and 1.0.
Pmarker_color_rgb sets the marker color specified by *red*, *green*, *blue* with values ranging between 0.0 and 1.0.

NOTE

The color table must be set with **Pset_color_table** before the index is referenced by the **Pmarker_color_ind** routine.
The current marker color is an attribute and is saved by **Ppush_attributes** and restored with **Ppop_attributes**.

SEE ALSO

Pmarker, Pmarker_scale, Pmarker_type, Pset_color_table

NAME

Pmarker_scale - Sets the scale to use when drawing markers

SPECIFICATION

```
void Pmarker_scale(Fdata scale)
```

ARGUMENTS

scale	The marker scale to be used when drawing marker primitives

RETURNS

Nothing

DESCRIPTION

Pmarker_scale sets the scale for all markers that follow. The nominal marker size is scaled by the value given by *scale* to achieve the final marker size. The default marker scale is 1.0. The current marker scale is an attribute and is saved by **Ppush_attributes** and restored with **Ppop_attributes**.

NOTE

The current marker scale is an attribute and is saved by **Ppush_attributes** and restored with **Ppop_attributes**.
For the marker type PEXtk_MRKR_POINT, the scale factor is ignored.

SEE ALSO

Pmarker, Pmarker_color, Pmarker_type

NAME

Pmarker_type - Sets the type to use for markers

SPECIFICATION

void Pmarker_type(int type)

ARGUMENTS

type	The marker type to be used when drawing marker primitives

RETURNS

Nothing

DESCRIPTION

Pmarker_type sets the type of all markers that follow. The possible marker types are:

PEXtk_MRKR_POINT
PEXtk_MRKR_CIRCLE
PEXtk_MRKR_CROSS
PEXtk_MRKR_X
PEXtk_MRKR_ASTERISK
PEXtk_MRKR_BOX

The current marker type is an attribute and is saved by **Ppush_attributes** and restored with **Ppop_attributes**.

NOTE

The current marker type is an attribute and is saved by **Ppush_attributes** and restored with **Ppop_attributes**.
The marker type PEXtk_MRKR_BOX is only available on platforms that support user settable marker glyphs.

SEE ALSO

Pmarker, Pmarker_color, Pmarker_scale

NAME

Pmatrix_mode - Sets the matrix mode

SPECIFICATION

void Pmatrix_mode(int mode)

ARGUMENTS

mode	The matrix mode to be used when updating the matrix stack

RETURNS

Nothing

DESCRIPTION

Pmatrix_mode sets the mode for updating the matrix stack. The possible matrix modes are:

PEXtk_MATRIX_MODELING
PEXtk_MATRIX_VIEWING
PEXtk_MATRIX_PROJ

The default mode is PEXtk_MATRIX_MODELING.
The matrix mode PEXtk_MATRIX_PROJ can only be used for **Pget_matrix**, **Pload_identity**, and **Pload_matrix**.

NOTE

Two matrix stacks are maintained, the modeling and viewing matrix. The projection matrix uses the same stack pointer as the viewing matrix.

SEE ALSO

Pget_matrix, Pload_matrix

NAME

Pmove_2d - Move to a 2D point

SPECIFICATION

void Pmove_2d(Fcoord x, Fcoord y)

ARGUMENTS

x	The x coordinate value
y	The y coordinate value

RETURNS

Nothing

DESCRIPTION

Pmove_2d moves the current graphics position to *x*, *y*. No primitives are actually drawn. It is used primarily for drawing line segments in conjunction with **Pdraw_2d**.

SEE ALSO

Pdraw_2d, Ppoint_2d.

NAME

Pmove_3d - Move to a 3D point

SPECIFICATION

void Pmove_3d(Fcoord x, Fcoord y, Fcoord z)

ARGUMENTS

x	The x coordinate value
y	The y coordinate value
z	The z coordinate value

RETURNS

Nothing

DESCRIPTION

Pmove_3d moves the current graphics position to *x*, *y*, *z*. No primitives are actually drawn. It is used primarily for drawing line segments in conjunction with **Pdraw_3d**.

SEE ALSO

Pdraw_3d, Ppoint_3d.

NAME

Pmultiply_matrix - Multiply the current transformation matrix

SPECIFICATION

void Pmultiply_matrix(Matrix mat, int postflag)

ARGUMENTS

mat	The 4x4 transformation matrix
postflag	Flag set to TRUE to perform a postconcatenation

RETURNS

Nothing

DESCRIPTION

Pmultiply_matrix replaces the top of the matrix stack with the concatenation of *mat* and the current transformation matrix.
If *postflag* is TRUE, then a postconcatenation is performed such that $T_S = T_{new} \cdot T_S$, where T_S is the current matrix on top of the stack, T_{new} is the input matrix *mat*. If *postflag* is not TRUE, then the matrix mat is preconcatenated to the matrix stack, where: $T_S = T_S \cdot T_{new}$.
Use **Ppush_matrix** to save the previous matrix.

NOTE

Two matrix stacks are maintained, the modeling and viewing matrix. The projection matrix uses the same stack pointer as the viewing matrix.

SEE ALSO

Pedit_multiply_matrix, Pload_matrix, Ppop_matrix, Ppush_matrix

NAME

Pnurb_curve- Draw a rational or non-rational B-spline curve

SPECIFICATION

void Pnurb_curve(int type, int order, int numKnots, float *knots, int numPoints,
 Fcoord *points)

ARGUMENTS

type	The type of B-spline curve, either rational or non-rational.
order	The order of the polynomial expression.
numKnots	The number of knot values in the *knots* array
**knots*	Array of floats representing the knot values for the curve
numPoints	The number of control points
**points*	Array of floats that define the control point data for the curve

RETURNS

Nothing

DESCRIPTION

Pnurb_curve generates a Non Uniform Rational B-spline curve primitive. For this primitive. the vertex data *points* specifies the control points of the curve. If *type* is set to PEXtk_RATIONAL, then a rational B-spline (NURB) curve is drawn, and *points* must be in homogeneous coordinates (4D space with 4 floats per control point). If *type* is set to PEXtk_NONRATIONAL, then a non-rational B-spline curve is drawn, and *points* must be specified in non-homogenous coordinates (3D space with 3 floats per control point).The surface is evaluated over the parameter range *umin* and *umax* set by **Pcurve_parm_range**. The value *umin* must be greater than or equal to the *uKnot*[*uOrder*-1] knot value. The *umax* value must be greater than or equal to the *uKnot*[*numUKnots*-*uOrder*] value.

The curve is drawn with the current line color.

SEE ALSO

Pcurve_approx_method, Pcurve_parm_range, Pline_color, Pnurb_surface

NAME

Pnurb_surface- Draw a rational or non-rational B-spline surface

SPECIFICATION

```
void Pnurb_surface(int type, int uOrder, int vOrder, int numUKnots, float *uKnots,
        int numVKnots, float *vKnots, int uPts, int vPts, Fcoord *points,
        PEXtkTrimLoop *trimLoops, int numLoops)
```

ARGUMENTS

type	The type of B-spline curve, either rational or non-rational.
uOrder	The order of the polynomial expression in the u direction.
vOrder	The order of the polynomial expression in the v direction.
numUKnots	The number of knot values in the *uKnots* array
**uKnots*	Array of floats representing the knot values in the u direction.
numVKnots	The number of knot values in the *vKnots* array
**vKnots*	Array of floats representing the knot values in the v direction.
uPts	The number of control points in the u direction
vPts	The number of control points in the v direction
**points*	Array of floats that define the control point data for the surface
**trimLoops*	Pointer to a structure containing trimming data for the surface
numLoops	The number of trimming loops in the *trimLoops* structure

RETURNS

Nothing

DESCRIPTION

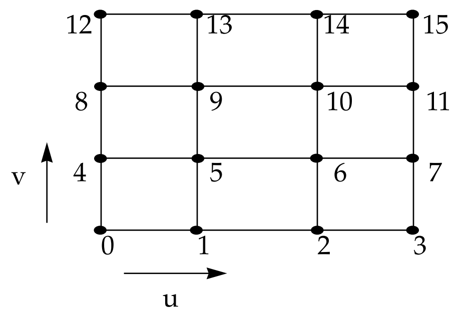
Pnurb_surface generates a Non Uniform Rational B-spline surface primitive. For this primitive. the vertex data *points* specifies the control points of the surface. If *type* is set to PEXtk_RATIONAL, then a rational B-spline (NURB) surface is drawn, and *points* must be in homogeneous coordinates (4D space with 4 floats per control point). If *type* is set to PEXtk_NONRATIONAL, then a non-rational B-spline surface is drawn, and *points* must be specified in non-homogenous coordinates (3D space with 3 floats per control point).

The uOrder and vOrder values must be positive integers, and they represent the order of the polynomial expression (order = degree + 1). The *numUKnots* value represents the number of values in the *uKnots* array, and *numVKnots* represents the number of values in the *vKnots* array. The *uKnots* and *vKnots* array contain the list of knot values in the parametric space. The values in the knot arrays must be a non-decreasing sequence of numbers.

The surface is evaluated over the parameter range *umin*, *umax*, *vmin*, and *vmax* set by **Psurface_parm_range**. The value *umin* must be greater than or equal to the *uKnot*[*uOrder*-1] knot value, and *vmin* must be greater than or equal to the *vKnot*[*vOrder*-1] knot value. The *umax* value must be greater than or equal to the *uKnot*[*numUKnots*-*uOrder*] value, and the *vmax* value must be greater than or equal to the *vKnot*[*numVKnots*-*vOrder*] value.

The surface is drawn with the current polygon color.

The control points are ordered in the points array as [v,u], where the *u* parameter varies the fastest. For example, a bicubic B-spline patch would be arranged as follows:



SEE ALSO

Psurface_approx_method, Psurface_parm_range, Pcolor, Pnurb_curve

NAME

Portho_2d_view - Define a 2D orthographic viewing projection

SPECIFICATION

void Portho_2d_view(int id, Fcoord left, Fcoord right, Fcoord top, Fcoord bottom)

ARGUMENTS

id	The viewing id number
left	The left screen coordinate
right	The right screen coordinate
bottom	The bottom screen coordinate
top	The top screen coordinate

RETURNS

Nothing

DESCRIPTION

Portho_2d_view sets up an orthographic viewing projection where the world coordinates fall within the rectangle defined by *left*, *right*, *top* and *bottom*.
The clipping planes are also set to this rectangle.
The view *id* is a reference number. If it is set to zero, then the viewing parameters are saved in the viewing table at the current viewing stack location for later reference with **Pset_view_index**. If *id* is be set to a value from 1 to PEXtk_VIEW_SIZE, then the viewing parameters are saved in the viewing table at location *id*, and *id* may then be used to refer to this view with **Pset_view_index**. This is the recommended alternative technique to saving and restoring views with **Ppush_matrix** and **Ppop_matrix**.

NOTE

This routine cannot be used inside a structure.
The only method to specify the viewing parameters for a view *id* of zero is to use the **Pset_view_index** routine in conjunction with this routine.
The resulting matrix is loaded into the view mapping matrix in the viewing table. A unity matrix will be loaded into the view orientation matrix by default.

SEE ALSO

Portho_view, Ppersp_view, Pwindow_view

NAME

Portho_view - Define a 3D orthographic viewing projection

SPECIFICATION

```
void Portho_view(int id, Fcoord left, Fcoord right, Fcoord bottom, Fcoord top,
        Fcoord near, Fcoord far)
```

ARGUMENTS

id	The view id number
left	The left screen coordinate
right	The right screen coordinate
bottom	The bottom screen coordinate
top	The top screen coordinate
near	The near clipping plane
far	The far clipping plane

RETURNS

Nothing

DESCRIPTION

Portho_view sets up an orthographic viewing projection view where the world coordinates fall within the parallelepiped defined by *left*, *right*, *top, bottom, near* and *far.*
The clipping planes are also set to this parallelepiped.
The view *id* is a reference number. If it is set to zero, then the viewing parameters are saved in the viewing table at the current viewing stack location for later reference with **Pset_view_index**. If *id* is set to a value from 1 to PEXtk_VIEW_SIZE, then the viewing parameters are saved in the viewing table at location *id*, and *id* may then be used to refer to this view with **Pset_view_index**.This is the recommended alternative technique to saving and restoring views with **Ppush_matrix** and **Ppop_matrix**.

NOTE

This routine cannot be used inside a structure.
The only method to specify the viewing parameters for a view *id* of zero is to use the **Pset_view_index** routine in conjunction with this routine.
The resulting matrix is loaded into the view mapping matrix in the viewing table. A unity matrix will be loaded into the view orientation matrix by default.

SEE ALSO

Portho_2d_view, Ppersp_view, Pwindow_view

NAME

Ppersp_view - Define a 3D perspective viewing pyramid

SPECIFICATION

```
void Ppersp_view(int id, Angle fovy, Fcoord aspect, Fcoord near, Fcoord far)
```

ARGUMENTS

id	The view id number
fovy	The field of view in the y direction
aspect	The right screen coordinate
near	The near clipping plane
far	The far clipping plane

RETURNS

Nothing

DESCRIPTION

Ppersp_view defines a viewing frustum from the supplied parameters. The *fovy* is the field of view in the y direction, *aspect* sets the aspect ratio x/y, and *near* and *far* represent the near and far clipping planes. The clipping planes are set to clip to a frustum that matches the field of view.

The view *id* is a reference number. If it is set to zero, then the viewing parameters are saved in the viewing table at the current viewing stack location for later reference with **Pset_view_index**. If *id* is set to a value from 1 to PEXtk_VIEW_SIZE, then the viewing parameters are saved in the viewing table at location *id*, and *id* may then be used to refer to this view with **Pset_view_index**. This is the recommended alternative technique to saving and restoring views with **Ppush_matrix** and **Ppop_matrix**.

NOTE

This routine cannot be used inside a structure.

The only method to specify the viewing parameters for a view *id* of zero is to use the **Pset_view_index** routine in conjunction with this routine.

The resulting matrix is loaded into the view mapping matrix in the viewing table. A unity matrix will be loaded into the view orientation matrix by default.

SEE ALSO

Portho_2d_view, Portho_view, Pwindow_view

NAME

Ppick_id - Sets the element pick id

SPECIFICATION

void Ppick_id(int id)

ARGUMENTS

id	An integer representing the element pick id

RETURNS

Nothing

DESCRIPTION

Ppick_id sets the element pick id to *id*. All elements that follow will have *id* as its pick id.

SEE ALSO

Ppoly, Ppoly_fill, Ppoly_index, Ppoly_line, Ppoly_fill_area, Ppoly_shade, Ppoly_shade_texture, Ppoly_texture

NAME

Pplane_mask - Sets which bitplanes are to be updated

SPECIFICATION

void Pplane_mask(unsigned long mask)

ARGUMENTS

mask	The bitmask representing which bitplanes are to be updated.

RETURNS

Nothing

DESCRIPTION

Pplane_mask sets *mask* as the bitmask to be used when updating the display drawable. The bitmask affects the planes in the current display drawable set by **Pset_display_buffer**, or the front buffer when in single/double buffer mode. Each bit in *mask* represents a bitplane of the display.

NOTE

Support for plane mask is platform dependent.

SEE ALSO

Plogical_function, Pset_display_buffer, Pset_drawable, Pset_display_buffer

NAME

Ppoint_2d - Draw a 2D point

SPECIFICATION

void Ppoint_2d(Fcoord x, Fcoord y)

ARGUMENTS

x	The x coordinate value
y	The y coordinate value

RETURNS

Nothing

DESCRIPTION

Ppoint_2d draws a point at the world coordinates *x*, *y*. The point is drawn in the current marker color. The current graphics position is set to *x*, *y*.

NOTE

This routine uses Pmarker with a marker type of PEXtk_MRKR_POINT.

SEE ALSO

Pcolor, Pdraw_2d, Pmarker, Pmove_2d

NAME

Ppoint_3d - Draw a 3D point

SPECIFICATION

void Ppoint_3d(Fcoord x, Fcoord y, Fcoord z)

ARGUMENTS

x	The x coordinate value
y	The y coordinate value
z	The z coordinate value

RETURNS

Nothing

DESCRIPTION

Ppoint_3d draws a point at the world coordinates *x*, *y*, *z*. The point is drawn in the current marker color. The current graphics position is set to *x*, *y*, *z*.

NOTE

This routine uses Pmarker with a marker type of PEXtk_MRKR_POINT.

SEE ALSO

Pcolor, Pdraw_3d, Pmarker, Pmove_3d

NAME

Ppolar_view - Define view point and view direction in polar coordinates

SPECIFICATION

void Ppolar_view(Fcoord dist, Angle azim, Angle inc, Angle twist)

ARGUMENTS

dist	The distance from the origin
azim	The angle in the x-y plane
inc	The angle in the y-z plane
twist	The twist is the rotation around the z axis

RETURNS

Nothing

DESCRIPTION

Ppolar_view defines a viewing orientation transformation matrix. This routine is usually used in conjunction with either **Portho_view**, **Ppersp_view**, or **Pwindow_view**.
The transform is generated from the supplied parameters, where *dist* is the distance from the origin, *azim* is the angle in the x-y plane, *inc* is the angle in the y-z plane, *twist* is the rotation around the z axis.
The resulting transformation matrix replaces the top of the matrix stack.

NOTE

This routine cannot be used inside a structure.

SEE ALSO

Plookat_view, Portho_2d_view, Portho_view, Ppersp_view, Pwindow_view

NAME

Ppoly - Draw a vector polygon with n vertices

SPECIFICATION

void Ppoly(int n, Fcoord3D points[])

ARGUMENTS

n	The number of vertices in the *points* array
points	Array of floats that define the vertices of the polygon

RETURNS

Nothing

DESCRIPTION

Ppoly generates an outline (or vector) polygon, with *n* points stored in the array of vertices *points*. The last point is automatically joined to the first point, thereby closing the polygon. (Use **Ppoly_line** for a non-closed vector polygon). The outline is drawn in the current polygon color. If the normal for the polygon is not set with **Ppoly_normal**, then it is generated automatically.

SEE ALSO

Pcolor, Ppoly_fill, Ppoly_normal, Prectangle, Pshade_mode

NAME

Ppoly_close - Close a polygon

SPECIFICATION

int Ppoly_close(void)

ARGUMENTS

None

RETURNS

0 if successful, non-zero if an error occurred

DESCRIPTION

Ppoly_close closes the polygon being constructed with one of the **Ppoly_point**... commands. The type of the polygon will be determined by the preceding **Ppoly_point**... commands, but will always be a filled polygon and therefore must be convex. If the normal for the polygon is not set with **Ppoly_normal**, then it is generated automatically.

SEE ALSO

Ppoly_normal, Ppoly_point_2d, Ppoly_point_3d, Ppoly_point_nv, Ppoly_point_uv, Ppoly_point_nv_uv

NAME

Ppoly_fill - Draws a filled polygon with n vertices

SPECIFICATION

void Ppoly_fill(int n, Fcoord3D points[])

ARGUMENTS

n	The number of vertices in the *points* array
points	Array of floats that define the vertices of the polygon

RETURNS

Nothing

DESCRIPTION

Ppoly_fill generates a flat shaded polygon, with *n* points stored in the array of vertices *points*. The last point is automatically joined to the first point, thereby closing the polygon. The polygon is filled with the current color. Colors per vertex may be supplied using the **Pvertex_color** routine.
As with all filled polygons it must be convex, unless the shape flag is set to something other than PEXtk_SHAPE_CONVEX.

SEE ALSO

Ppoly_shade, Ppoly_shade_texture, Ppoly_texture, Pvertex_color

NAME

Ppoly_fill_area- Draw a set of polygon primitives

SPECIFICATION

```
void Ppoly_fill_area(long polyMask, long vertexMask, int numVerts, int colorType,
        Fcoord *polyData, Fcoord *vertexData)
```

ARGUMENTS

polyMask	A bitmask indicating the polygon attributes given for each polygon
vertexMask	A bitmask indicating the vertex attributes given for each polygon
numVerts	The number of vertices in the *vertexData* array
colorType	A integer representing the color type used for color data
**polyData*	Array of floats that define the polygon data for each polygon
**vertexData*	Array of floats that define the vertex data for each point in each polygon

RETURNS

Nothing

DESCRIPTION

Ppoly_fill_area generates a set of polygon primitives. This routine corresponds to the PHIGS/+ concept of a set of "Fill Area 3D with data". For this primitive, vertex data, such as colors per vertex and normals per vertex, are packed with the vertex point data.

When color data is provided, it must be in the form specified with *colorType*. The *colorType* must be one of PEXtk_COLOR_RGB, PEXtk_COLOR_INDEXED, PEXtk_COLOR_HSV, PEXtk_COLOR_HLS, or PEXtk_COLOR_CIE.

The *polyMask* indicates the data contained in the *polyData* list. The data must contain color values if the GACOLOR bit in *polyMask* is set, a polygon normal if the GANORMAL bit is set, or a color followed by a normal if both bits are set.

The *vertexMask* indicates the data contained in *vertexData*. Each entry must contain at least the x,y,z coordinate of the vertex (3 floats). Each entry must contain a coordinate followed by a color value if the GACOLOR bit in *vertexMask* is set (6 floats or 3 floats and an integer), a coordinate followed by a vertex normal if the GANORMAL bit is set (6 floats), or a coordinate followed by a color followed by a normal if both bits are set (9 floats or 6 floats and an integer).

For each polygon, the last point is automatically joined to the first point, thereby closing the polygon. The polygon is filled with the current color if no polygon color or vertex colors are given.

This primitive is affected by the current shade mode given in **Pshade_mode**. If the current shade mode is PEXtk_SHADE_WIRE, then the polygon is not filled. If the current shade mode does not include PEXtk_SHADE_WIRE, then it is filled, and as with all filled polygons it must be convex. If the current shade mode is PEXtk_SHADE_HIDDENLINE, then the polygon will appear as a wireframe polygon with hidden surfaces removed.

SEE ALSO

Pcolor, Pshade_mode

NAME

Ppoly_index - Draw an indexed polygon primitive

SPECIFICATION

```
void Ppoly_index(long polyMask, long vertexMask, int numPolys, int numVerts,
        int numEdges, int colorType, short *polyCounts, Fcoord *polyData, short *edges,
        Fcoord *vertexData)
```

ARGUMENTS

polyMask	A bitmask indicating the polygon attributes given for each polygon
vertexMask	A bitmask indicating the vertex attributes given for each polygon
numPolys	The number of polygons in the *points* array
numVerts	The number of vertices in the *points* array
numEdges	The total number of edges listed in the *edges* array
colorType	A integer representing the color type used for color data
polyCounts	Array of floats that contain polygon data
**edges*	Array containing the edge connectivity data
**vertexData*	Array of floats that define the vertex data for each point in each polygon

RETURNS

Nothing

DESCRIPTION

Ppoly_index generates a set of indexed polygon primitives. This routine corresponds to the PHIGS/+ concept of a “Set of Fill Area Sets”. For this primitive, vertex data, such as colors per vertex and normals per vertex, are packed with the vertex coordinate data.

When color data is provided, it must be in the form specified with *colorType*. The *colorType* must be one of PEXtk_COLOR_RGB, PEXtk_COLOR_INDEXED, PEXtk_COLOR_HSV, PEXtk_COLOR_HLS, or PEXtk_COLOR_CIE.

The *polyMask* indicates the data contained in the *polyData* list. There must be *numPolys* entries in the array pointed to by *polyData*. Each entry must contain color values if the GACOLOR bit in *polyMask* is set, a polygon normal if the GANORMAL bit is set, or a color followed by a normal if both bits are set.

The *vertexMask* indicates the data contained in *vertexData*. Each entry must contain at least the x,y,z coordinate of the vertex (3 floats). Each entry must contain a coordinate followed by a color value if the GACOLOR bit in *vertexMask* is set (6 floats, or 3 floats and an integer if in *Indexed Color* mode), a coordinate followed by a vertex normal if the GANORMAL bit is set (6 floats), or a coordinate followed by a color followed by a normal if both bits are set (9 floats, or 6 floats and an integer if in *Indexed Color* mode).

The *polyCounts* array has *numPolys* entries, and it contains the number of vertices in each polygon. For example, if the first entry in *polyCounts* has the value three and the second entry has the value four, the first polygon will have three vertices, and the first three entries in *edges* contain the index values of the polygon’s three vertices. The next four entries in *edges* contain the indices that define the second polygon. The index values in the *edges* array are zero based, meaning that the first vertex is vertex zero. For example, if in the above example the values in the *edges* array are {7,4,6,1,0,2,3}, then the second polygon is composed of the second, first, third, and fourth entries in the *vertexData* array. The total number of edges in the *edges* array is given by *numEdges*. If the value for *numEdges* is 0.0, then PEXtk will traverse the list and calculate the total.

For each polygon, the last point is automatically joined to the first point, thereby closing the polygon. The polygon is filled with the current color if no polygon color or vertex colors are given.

This primitive is affected by the current shade mode given in **Pshade_mode**. If the current shade mode is PEXtk_SHADE_WIRE, then the polygon is not filled. If the current shade mode does not include PEXtk_SHADE_WIRE, then it is filled, and as with all filled polygons it must be convex. If the current shade mode is PEXtk_SHADE_HIDDENLINE, then the polygon will appear as a wireframe polygon with hidden surfaces removed.

EXAMPLE

The following is an example of using **Ppoly_index**

```
Fcoord polyData[] ={0.0, 0.0, -1.0, 0.0, 0.0, 1.0};
short polyCounts[] = {4, 4};
short edges[] = {0, 1, 2, 3, 3, 2, 4, 5};
Fcoord vertexData[] = {-5.0, 5.0, -4.0, 1.0, 0.0, 0.0,
                       -5.0, -5.0, -4.0, 1.0, 0.0, 0.0,
                       0.0, -5.0, -4.0, 0.0, 1.0, 0.0,
                       0.0, 5.0, -4.0, 0.0, 1.0, 0.0,
                       5.0, -5.0, -4.0, 0.0, 0.0, 1.0,
                       5.0, 5.0, -4.0, 0.0, 0.0, 1.0};

Pshade_mode(PEXtk_SHADE_COLORS);
Ppoly_index(GANORMAL, GACOLOR, 2, 6, 8, PEXtk_COLOR_RGB,
        polyCounts, polyData, edges, vertexData);
```

This will produce two polygons, side by side, with the color of left side of the first polygon red, and linearly blending to green at the right side of the first polygon. The left side of the second polygon will be green and linearly blending to blue at the right side.

NOTE

The interface for this routine may change to take advantage of future releases of PEX.

SEE ALSO

Pcolor, Pdither, Pshade_mode

NAME

Ppoly_line - Draw a set of polylines

SPECIFICATION

void Ppoly_line(int n, Fcoord3D points[])

ARGUMENTS

n	The number of vertices in the *points* array
points	Array of floats that define the vertices of the polygon

RETURNS

Nothing

DESCRIPTION

Ppoly_line draws a series of *n*-1 polylines from the vertex array *points*.
If the lines describe a polygon, then it would need to be closed by making the last point the same as the first point (or use **Ppoly**).
Vertex colors may be specified with the routine **Pvertex_color_array**, which must precede the call to **Ppoly_line**. If they are specified, then they are used instead of the line color attribute set with **Pline_color** to determine the color of the line. They are utilized in two different ways, depending on the shading mode set with **Pshade_mode**. If the current shade mode does not include PEXtk_SHADE_COLORS, then the color at the i^{th} vertex is used to draw the line between the i^{th} and $(i+1)^{th}$ vertex. If the current shade mode does include PEXtk_SHADE_COLORS, then a linear interpolation is performed with the vertex colors to draw points along the line.

SEE ALSO

Pline_color, Ppoly, Pshade_mode, Pvertex_color_array

NAME

Ppoly_normal - Define a polygon normal vector

SPECIFICATION

void Ppoly_normal(Fcoord *x*, Fcoord *y*, Fcoord *z*)

ARGUMENTS

x	The normal x value
y	The normal y value
z	The normal z value

RETURNS

Nothing

DESCRIPTION

Ppoly_normal sets the normal for the next piece of geometry drawn. If a normal for a polygon is to be set, then this routine must be called before the relevant **Ppoly**... routine.

NOTE

Most geometry commands generate their own normal if one is not specified.
This is usually used to improve performance at rendering time if the polygon normal is already known.

SEE ALSO

Ppoly, Ppoly_fill, Ppoly_shade Ppoly_shade_normal. Ppoly_shade_texture

NAME

Ppoly_point_2d - enter a 2D polygon point

SPECIFICATION

void Ppoly_point_2d(Fcoord *x*, Fcoord *y*)

ARGUMENTS

x	The x coordinate value
y	The y coordinate value

RETURNS

Nothing

DESCRIPTION

Ppoly_point_2d enters a point into the polygon currently being constructed. The flat shaded filled polygon will be drawn when the **Ppoly_close** command is given. Using any other **Ppoly_point**... command before the **Ppoly_close** will have undefined results.

SEE ALSO

Pcolor, Ppoly_close, Ppoly_fill, Ppoly_point_3d

NAME

Ppoly_point_3d - enter a 3D polygon point

SPECIFICATION

void Ppoly_point_3d(Fcoord *x*, Fcoord *y*, Fcoord *z*)

ARGUMENTS

x	The x coordinate value
y	The y coordinate value
z	The z coordinate value

RETURNS

Nothing

DESCRIPTION

Ppoly_point_3d enters a point *x*, *y*, *z* into the polygon currently being constructed. The flat shaded filled polygon will be drawn when the **Ppoly_close** command is given. Using any other **Ppoly_point**... command before the **Ppoly_close** will have undefined results.

SEE ALSO

Pcolor, Ppoly_close, Ppoly_fill, Ppoly_point_2d

NAME

Ppoly_point_nv - enter a 3D polygon point with vector normal

SPECIFICATION

void Ppoly_point_nv(Fcoord *x*, Fcoord *y*, Fcoord *z*, Fcoord *nx*, Fcoord *ny*, Fcoord *nz*)

ARGUMENTS

x	The x coordinate value
y	The y coordinate value
z	The z coordinate value
nx	The normal x value
ny	The normal y value
nz	The normal z value

RETURNS

Nothing

DESCRIPTION

Ppoly_point_nv enters a point into the polygon currently being constructed. The “Gouraud” shaded polygon will be entered when the **Ppoly_close** command is given. The vertex normal *nx*, *ny* and *nz* is usually derived from the average of the polygon normals of all the polygons that share the vertex. Using any other **Ppoly_point**... command before the **Ppoly_close** will have undefined results.

SEE ALSO

Ppoly_close, Ppoly_shade

NAME

Ppoly_point_nv_uv - enter a 3D polygon point with normal and texture

SPECIFICATION

void Ppoly_point_nv_uv(Fcoord *x*, Fcoord *y*, Fcoord *z*, Fcoord *nx*, Fcoord *ny*, Fcoord *nz*,
Fcoord *ux*, Fcoord *uy*)

ARGUMENTS

x	The x coordinate value
y	The y coordinate value
z	The z coordinate value
nx	The normal x value
ny	The normal y value
nz	The normal z value
ux	The texture x value
uy	The texture y value

RETURNS

Nothing

DESCRIPTION

Ppoly_point_nv_uv enters a point into the polygon currently being constructed. The "Gouraud" shaded textured polygon will be drawn when the **Ppoly_close** command is given. The vertex normal *nx*, *ny* and *nz* is usually derived from the average of the polygon normals of all the polygons that share the vertex. The texture variables *ux*, *uy* are described in the Texture Section in the Overview. Using any other **Ppoly_point**... command before the **Ppoly_close** will have undefined results.

NOTE

Support for texture mapping is platform dependent.

SEE ALSO

Ppoly_close, Ppoly_shade_texture

NAME

Ppoly_point_uv - enter a 3D polygon point with texture variable

SPECIFICATION

void Ppoly_point_uv(Fcoord x, Fcoord y, Fcoord z, Fcoord ux, Fcoord uy)

ARGUMENTS

x	The x coordinate value
y	The y coordinate value
z	The z coordinate value
ux	The texture x value
uy	The texture y value

RETURNS

Nothing

DESCRIPTION

Ppoly_point_uv enters a point into the polygon currently being constructed. The texture mapped polygon will be drawn when the **Ppoly_close** command is given. The texture variables *ux*, *uy* are described in the Texture Section in the Overview. Use the **Ptexture**... commands to set up the texture mapping. Using any other **Ppoly_point**... command before the **Ppoly_close** will have undefined results.

NOTE

Support for texture mapping is platform dependent.

SEE ALSO

Ppoly_close, Ppoly_texture

NAME

Ppoly_shade - Draw a Gouraud shaded polygon

SPECIFICATION

void Ppoly_shade(int n, Fcoord3D points[], Fcoord3D vnarray[])

ARGUMENTS

n	The number of vertices in the *points* array
points	Array of floats that define the vertices of the polygon
vnarray	Array of floats that define the normals for each vertex

RETURNS

Nothing

DESCRIPTION

Ppoly_shade generates a Gouraud shaded polygon, with *n* points stored in the array of vertices *points*. The array *vnarray* stores the array of vertex normals associated with each vertex. Vertex normals are usually derived from the average of the polygon normals of all the polygons that share the vertex. There must be the same number of normals in the *vnarray* as there are points in the *points* array.
The last point is automatically joined to the first point, thereby closing the polygon. The polygon is filled with the current color.
If the normal for the polygon is not set with **Ppoly_normal**, then it is generated automatically. As with all filled polygons it must be convex, unless the shape flag is set to something other than PEXtk_SHAPE_CONVEX. Use **Pdither** to set the dither pattern to help reduce the effect of “Mach banding”.

SEE ALSO

Pcolor, Ppoly_normal

NAME

Ppoly_shade_texture - Draw a shaded textured polygon

SPECIFICATION

void Ppoly_shade_texture(int n, Fcoord3D points[], Fcoord3D vnarray[], Tvar tarray[])

ARGUMENTS

n	The number of vertices in the *points* array
points	Array of floats that define the vertices of the polygon
vnarray	Array of floats that define the normals for each vertex
tarray	Array of floats that define the texture vertices for each vertex

RETURNS

Nothing

DESCRIPTION

Ppoly_shade_texture generates a shaded, textured polygon, with *n* points stored in the array of vertices *points*. The array *vnarray* stores the array of vertex normals associated with each vertex. Vertex normals are usually derived from the average of the polygon normals of all the polygons that share the vertex. The *tarray* stores the array of texture variables associated with each vertex. Texture variables are described in the Texture Section in the Overview. Use the **Ptexture**... commands to set up the texture mapping.
The last point is automatically joined to the first point, thereby closing the polygon. If the normal for the polygon is not set with **Ppoly_normal**, then it is generated automatically. As with all filled polygons it must be convex, unless the shape flag is set to something other than PEXtk_SHAPE_CONVEX.

NOTE

Support for texture mapping is platform dependent.

SEE ALSO

Ppoly_close, Ppoly_normal

NAME

Ppoly_texture - Draw a textured polygon

SPECIFICATION

void Ppoly_texture(int n, Fcoord3D points[], Tvar tarray[])

ARGUMENTS

n	The number of vertices in the *points* array
points	Array of floats that define the vertices of the polygon
tarray	Array of floats that define the texture vertices for each vertex

RETURNS

Nothing

DESCRIPTION

Ppoly_texture generates a texture mapped polygon, with *n* points stored in the array of vertex coordinates *points*. The array *tarray* stores the array of texture variables associated with each vertex. Texture variables are described in the Texture Section in the Overview. The last point is automatically joined to the first point, thereby closing the polygon. If the normal for the polygon is not set with **Ppoly_normal**, then it is generated automatically. As with all filled polygons it must be convex, unless the shape flag is set to something other than PEXtk_SHAPE_CONVEX.

NOTE

Support for texture mapping is platform dependent.

SEE ALSO

Ppoly_close, Ppoly_normal, Ppoly_shade_texture

NAME

Ppoly_with_data - Defines a polygon with data

SPECIFICATION

void Ppoly_with_data(int type, int n, Fcoord3D points[], int ndata, void *poly_data)

ARGUMENTS

type	The user defined polygon type
n	The number of vertices in the *points* array
points	Array of floats that define the vertices of the polygon
ndata	The number of items in *poly_data*
poly_data	Pointer to type dependent data

RETURNS

Nothing

DESCRIPTION

Ppoly_with_data allows a user defined polygon *type* to be entered that has data attached to each vertex.The pointer *poly_data* points to an array that contains type dependent data. *ndata* is the amount of data (in bytes) that is attached to each vertex.
The last point is automatically joined to the first point, thereby closing the polygon. If the normal for the polygon is not set with **Ppoly_normal**, then it is generated automatically.

NOTE

This routine would be used for extensions to PEXtk.

SEE ALSO

Ppoly_normal, Ppoly_shade, Ppoly_shade_texture, Ppoly_texture

NAME

Ppop_attributes - Pop attributes off stack

SPECIFICATION

```
void Ppop_attributes(void)
```

ARGUMENTS

None

RETURNS

Nothing

DESCRIPTION

Ppop_attributes restores the attributes saved on the stack by a previous **Ppush_attributes** command.

NOTE

This routine cannot be used inside a structure.

SEE ALSO

Pcolor, Pline_color, Pload_attribute, Ppush_attributes

NAME

Ppop_matrix - Restore matrix

SPECIFICATION

void Ppop_matrix(void)

ARGUMENTS

None

RETURNS

Nothing

DESCRIPTION

Ppop_matrix restores the matrix stack that was saved by a previous **Ppush_matrix**.
If the matrix mode is PEXtk_MATRIX_VIEWING, the modeling transformation matrix is popped, otherwise the viewing/projection matrices are popped.

NOTE

This routine cannot be used inside a structure.
Two matrix stacks are maintained, the modeling and viewing matrix. The projection matrix itself is never modified directly, however it is multiplied by the viewing matrix.

SEE ALSO

Pload_matrix, Pmatrix_mode, Pmultiply_matrix, Ppush_matrix

NAME

Ppop_viewport - Pop the viewport stack

SPECIFICATION

```
void Ppop_viewport(void)
```

ARGUMENTS

None

RETURNS

Nothing

DESCRIPTION

Ppop_viewport restores the viewport that was saved by a previous **Ppush_viewport**.

NOTE

This routine cannot be used inside a structure.

SEE ALSO

Ppush_viewport, Pviewport

NAME

Ppush_attributes - Push attributes on stack

SPECIFICATION

void Ppush_attributes(void)

ARGUMENTS

None

RETURNS

Nothing

DESCRIPTION

Ppush_attributes saves the current attributes by duplicating the top of the stack. Effectively the stack is pushed down one and the old top is copied to the new top.

NOTE

This routine cannot be used inside a structure.

SEE ALSO

Pcolor, Pline_color, Pload_attribute, Ppop_attributes

NAME

Ppush_matrix - Save current transformation matrix

SPECIFICATION

void Ppush_matrix(void)

ARGUMENTS

None

RETURNS

Nothing

DESCRIPTION

Ppush_matrix saves the current transformation by duplicating the top of the stack. Effectively the stack is pushed down one and the old top is copied to the new top.
If the matrix mode is PEXtk_MATRIX_VIEWING, the modeling transformation matrix is pushed, otherwise the viewing/projection matrices are pushed.

NOTE

This routine cannot be used inside a structure.
Two matrix stacks are maintained, the modeling and viewing matrix. The projection matrix itself is never modified directly, however it is multiplied by the viewing matrix.

SEE ALSO

Pload_matrix, Pmatrix_mode, Pmultiply_matrix, Ppop_matrix

NAME

Ppush_viewport - Push the viewport stack

SPECIFICATION

```
void Ppush_viewport(void)
```

ARGUMENTS

None

RETURNS

Nothing

DESCRIPTION

Ppush_viewport saves the current viewport by duplicating the top of the stack. Effectively the stack is pushed down one and the old top is copied to the new top.

NOTE

This routine cannot be used inside a structure.

SEE ALSO

Ppop_viewport, Pviewport

NAME

Prectangle - Draw a rectangle

SPECIFICATION

void Prectangle(Fcoord x1, Fcoord y1, Fcoord x2, Fcoord y2)

ARGUMENTS

x1	The upper left x coordinate
y1	The upper left y coordinate
x2	The bottom right x coordinate
y2	The bottom right y coordinate

RETURNS

Nothing

DESCRIPTION

Prectangle draws an outlined rectangle in the current color. *x1*, *y1* is the upper left corner and *x2*, *y2* is the bottom right corner and z is 0. If the normal for the polygon is not set with **Ppoly_normal**, then it is generated automatically.

NOTE

This command is identical to calling **Ppoly** with appropriate coordinates.

SEE ALSO

Pcolor, Ppoly, Prectangle_fill

NAME

Prectangle_fill - Draw a filled rectangle

SPECIFICATION

void Prectangle_fill(Fcoord x1, Fcoord y1, Fcoord x2, Fcoord y2)

ARGUMENTS

x1	The upper left x coordinate
y1	The upper left y coordinate
x2	The bottom right x coordinate
y2	The bottom right y coordinate

RETURNS

Nothing

DESCRIPTION

Prectangle_fill draws a flat shaded rectangle in the current color. The values *x1*, *y1* are the upper left corner and *x2*, *y2* are the bottom right corner and z is 0. If the normal for the polygon is not set with **Ppoly_normal**, then it is generated automatically.

NOTE

This command is identical to calling **Ppoly_fill** with appropriate coordinates.

SEE ALSO

Pcolor, Ppoly_fill, Prectangle

NAME

Preflection_property - Sets the various polygon reflection coefficients

SPECIFICATION

void Preflection_property(int type, Fdata value)

ARGUMENTS

type	The type of reflection property specified.
value	The reflection property value to be used.

RETURNS

Nothing

DESCRIPTION

Preflection_property sets the current reflection property *type* to the value given by *value*. The currently supported types are

PEXtk_REFL_AMBIENT
PEXtk_REFL_DIFFUSE
PEXtk_REFL_SPECULAR
PEXtk_REFL_EXPONENT

The type PEXtk_REFL_AMBIENT specifies the ambient coefficient to be used in calculating the ambient contribution in the lighting computation (default is 0.5).The type PEXtk_REFL_DIFFUSE specifies the diffuse coefficient to be used in calculating the ambient contribution in the lighting computation (default is 0.7).The type PEXtk_REFL_SPECULAR specifies the specular coefficient to be used in calculating the ambient contribution in the lighting computation (default is 0.8).The type PEXtk_REFL_EXPONENT specifies the specular exponent (or specular concentration) to be used in calculating the specular contribution in the lighting computation (default is 0.0).

SEE ALSO

Plight, Plight_ambient, Pshade_mode, Pspecular_color

NAME

Preplace_struct - Positions the edit position to a label within a structure

SPECIFICATION

int Preplace_struct(int label)

ARGUMENTS

label	An integer representing the label at which elements are replaced

RETURNS

0 is successful, non-zero if an error occurs

DESCRIPTION

Preplace_struct overwrites the commands at position *label* in the structure that has been opened by **Pedit_struct**. An error will be generated if the structure is not open for edit or the *label* does not exist.

NOTE

Any changes made to the structure will not be reflected on the display until the structure is traversed again.

SEE ALSO

Pclose_struct, Pedit_struct, Pmake_label, Pinsert_struct

NAME

Protate - Rotate about a specified axis by a specified angle

SPECIFICATION

void Protate(Angle a, char axis, int postflag)

ARGUMENTS

a	The rotation angle, in degrees
axis	The rotation axis, 'x, 'y', or 'z'
postflag	Flag set to TRUE to perform a postconcatenation

RETURNS

Nothing

DESCRIPTION

Protate concatenates the current transformation matrix with one that causes a rotation of *a* degrees about an axis which is one of x','y'or'z'.If *postflag* is TRUE, then a postconcatenation is performed.
The old transformation matrix may be saved with **Ppush_matrix**.

NOTE

This routine simply calls the appropriate **Protate_x**, **Protate_y** or **Protate_z** routine.
Two matrix stacks are maintained, the modeling and viewing matrix. The projection matrix uses the same stack pointer as the viewing matrix.

SEE ALSO

Pload_matrix, Pmatrix_mode, Pmultiply_matrix

NAME

Protate_vector - Rotate about the specified vector

SPECIFICATION

```
void Protate_vector(Fcoord ox, Fcoord oy, Fcoord oz, Fcoord vx, Fcoord vy, Fcoord vz,
        Angle a, int postflag)
```

ARGUMENTS

ox	The vector start x value
oy	The vector start y value
oz	The vector start z value
vx	The vector end x value
vy	The vector end y value
vz	The vector end z value
a	The rotation angle, in degrees
postflag	Flag set to TRUE to perform a postconcatenation

RETURNS

Nothing

DESCRIPTION

Protate_vector concatenates the current transformation matrix with one that causes a rotation of *a* degrees about a vector defined by the points *ox*, *oy*, *oz*, *vx*, *vy*, *vz*. If *postflag* is true then a postconcatenation is performed.
The old transformation matrix may be saved with **Ppush_matrix**.

NOTE

Two matrix stacks are maintained, the modeling and viewing matrix. The projection matrix uses the same stack pointer as the viewing matrix.
This routine can only be used in matrix mode PEXtk_MATRIX_MODELING.

SEE ALSO

Pload_matrix, Pmatrix_mode, Pmultiply_matrix

NAME

Protate_x - Rotate about the x axis by a specified angle

SPECIFICATION

void Protate_x(Angle a, int postflag)

ARGUMENTS

a	The rotation angle, in degrees
postflag	Flag set to TRUE to perform a postconcatenation

RETURNS

Nothing

DESCRIPTION

Protate_x concatenates the current transformation matrix with one that causes a rotation of *a* degrees about the x axis. If *postflag* is TRUE, then a postconcatenation is performed. The old transformation matrix may be saved with **Ppush_matrix**.

NOTE

Two matrix stacks are maintained, the modeling and viewing matrix. The projection matrix uses the same stack pointer as the viewing matrix.

SEE ALSO

Pload_matrix, Pmatrix_mode, Pmultiply_matrix, Protate, Protate_y, Protate_z

NAME

Protate_y - Rotate about the y axis by a specified angle

SPECIFICATION

void Protate_y(Angle a, int postflag)

ARGUMENTS

a	The rotation angle, in degrees
postflag	Flag set to TRUE to perform a postconcatenation

RETURNS

Nothing

DESCRIPTION

Protate_y concatenates the current transformation matrix with one that causes a rotation of *a* degrees about the y axis. If *postflag* is TRUE, then a postconcatenation is performed. The old transformation matrix may be saved with **Ppush_matrix**.

NOTE

Two matrix stacks are maintained, the modeling and viewing matrix. The projection matrix uses the same stack pointer as the viewing matrix.

SEE ALSO

Pload_matrix, Pmatrix_mode, Pmultiply_matrix, Protate, Protate_x, Protate_z

NAME

Protate_z - Rotate about the z axis by a specified angle

SPECIFICATION

void Protate_z(Angle a, int postflag)

ARGUMENTS

a	The rotation angle, in degrees
postflag	Flag set to TRUE to perform a postconcatenation

RETURNS

Nothing

DESCRIPTION

Protate_z concatenates the current transformation matrix with one that causes a rotation of *a* degrees about the z axis. If *postflag* is TRUE, then a postconcatenation is performed. The old transformation matrix may be saved with **Ppush_matrix**.

NOTE

Two matrix stacks are maintained, the modeling and viewing matrix. The projection matrix uses the same stack pointer as the viewing matrix.

SEE ALSO

Pload_matrix, Pmatrix_mode, Pmultiply_matrix, Protate, Protate_x, Protate_y

NAME

Pscale - Scale x, y and z

SPECIFICATION

void Pscale(Fcoord x, Fcoord y, Fcoord z, int postflag)

ARGUMENTS

x	The scale x value
y	The scale y value
z	The scale z value
postflag	Flag set to TRUE to perform a postconcatenation

RETURNS

Nothing

DESCRIPTION

Pscale concatenates the current transformation matrix with one that causes a scaling of *x*, *y*, and *z*. If *postflag* is TRUE, then a postconcatenation is performed.
The old transformation matrix may be saved with **Ppush_matrix**.

NOTE

Two matrix stacks are maintained, the modeling and viewing matrix. The projection matrix uses the same stack pointer as the viewing matrix.

SEE ALSO

Pload_matrix, Pmultiply_matrix, Protate, Ptranslate_x, Ptranslate_y, Ptranslate_z

NAME

Pset_color_equation - Set values to use for calculating pixel values

SPECIFICATION

int Pset_color_equation(Colormap *cmap, int *base_pixel, int num_red, int num_green,
    int num_blue, int mult_red, int mult_green, int mult_blue, Fdata gamma_value)

ARGUMENTS

**cmap*	Pointer to an optional colormap id
**base_pixel*	A pointer to an integer representing the value of the base pixel
num_red	An integer representing the number of red values in ramp
num_green	An integer representing the number of green values in ramp
num_blue	An integer representing the number of blue values in ramp
mult_red	An integer representing the red coefficient value
mult_green	An integer representing the green coefficient value
mult_blue	An integer representing the blue coefficient value
gamma_value	Gamma correction value

RETURNS

0 if successful, non-zero if an error occurred

DESCRIPTION

Pset_color_equation sets the equation values to be used in computing the color value for a pixel. PEXtk needs this information to generate a color ramp. The equation used is:

pixel = base_pixel + (num_red-1)*mult_red + (num_green-1)*mult_green + (num_blue-1)*mult_blue

If *cmap* is not NULL, and if **cmap* is not zero, then the value of **cmap* is used as the colormap, and therefore a colormap will not be created.
If *basePixel* is not a NULL pointer, then if the value of **basePixel* is not < 0, this routine will attempt to place a pseudo colormap at the 'end' of the default colormap, after slot/location **base_pixel*. If **basePixel* is < 0, then PEXtk will attempt to use a 'good' base number of pixels from the existing colormap. If this cannot be done, a new colormap is created, and the first *base_pixel* number of entries in the default colormap are copied into the new colormap. If *cmap* is zero, then a new colormap is allocated. If **base_pixel* < 0, then PEXtk will attempt to put in an 'optimum' number of base cells from the default colormap into the new colormap, if one needs to be created. Note that a pointer to *base_pixel* is passed, since the value requested may not available, or when **base_pixel* < 0, then PEXtk will set the value. Applications may elect to have PEXtk handle the colormap creation, and the choosing of the basePixel value, and to do this they would simply pass in NULL for **cmap* and NULL for **basePixel*.

NOTE

This routine is intended to be used only for 8-bit color systems, and may not be used in a structure or display list

SEE ALSO

Pcolor, Plight_color, Pline_color, Pspecular_color

NAME

Pset_color_mode - Sets the color mode to Indexed, RGB or HSV

SPECIFICATION

int Pset_color_mode(int mode)

ARGUMENTS

mode	The color mode, PEXtk_COLOR_INDEXED, PEXtk_COLOR_RGB, or PEXtk_COLOR_HSV

RETURNS

Nothing

DESCRIPTION

Pset_color_mode sets *mode* as the color mode. For a *mode* of PEXtk_COLOR_INDEXED, a color table should be setup with **Pset_color_table**, and the **P_...ind** routines should be used to set the colors for primitives. For a *mode* of PEXtk_COLOR_RGB, the **P_...rgb** routines should be used, and for a *mode* of PEXtk_COLOR_HSV the **P..._hsv** routines should be used.

NOTE

The **Pcolor**, **Pline_color**, **Pmarker_color**, **Pspecular_color**, and **Ptext_color** routines must be passed with a color type that is the same as the current color mode.
PEXtk currently supports only Indexed and RGB modes.
This routine cannot be used inside a structure or Display List.

SEE ALSO

Pset_color_equation, Pset_color_table

NAME

Pset_color_table - Enter color into color table
Pset_color_table_rgb - Enter RGB color into color table
Pset_color_table_hsv - Enter HSV color into color table

SPECIFICATION

```
void Pset_color_table(int ind, PCOLOR *col)
void Pset_color_table_rgb(int ind, Fdata red, Fdata green, Fdata blue)
void Pset_color_table_hsv(int ind, Fdata hue, Fdata sat, Fdata value)
```

ARGUMENTS

ind	An integer representing the table index
**col*	Pointer to a PCOLOR structure containing the RGB, HSV, or CIE values to insert into the table
red,green,blue	The floating point values representing the red, green, blue
hue	The HSV color hue
sat	The HSV color saturation
value	The HSV color value (intensity)

RETURNS

Nothing

DESCRIPTION

Pset_color_table sets table entry *ind* to *col*.
Pset_color_table_hsv sets table entry *ind* to the color specified by the HSV color cone *hue*, *saturation* and *value* where hue is between 0.0 and 360.0, *saturation* is between 0.0 and 1.0 and *value* is between 0.0 and 1.0.
Pset_color_table_rgb sets table entry *ind* to the color specified by *red*, *green, blue* with values ranging between 0.0 and 1.0.

NOTE

The use of this routine will put PEXtk into *Indexed Color* mode. Colors should then be specified with one of the **P.._ind** routines.
The color table must be set with **Pset_color_table** before the indices are referenced by the **Pcolor_ind**, **Pclear_ind**, and **Pline_color_ind** routines.

SEE ALSO

Pcolor_ind, Pclear_ind, Pline_color_ind, Pmarker_color_ind, Pspecular_color_ind, Ptext_color_ind

NAME

Pset_colormap - Sets the colormap to be used for 8-bit systems

SPECIFICATION

```
int Pset_colormap(Colormap cmap)
```

ARGUMENTS

cmap	The colormap to be used

RETURNS

Nothing

DESCRIPTION

Pset_colormap sets *cmap* as the colormap. PEXtk only uses this for 8-bit systems where a pseudo colormap needs to be used. This routine will not perform an *XSetWindowColormap*. If a colormap is set before **Pset_drawable** is called, then **Pset_color_equation** will not be called by **Pset_drawable**. It will be up to the application to create its own pseudo colormap.

NOTE

This routine cannot be used inside a structure or Display List.

SEE ALSO

Pset_color_equation, Pset_color_table, Pset_drawable

NAME

Pset_drawable - Sets the destination drawable which receives output

SPECIFICATION

int Pset_drawable(Window id)

ARGUMENTS

id	The X window id for the destination of rendered primitives

RETURNS

0 if successful, non-zero if an error occurred

DESCRIPTION

Pset_drawable sets *id* as the destination for all graphics primitives. This must be called before any PEXtk primitive routines are invoked, and should be called immediately after **Pinit** is called.
This routine will call **Pset_color_equation** for 8-bit systems and setup a 6,6,6 colormap if the colormap has not been specified with **Pset_colormap**.

NOTE

There will only be one drawable active at a time.
This routine cannot be used inside a structure.

SEE ALSO

Pbuffer_mode, Pset_color_equation, Pset_colormap, Pset_display_buffer, Pset_update_buffer

NAME

Pset_function - Sets the PEXtk support function pointers

SPECIFICATION

int Pset_function(int type, int (*func)())

ARGUMENTS

type	An integer representing the type of function to set
*(*func)()*	The function pointer of the function to use

RETURNS

0 if successful, non-zero if an error occurred

DESCRIPTION

Pset_function sets the PEXtk support function of type *type* to the function pointer specified with (*func*)(). Available values for *type* are:

Type values	**PEXtk function**	**Description**
`PEXtk_FUNC_CLEAR_BUFFER`	*	Sets the buffer clearing function
`PEXtk_FUNC_CREATE_BUFFER`	Pbuffer_mode	Sets the backbuffer creation function
`PEXtk_FUNC_DELETE_BUFFER`	Pexit	Sets the backbuffer deletion function
`PEXtk_FUNC_SWAP_BUFFER`	Pswap_buffers	Sets the swap buffer function
`PEXtk_FUNC_PIXEL_INDEX`	Pclear,Pclear_ind	Returns a pixel value given an index color
`PEXtk_FUNC_PIXEL_RGB`	Pclear,Pclear_rgb	Returns a pixel value given RGB (3 Fdata)
`PEXtk_FUNC_DITHER`	Pdither	Sets the dithering function
`PEXtk_FUNC_TRANSPARENCY`	Ptransparency	Sets the transparency function

* gets called by **Pclear, Pclear_ind**, **Pclear_hsv**, or **Pclear_rgb** to clear the buffer

The PEXtk_FUNC_CLEAR_BUFFER routine will be called by PEXtk when one of the **Pclear** routines is called. For a color mode of PEXtk_COLOR_INDEXED, the routine will be passed one integer, for a color mode of PEXtk_COLOR_RGB or PEXtk_COLOR_HSV, the routine will be passed 3 floats.
The PEXtk_FUNC_PIXEL_INDEX routine will be passed an index, and is called by **Pclear** or **Pclear_ind**. The PEXtk_FUNC_PIXEL_RGB routine will be passed 3 floats, and it is called by **Pclear** or **Pclear_rgb**. They should return a pixel value.

NOTE

This routine cannot be used inside a structure or display list.

SEE ALSO

Pdither, Pset_update_buffer, Pset_display_buffer,Pswap_buffers, Ptransparency

NAME

Pset_precision - Sets the number of segments for circles, arcs, and spheres

SPECIFICATION

void Pset_precision(int u_segs, int v_segs)

ARGUMENTS

u_segs	The number of segments to use in the *u* direction.
v_segs	The number of segments to use in the *v* direction.

RETURNS

Nothing

DESCRIPTION

Pset_precision sets the number of line segments to be used when drawing circle, arc and sphere primitives. The *u_segs* value sets the number of segments to be used by circles and arcs, and also sets the number of segments for a sphere in the horizontal direction. The *v_segs* value sets the number of segments for a sphere in the vertical direction.
The current precision values are an attribute and are saved by **Ppush_attributes** and restored with **Ppop_attributes**.

SEE ALSO

Parc, Parc_fill, Pcircle, Pcircle_fill, Psphere, Psphere_fill

NAME

Pset_view_index - Sets the current viewing id

SPECIFICATION

int Pset_view_index(int id)

ARGUMENTS

id	The view id

RETURNS

0 if successful, non-zero if an error occurred

DESCRIPTION

Pset_view_index sets *id* as the current view. The *id* must be a value which was specified in creating a viewing projection matrix with **Portho_2d_view**, **Portho_view**, **Ppersp_view**, or **Pwindow_view**. The viewing index must be between 1 to PEXtk_VIEW_SIZE. Note that if the viewing projection has an associated viewing orientation, it will be activated also. This routine also sets the view stack pointer. If any subsequent viewing routine is called with an *id* of zero, then the *id* passed to **Pset_view_index** will be used. For example,

```
Ppersp_view( 2, fov, aspect, near, far);
```

is identical to:

```
Pset_view_index(2);
Ppersp_view( 0, fov, aspect, near, far);
```

which is also identical to

```
Pmatrix_mode(PEXtk_MATRIX_VIEWING);
Ppush_matrix();
Ppush_matrix();
Ppersp_view( 0, fov, aspect, near, far);
Ppop_matrix();
Ppop_matrix();
```

When using **Ppush_matrix**/**Ppop_matrix** however, the current viewing parameters are duplicated to the top of the stack, however in the above example they are immediately overwritten.

NOTE

The only method to specify the viewing parameters for a view id of zero, is to use this routine in conjunction with one of the **P..._view** routines.

SEE ALSO

Portho_2d_view, Portho_view, Ppersp_view, Pwindow_view

NAME

Pshade_mode - Sets the shading mode

SPECIFICATION

void Pshade_mode(int mode)

ARGUMENTS

mode	A bit mask representing the shade mode

RETURNS

Nothing

DESCRIPTION

Pshade_mode sets the current shading mode to *mode*. The shading mode describes which light effects and rendering attributes are to be applied to a polygon or polyline. Different shading modes affect different primitives.

The **Ppoly**, **Ppoly_fill** and **Ppoly_shade**... routines are affected by the shading modes PEXtk_SHADE_SPECULAR and PEXtk_SHADE_LIGHTS.

A light specified by **Plight** becomes effective only when PEXtk_SHADE_LIGHTS is specified in the bitmask. When PEXtk_SHADE_LIGHTS is turned on, the polygon color uses the color defined by Pcolor for ambient and diffuse, and the color specified by **Pspecular_color** for the specular highlight color of the polygon.

All of the shading modes affect **Ppoly_index**, **Ppoly_fill_area** and **Ppoly_with_data**. When *mode* is set to PEXtk_SHADE_WIRE and **Ppoly_index/Ppoly_fill_area/Ppoly_with_data** is invoked, a polygon outline (edges only) is created using the color specified with **Pcolor**. The color of the polygon is affected by any active lights if PEXtk_SHADE_LIGHTS is used.

The other cases involve a bit mask for *mode*. By *or*ing certain values to *mode*, subsequent polygons created with **Ppoly_index/Ppoly_fill_area/Ppoly_with_data** will take on the following attributes:

PEXtk_SHADE_NONE	No attributes are defined
PEXtk_SHADE_WIRE	Edges only, polygon is not filled
PEXtk_SHADE_FLAT	Turns off color interpolation
PEXtk_SHADE_COLORS	Turns on color interpolation
PEXtk_SHADE_DOTPRODUCT	Turns on dot product interpolation
PEXtk_SHADE_NORMALS	Turns on normal interpolation
PEXtk_SHADE_SPECULAR	Turns on specular highlights
PEXtk_SHADE_HIDDENLINE	Turns on Hiddenline mode
PEXtk_SHADE_ANTIALIAS	Turns on line antialiasing mode
PEXtk_SHADE_LIGHTS	Turns on lighting mode

The current shading mode may be saved with **Ppush_attributes**.

NOTE

Different hardware will support different shading modes. This routine cannot be used inside a structure.

SEE ALSO

Pget_shade_mode, Plight, Plight_ambient, Preflection_property, Pspecular_color

NAME

Pshape_flag - Sets the shape type for a polygon

SPECIFICATION

void Pshape_flag(int flag)

ARGUMENTS

flag	An integer which represents the polygon shape flag

RETURNS

Nothing

DESCRIPTION

Pshape_flag sets the shape flag to use for all subsequent polygon drawing routines. The allowable values for *flag* are:

PEXtk_SHAPE_COMPLEX
PEXtk_SHAPE_NONCONVEX
PEXtk_SHAPE_CONVEX
PEXtk_SHAPE_UNKNOWN

A shape of *PEXtk_SHAPE_COMPLEX* implies that the polygon has self-intersecting edges. A shape of *PEXtk_SHAPE_NONCONVEX* implies that the polygon has no self-intersecting edges, but all of the interior angles may not be convex. A shape of *PEXtk_SHAPE_CONVEX* implies that all interior angles are convex. A shape of *PEXtk_SHAPE_UNKNOWN* implies that no information is known about the polygon, and the PEX server will either attempt to determine its shape, or it will assume *PEXtk_SHAPE_COMPLEX*. The default shape is PEXtk_SHAPE_CONVEX.

NOTE

This routine provides the user with the ability to specify non-convex polygons. The use of a shape flag of anything other than PEXtk_SHAPE_CONVEX may result in a substantial decrease in performance.

SEE ALSO

Ppoly, Ppoly_fill, Ppoly_fill_area, Ppoly_index, Ppoly_shade

NAME

Pspecular_color - Sets the specular highlight color for a polygon
Pspecular_color_ind - Set the specular highlight index color for a polygon
Pspecular_color_hsv - Set the specular highlight hsv color for a polygon
Pspecular_color_rgb - Set the specular highlight rgb color for a polygon

SPECIFICATION

```
void Pspecular_color(PCOLOR *col)
void Pspecular_color_ind(int colind)
void Pspecular_color_hsv(Fdata hue, Fdata saturation, Fdata value)
void Pspecular_color_rgb(Fdata red, Fdata green, Fdata blue)
```

ARGUMENTS

**col*	Pointer to a PCOLOR structure containing the RGB, HSV, or CIE values
colind	An integer representing the color index
hue	A float representing the color hue
saturation	A float representing the color saturation
value	A float representing the color value
red,green,blue	Floats representing the RGB colors

RETURNS

Nothing

DESCRIPTION

Pspecular_color sets the specular color to *co*l.
Pspecular_color_ind sets the specular color to the index *ind*.
Pspecular_color_hsv sets the specular color to the HSV color specified by *hue*, *saturation*, *value* where *hue* is between 0.0 and 360.0, *saturation* is between 0.0 and 1.0 and *value* is between 0.0 and 1.0.
Pspecular_color_rgb sets the specular color to the RGB color specified by *red*, *green*, *blue* with values ranging between 0.0 and 1.0.
This routine specifies the color to be used in calculating the color of the polygon contributed from the specular component of the light source. The specular coefficient is specified with **Preflection_property**.

NOTE

The color table must be set with **Pset_color_table** before the index is referenced by the **Pspecular_color_ind** routine.
The specular highlight does not take effect unless the shade mode includes PEXtk_SHADE_SPECULAR.
The current specular color is an attribute and is saved by **Ppush_attributes** and restored with **Ppop_attributes**.

SEE ALSO

Preflection_property, Pshade_mode, Plight

NAME

Psphere - Draws a sphere primitive

SPECIFICATION

void Psphere(Fcoord x, Fcoord y, Fcoord z, Fcoord radius)

ARGUMENTS

x	The x coordinate of the center of the sphere
y	The y coordinate of the center of the sphere
z	The z coordinate of the center of the sphere
radius	The radius of the sphere

RETURNS

Nothing

DESCRIPTION

Psphere draws an outlined sphere where *x*, *y*, *z* is the center of the sphere, and the radius is defined by *radius*. The color is taken from the current line color attribute.
The number of line segments used to draw the sphere is set with **Pset_precision**.

NOTE

A sphere is a line primitive.

SEE ALSO

Pset_precision, Psphere_fill, Pline_color

NAME

Psphere_fill - Draws a filled sphere primitive

SPECIFICATION

void Psphere_fill(Fcoord x, Fcoord y, Fcoord z, Fcoord radius)

ARGUMENTS

x	The x coordinate of the center of the sphere
y	The y coordinate of the center of the sphere
z	The z coordinate of the center of the sphere
radius	The radius of the sphere

RETURNS

Nothing

DESCRIPTION

Psphere_fill draws a filled sphere where *x*, *y*, *z* is the center of the sphere, and the radius is defined by *radius*. The color is taken from the current color attribute.
The number of polygon segments used to draw the sphere is set with **Pset_precision**.

NOTE

A filled sphere is a polygon primitive.

SEE ALSO

Pset_precision, Psphere, Pcolor

NAME

Pstruct_used - Indicates whether a structure exists

SPECIFICATION

Boolean Pstruct_used(int name)

ARGUMENTS

name	An integer representing the structure name

RETURNS

0 if structure name is not in use, 1 if the structure name is in use

DESCRIPTION

Pstruct_used returns TRUE (1) if the structure *name* is currently in use. Otherwise it returns FALSE (0).

NOTE

This routine cannot be used inside a structure.

SEE ALSO

Plabel_used, Pcreate_struct

NAME

Psurface_approx_method - Sets surface approximation method and tolerance

SPECIFICATION

void Psurface_approx_method(int meth, Fdata u_tol, v_tol)

ARGUMENTS

meth	The surface approximation method to be used when tessellating surface primitives
u_tol	The surface tolerance in u direction
v_tol	The surface tolerance in v direction

RETURNS

Nothing

DESCRIPTION

Psurface_approx_method sets the surface approximation method and approximation tolerance criteria to be used when tessellating **Pnurb_surface** primitives. Available methods are:

PEXtk_APPROX_IMP_DEP	This technique will vary from one platform to another, but it is always available.
PEXtk_APPROX_CONSTANT	Surface is tessellated with equal parametric increments between successive pairs of knots. Only the integer portion of the tolerance is used. If *(u_tol,v_tol)* <= 0, the surface is evaluated at the parameter limits and at those knots within the parameter limits. If *(u_tol,v_tol)* > 0, the surface is evaluated at the parameter limits, at the knots within the parameter limits, and at the number of intervals specified by *(u_tol,v_tol)* between each pair of knots.
PEXtk_APPROX_WC_CHORD_LEN	Surface is tessellated until the length of each line segment in world coordinates is less than *(u_tol,v_tol)*.
PEXtk_APPROX_NC_CHORD_LEN	Surface is tesselated until the length of each line segment in normalized projection coordinates is less than *(u_tol,v_tol)*.
PEXtk_APPROX_DC_CHORD_LEN	Surface is tesselated until the length of each line segment in device coordinates is less than *(u_tol,v_tol)*.
PEXtk_APPROX_WC_CHORD_DEV	Surface is tesselated until the maximum deviation in world coordinates is less than (*u_tol,v_tol*).
PEXtk_APPROX_NC_CHORD_DEV	Surface is tesselated until the maximum deviation in normalized projection coordinates is less than *(u_tol,v_tol)*.
PEXtk_APPROX_DC_CHORD_DEV	Surface is tesselated until the maximum deviation in device coordinates is less than (*u_tol,v_tol*).

PEXtk_APPROX_WC_RELATIVE	Surface is tesselated using a relative level of quality in world coordinates.
PEXtk_APPROX_NC_RELATIVE	Surface is tesselated using a relative level of quality in normalized projection coordinates.
PEXtk_APPROX_DC_RELATIVE	Surface is tesselated using a relative level of quality in device coordinates.

The current surface approximation method and tolerance is an attribute and is saved by **Ppush_attributes** and restored with **Ppop_attributes**.

SEE ALSO

Pcurve_approx_method, Pnurb_curve, Pnurb_surface

NAME

Pswap_buffers - swap buffers
Pset_display_buffer - set the current display buffer
Pset_update_buffer - set the current update buffer

SPECIFICATION

```
void Pswap_buffers(void)
void Pset_display_buffer(int buf)
void Pset_update_buffer(int buf)
```

ARGUMENTS

buf	An integer representing the buffer id

RETURNS

Nothing

DESCRIPTION

Pswap_buffers makes the current update buffer the current display buffer and vice versa. This is used when the current buffer is finished and it should become visible in double buffer mode.
Pset_display_buffer sets buffer *buf* to the current display buffer.
Pset_update_buffer sets buffer *buf* to the current update buffer.

NOTE

There can only be one display buffer.
This routine cannot be used inside a structure.

SEE ALSO

Pbuffer_mode

NAME

Pswap_tmesh - Swaps the last two triangle mesh vertices

SPECIFICATION

void Pswap_tmesh(void)

ARGUMENTS

None

RETURNS

Nothing

DESCRIPTION

Pswap_tmesh swaps the last two saved vertices (i.e., from **Pvertex**) when in triangle mesh mode.

SEE ALSO

Pbegin_tmesh, Pend_tmesh, Pvertex

NAME

Ptext - Write a text string as stroke text
Ptext_size - set text size

SPECIFICATION

void Ptext(Fcoord xpos, Fcoord ypos, Fcoord zpos, char *string)
void Ptext_size(Fdata xsize, Fdata ysize)

ARGUMENTS

xpos	The text x value
ypos	The text y value
zpos	The text z value
string	The text string
xsize	Horizontal scale to be applied to font used for text string
ysize	Vertical scale to be applied to font used for text string

RETURNS

Nothing

DESCRIPTION

Ptext writes *string* as stroke text at *xpos*, *ypos*, *zpos*.The position of the text is also affected by the current modeling transformation matrix (**Protate**, **Ptranslate**, **Pload_matrix**).
Ptext_size scales the text in the horizontal direction by x*size,* and in the vertical direction by *ysize*. The default text size is *xsize* = *ysize* = 0.25.

NOTE

Ptext is stroke text that can be oriented in any 3D direction. It may be rotated to be non-parallel to the viewing plane. The visibility of the text is affected by the current viewport (clipping) and the z buffer state (on or off) set with **Pzbuffer**.

SEE ALSO

Pzbuffer, Portho_2d_view, Ptext_color, Ptext_orientation, Ptext_font, Ptext_path, Pview-port

NAME

Ptext_annotate - Write a text string as stroke text
Ptext_annotate_size - set text size

SPECIFICATION

void Ptext_annotate(Fcoord xpos, Fcoord ypos, Fcoord zpos, char *string)
void Ptext_annotate_size(Fdata xsize, Fdata ysize)

ARGUMENTS

xpos	The text x value
ypos	The text y value
zpos	The text z value
string	The text string
xsize	Horizontal scale to be applied to font used for text string
ysize	Vertical scale to be applied to font used for text string

RETURNS

Nothing

DESCRIPTION

Ptext_annotate writes *string* as stroke text at *xpos*, *ypos*, *zpos*.The position of the text is also affected by the current modeling transformation matrix (**Protate**, **Ptranslate**, **Pload_matrix**), but the text is always parallel to the viewing plane.
Ptext_annotate_size scales the text in the horizontal direction by x*size*, and in the vertical direction by *ysize*. The default size is *xsize* = *ysize* = 0.01.

NOTE

Usually you will use **Portho_2d_view** and **Pviewport** to set up a window for text. The visibility of the text is affected by the current viewport (clipping) and z buffer state (on or off).

SEE ALSO

Pzbuffer, Portho_2d_view, Ptext_color, Ptext_orientation, Ptext_font, Ptext_path

NAME

Ptext_font - Sets the text font
Ptext_path - Sets the text direction
Ptext_orientation - Sets the text up vector

SPECIFICATION

void Ptext_font(int font)
void Ptext_path(int path)
void Ptext_orientation(Fcoord x, Fcoord y)

ARGUMENTS

font	Font to be used for text string
path	An integer representing the direction along which the text string is written.
x,y	Two floats describing the up direction for the annotation text string

RETURNS

Nothing

DESCRIPTION

Ptext_font sets the text font to *font*.
Ptext_path specifies the writing direction of the text string. The *path* can be one of:

PEXtk_TEXT_PATH_UP
PEXtk_TEXT_PATH_DOWN
PEXtk_TEXT_PATH_LEFT
PEXtk_TEXT_PATH_RIGHT

The default path is PEXtk_TEXT_PATH_RIGHT.
Ptext_orientation specifies the text up direction with the vector *vec*. The default up direction is (0.0, 1.0). The position of the text is also affected by the current transformation matrix (**Protate**, **Ptranslate**, **Pload_matrix**).

NOTE

These routines affect **Ptext** and **Ptext_annotate**.

SEE ALSO

Pzbuffer, Portho_2d_view, Ptext, Ptext_annotate

NAME

Ptext_color - Set the text color

SPECIFICATION

void Ptext_color(PCOLOR *col)
void Ptext_color_ind(int colind)
void Ptext_color_hsv(Fdata hue, Fdata saturation, Fdata value)
void Ptext_color_rgb(Fdata red, Fdata green, Fdata blue)

ARGUMENTS

**col*	Pointer to a PCOLOR structure containing the RGB, HSV or CIE values
colind	An integer representing the color index
hue	A float representing the color hue
saturation	A float representing the color saturation
value	A float representing the color value
red,green,blue	Floats representing the RGB colors

RETURNS

Nothing

DESCRIPTION

Ptext_color sets the text color to *col* for all **Ptext** calls that follow.
Ptext_color_ind sets the text color to the index *colind*.
Ptext_color_hsv sets the text color specified by *hue*, *saturation* and *value* where *hue* is between 0.0 and 360.0, *saturation* is between 0.0 and 1.0 and *value* is between 0.0 and 1.0.
Ptext_color_rgb sets the text color specified by *red*, *green*, *blue* with values ranging between 0.0 and 1.0.

NOTE

The color table must be set with **Pset_color_table** before the index is referenced by the **Ptext_color_ind** routine.
The current text color is an attribute and is saved by **Ppush_attributes** and restored with **Ppop_attributes**.

SEE ALSO

Portho_2d_view, Ptext, Ptext_annotate

NAME

Ptexture_info - Sets the texture mapping information

SPECIFICATION

void Ptexture_info(long tb, long tc, int tn)

ARGUMENTS

tb	The texture bank
tc	An id which is the colormap to use for the texture
tn	The name to use for the texture map

RETURNS

Nothing

DESCRIPTION

Ptexture_info sets the texture bank to *tb*, the texture color lookup table to *tc*, and the texture name to *tn*.

NOTE

Support for texture mapping is platform dependent.

SEE ALSO

Ptexture_name, Ppoly_shade_texture, Ppoly_texture, Pvertex_texture

NAME

Ptexture_name - Sets the texture mapping name

SPECIFICATION

```
void Ptexture_name(int tn)
```

ARGUMENTS

tn	The name to use for the texture map

RETURNS

Nothing

DESCRIPTION

Ptexture_name sets the texture name to *tn*.

NOTE

Support for texture mapping is platform dependent.

SEE ALSO

Ptexture_info, Ppoly_shade_texture, Ppoly_texture, Pvertex_texture

NAME

Ptranslate - Translate x,y, and z

SPECIFICATION

void Ptranslate(Fcoord x, Fcoord y, Fcoord z, int postflag)

ARGUMENTS

x	The translation x value
y	The translation y value
z	The translation z value
postflag	Flag set to TRUE to perform a postconcatenation

RETURNS

Nothing

DESCRIPTION

Ptranslate concatenates the current transformation matrix with one that causes a translation of *x*, *y*, *z*. If *postflag* is TRUE, then a postconcatenation is performed.
The old transformation matrix may be saved with **Ppush_matrix**.

NOTE

Two matrix stacks are maintained, the modeling and viewing matrix. The projection matrix uses the same stack pointer as the viewing matrix.

SEE ALSO

Pload_matrix, Pmatrix_mode, Pmultiply_matrix, Ptranslate_x, Ptranslate_y, Ptranslate_z

NAME

Ptranslate_x - Translate x

SPECIFICATION

void Ptranslate_x(Fcoord x, int postflag)

ARGUMENTS

x	The translation x value
postflag	Flag set to TRUE to perform a postconcatenation

RETURNS

Nothing

DESCRIPTION

Ptranslate_x concatenates the current transformation matrix with one that causes a translation of *x*. If *postflag* is TRUE, then a postconcatenation is performed.
The old transformation matrix may be saved with **Ppush_matrix**.

NOTE

Two matrix stacks are maintained, the modeling and viewing matrix. The projection matrix uses the same stack pointer as the viewing matrix.

SEE ALSO

Pload_matrix, Pmatrix_mode, Pmultiply_matrix, Ptranslate, Ptranslate_y, Ptranslate_z

NAME

Ptranslate_y - Translate y

SPECIFICATION

void Ptranslate_y(Fcoord y, int postflag)

ARGUMENTS

y	The translation y value
postflag	Flag set to TRUE to perform a postconcatenation

RETURNS

Nothing

DESCRIPTION

Ptranslate_y concatenates the current transformation matrix with one that causes a translation of *y*. If *postflag* is TRUE, then a postconcatenation is performed.
The old transformation matrix may be saved with **Ppush_matrix**.

NOTE

Two matrix stacks are maintained, the modeling and viewing matrix. The projection matrix uses the same stack pointer as the viewing matrix.

SEE ALSO

Pload_matrix, Pmatrix_mode, Pmultiply_matrix, Ptranslate, Ptranslate_x, Ptranslate_z

NAME

Ptranslate_z - Translate z

SPECIFICATION

void Ptranslate_z(Fcoord z, int postflag)

ARGUMENTS

z	The translation z value
postflag	Flag set to TRUE to perform a postconcatenation

RETURNS

Nothing

DESCRIPTION

Ptranslate_z concatenates the current transformation matrix with one that causes a translation of *z*. If *postflag* is TRUE, then a postconcatenation is performed.
The old transformation matrix may be saved with **Ppush_matrix**.

NOTE

Two matrix stacks are maintained, the modeling and viewing matrix. The projection matrix uses the same stack pointer as the viewing matrix.

SEE ALSO

Pload_matrix, Pmatrix_mode, Pmultiply_matrix, Ptranslate, Ptranslate_x, Ptranslate_y

NAME

Ptransparency - Turns transparency rendering on or off

SPECIFICATION

void Ptransparency(int mode, Fdata value)

ARGUMENTS

mode	An integer representing the transparency mode
value	A float in range of 0.0 to 1.0, where 0.0 is opaque

RETURNS

Nothing

DESCRIPTION

Ptransparency turns transparency rendering on or off. If mode is TRUE, then transparency rendering is enabled if it is supported on the platform. The transparency value must be in the range of 0.0 to 1.0, where 0.0 represents a fully opaque primitive, and 1.0 represents a fully transparent primitive.

NOTE

Support for transparency is platform dependent.

SEE ALSO

Pload_matrix, Pmultiply_matrix, Protate, Ptranslate, Ptranslate_x, Ptranslate_y

NAME

Pvertex - Enter a vertex into current geometry

SPECIFICATION

void Pvertex(Fcoord vert[3])

ARGUMENTS

vert[3]	The vertex to add to the geometry

RETURNS

Nothing

DESCRIPTION

Pvertex inserts *vert* into the geometry which is currently being constructed.

SEE ALSO

Pbegin_line, Pbegin_poly, Pbegin_tmesh, Pend_line, Pend_poly, Pend_tmesh

NAME

Pvertex_color - Enter a color for a vertex
Pvertex_color_array - Enter a color for each vertex

SPECIFICATION

```
void Pvertex_color(PCOLOR *col)
void Pvertex_color_ind(int ind)
void Pvertex_color_hsv(Fdata hue, Fdata sat, Fdata value)
void Pvertex_color_rgb(Fdata red, Fdata green, Fdata blue)
void Pvertex_color_array(int n, PCOLOR col_array[])
void Pvertex_color_array_ind(int n, int ind_array[])
void Pvertex_color_array_hsv(int n, hsvFloat hsv_array[])
void Pvertex_color_array_rgb(int n, rgbFloat rgb_array[])
```

ARGUMENTS

**col*	The color to insert into the table
ind	An integer representing the table index
red,green,blue	The floating point values representing the red, green, blue
hue	The HSV color hue
sat	The HSV color saturation
value	The HSV color value (intensity)
n	An integer representing the number of items in an array
col_array	An array of colors to insert into the table
ind_array	An array of integers representing the table index
hsv_array	An array of hsv floats representing the HSV colors
rgb_array	An array of rgb floats representing the RGB colors

RETURNS

Nothing

DESCRIPTION

Pvertex_color sets a color defined by *col* to be attached to the next vertex defined by a **Ppoly_point**... command.
Pvertex_color_array defines an array of colors that are attached to each vertex of the next **Ppoly_**... command, the size of the array *n* should match the number of vertices in the polygon that follows.
These routines may be used to provide colors per vertex for any following **Ppoly_fill** or **Ppoly_shade** routines.

SEE ALSO

Ppoly_point..., Ppoly_...

NAME

Pvertex_normal - Enter a list of vertex normals into current geometry

SPECIFICATION

void Pvertex_normal(Fcoord nx, Fcoord ny, Fcoord nz)
void Pvertex_normals(int n, Fcoord3D normals[])

ARGUMENTS

nx, *ny*, *nz*	3 Fcoord's that define the vertex normal value
n	The number of vertex normals in the *norms* array
normals[]	Array of Fcoord3D's that define the vertex normal values

RETURNS

Nothing

DESCRIPTION

Pvertex_normal sets *nx, ny, nz* as the vertex normal for the current vertex, and **Pvertex_-normals** sets *n* normals using the array *normals* as the vertex normals for any following **Ppoly_shade** call. If geometry is being constructed with **Pbegin_line**, **Pbegin_poly**, or **Pbegin_tmesh**, then this routine will insert 1 (or *n)* normal(s) using the values *nx*, *ny*, *nz* (or the array *normals*) into the geometry which is currently being constructed.

SEE ALSO

Pbegin_poly, Pbegin_tmesh, Pend_poly, Pend_tmesh, Ppoly_shade, Pshade_mode

NAME

Pvertex_texture - Enter a list of texture vertices into current geometry

SPECIFICATION

void Pvertex_texture(Fcoord u, Fcoord v)
void Pvertex_textures(int n, Tvar texverts[])

ARGUMENTS

u,v	2 Fcoords that define the current texture vertex value
n	The number of texture vertices in the *tverts* array
texverts[]	Array of Tvar that define the texture vertex values

RETURNS

Nothing

DESCRIPTION

Pvertex_texture sets *u*, *v* as the texture values for the current vertex, and **Pvertex_textures** sets *n* texture vertices using the array *texverts* as the texture vertices for any following **Ppoly_shade** call. If geometry is being constructed with **Pbegin_line**, **Pbegin_poly**, or **Pbegin_tmesh**, then this routine will insert 1 (or *n*) texture vertice(s) using *u* and *v* (or the array *texverts*) into the geometry which is currently being constructed.

NOTE

Support for texture mapping is platform dependent.

SEE ALSO

Pbegin_poly, Pbegin_tmesh, Pend_poly, Pend_tmesh, Ppoly_shade, Pshade_mode

NAME

Pviewport - set up a viewport

SPECIFICATION

void Pviewport(Screencoord left, Screencoord right, Screencoord bottom, Screencoord top)

ARGUMENTS

left	The viewport left coordinate
right	The viewport right coordinate
bottom	The viewport bottom coordinate
top	The viewport top coordinate

RETURNS

Nothing

DESCRIPTION

Pviewport sets up a rectangle of the X window, in X window-relative coordinates, to hold the picture. The world coordinates defined by **Portho_view**, **Ppersp_view**, or **Pwindow_view** are mapped into this rectangle. The clip rectangle is also set by this call. If the viewport is not square then the world coordinates will be squeezed to fit, changing the aspect of the picture, this can be adjusted using the aspect parameter in **Ppersp_view**, or using a rectangle of similar dimensions in the **Portho_view** call.
The current viewport may be saved with **Ppush_viewport** and restored with **Ppop_viewport**.

NOTE

For this routine, the bottom left hand corner of the *X window* is (0,0), and the top right hand corner is (width, height).
This routine cannot be used inside a structure.

SEE ALSO

Pclip_rectangle, Portho_view, Ppersp_view, Ppop_viewport, Ppush_viewport

NAME

Pwindow_view - Define a 3D perspective viewing projection

SPECIFICATION

```
void Pwindow_view(int id, Fcoord left, Fcoord right, Fcoord bottom, Fcoord top,
        Fcoord near, Fcoord far)
```

ARGUMENTS

id	The view id number
left	The view left coordinate
right	The view right coordinate
bottom	The view bottom coordinate
top	The view top coordinate
near	The view near clipping plane
far	The view far clipping plane

RETURNS

Nothing

DESCRIPTION

Pwindow_view defines a viewing frustum from the supplied parameters. The values *left*, *right*, *bottom*, *top* are used to generate the field of view and aspect ratio, *near* and *far* set the near and far clipping planes.
The clipping planes are set to clip to a frustum that matches the field of view.
The view *id* is a reference number. If it is set to zero, then the viewing parameters are saved in the viewing table at the current viewing stack location for later reference with **Pset_view_index**. If *id* is set to a value from 1 to PEXtk_VIEW_SIZE, then the viewing parameters are saved in the viewing table at location *id*, and *id* may then be used to refer to this view with **Pset_view_index**. This is the recommended alternative technique to saving and restoring the view with **Ppush_matrix** and **Ppop_matrix**.

NOTE

This routine cannot be used inside a structure.
The only method to specify the viewing parameters for a view *id* of zero is to use the **Pset_view_index** routine in conjunction with this routine.
The resulting matrix is loaded into the view mapping matrix in the viewing table. A unity matrix will be loaded into the view orientation matrix by default.

SEE ALSO

Portho_2d_view, Portho_view, Ppersp_view

NAME

Pzbuffer - Set Z-buffer mode

SPECIFICATION

void Pzbuffer(int mode)

ARGUMENTS

mode	A flag to turn the zbuffer on or off.

RETURNS

Nothing

DESCRIPTION

Pzbuffer turns the zbuffer on or off. If *mode* is set to PEXtk_ZBUFFER_ON, then the zbuffer is turned on. If *mode* is set to PEXtk_ZBUFFER_OFF, then the zbuffer is turned off. The depth can be set with any of the view commands or with **Pdepth**.

NOTE

This routine cannot be used in a structure.

SEE ALSO

Pdepth