SunPHIGS 3.0 Extensions Reference Manual

Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043

Part No: 801-3609-10
Revision A, April 1993
SunPHIGS Release 3.0

NAME

INTRODUCTION TO SUNPHIGS EXTENSIONS – the introduction to the PHIGS PLUS Extensions in the SunPHIGS graphics library

DESCRIPTION

The SunPHIGS graphics library offers extensions to PHIGS, including many defined in the document: *Draft proposed PHIGS PLUS, ISO/IEC SC24-N454* (20 March 1990). That document describes extensions to the PHIGS standard to support additional output primitives and primitive attributes.

SunPHIGS Extensions contains a subset of the PHIGS PLUS functions. This subset includes support for lighting, shading, depth-cueing, and additional primitives.

These reference manual pages describe each non-PHIGS function in the SunPHIGS graphics library. Their organization is similar to the *SunPHIGS Reference Manual*. Each function is described separately and includes the C and FORTRAN syntax.

New Output Primitives

The graphical data produced by PHIGS are output primitives and attributes that are generated from the traversal of structure elements. SunPHIGS Extensions work the same way.

SunPHIGS Extensions augment the set of PHIGS output primitives by introducing a corresponding set of extended primitives. Some SunPHIGS Extension output primitive structure elements can have variant parameters, whose type can vary on an invocation by invocation basis. The amount and type of information specified is defined by a set of flag parameters for each invocation. Depending on the primitive being considered, the variant parameters are some combination of vertex normals, vertex colours, facet normals, facet colours, and edge flags. Special effects or higher quality results can be obtained through the use of this information. If the optional information is not provided by the application, defaults are calculated or derived from other attributes.

Non-uniform B-spline Curve

SunPHIGS generates a non-uniform B-spline curve of a user-specified order based on a list of knots in parameter space and a list of control points. Both rational and non-rational curves can be defined. A parameter range can be specified over which the independent variable is evaluated. All attributes that affect a polyline set 3 with data primitive also apply to this primitive.

Non-uniform B-spline Surface

SunPHIGS generates a non-uniform B-spline surface of two user-specified orders in two independent parameters based on a list of knots for each parameter and a grid of control points. Both rational and non-rational surfaces can be defined. A rectangular parameter range can be specified over which the independent variables are evaluated. An optional set of trimming curves (non-uniform B-spline curves) can be specified in the parameter space of the surface to limit the space over which the independent variables are evaluated and the B-spline surface is rendered. The edges of this primitive consist of the points on the surface corresponding to specified trimming curves. If no trimming curves are specified, then the edges consist of the curves of constant parameter at the

parameter bounds of the surface. All attributes that affect a fill area set 3 with data primitive apply to this primitive.

Fill Area Set 3 with Data

SunPHIGS generates a fill area set from a set of contours that can be empty, hollow, or filled with a colour distribution, a pattern, or a hatch style. The edge of a fill area set consists of the line segments that comprise its defining contours. The primitive definition can contain any combination of vertex colours, vertex normals, a facet normal, a facet colour, and edge visibility flags. Colours and normals can be used to light and shade the primitive. Edge visibility flags determine if individual edges are rendered. Ensuring that the points in all contours are coplanar is the responsibility of the application. The effect produced by non-planar fill area sets is implementation-dependent. All attributes that affect a fill area set primitive, plus additional attributes to control lighting and shading, apply to this primitive.

Polyline Set 3 with Data

SunPHIGS generates a set of unconnected polylines defined by a set of point sequences. Each point sequence in the set generates connected line segments. Colour information that may be used for shading the line segments can be specified with each point as part of the primitive definition. All attributes that affect a polyline primitive, plus additional attributes to control lighting and shading, apply to this primitive.

Quadrilateral Mesh 3 with Data

SunPHIGS generates a sequence of $(M-1)\times(N-1)$ quadrilaterals from a two-dimensional array of $M\times N$ vertices. A quadrilateral may be nonplanar, in which case the quadrilateral is rendered as two triangles. The method of decomposing the quadrilateral into triangles is implementation-dependent. The edges of this primitive consist of the line segments forming the boundary of each quadrilateral in the mesh. All attributes that affect the representation of fill area set 3 with data also affect the representation of this primitive. The primitive definition can contain any combination of vertex colours, vertex normals, facet colours, and facet normals. Colours and normals can be used to light and shade the primitive.

Set of fill Area Set 3 with Data (SOFAS)

SunPHIGS generates a group of facets, whose vertices are selected by indexing to a single list of vertices that is part of the primitive definition. The SOFAS primitive requires less data than a corresponding collection of fill area sets, if facets share common vertices. Only vertices and per-vertex data are shared among the facets, while edges and per-edge data are defined independently for each facet. All attributes that affect the representation of a fill area set 3 with data also affect the representation of this primitive. The primitive definition can contain any combination of vertex colours, vertex normals, facet colours, facet normals, and edge visibility flags. Colours and normals can be used to light and shade the primitive.

Triangle Strip 3 with Data

SunPHIGS generates a sequence of N-2 triangles from N vertices. The K'th triangle contains vertices K, K+1, and K+2. The edges of a triangle strip are the line segments forming the boundaries of all triangles in the strip. All attributes that affect the representation of a fill area set 3 with data also affect the representation of each triangle in this primitive. The primitive definition can contain any combination of vertex colours, vertex normals, facet colours, and facet normals. Colours and normals can be used to light and shade the primitive.

New Attributes and Extended Bundle Tables

General Colour

SunPHIGS Extensions support a general colour representation for direct and indirect colours. In this representation a colour consists of a colour type and a colour value. The colour type indicates whether the colour is direct or indirect. If the colour is indirect, then the colour value is an integer index to the workstation colour table. If the colour is direct, then the colour type also indicates the specific colour model, and the colour value is a triple of real numbers interpreted in the specified colour model As in PHIGS, four direct colour models are defined: RGB, CIE, HSV, and HLS.

All SunPHIGS Extensions colour-related structure elements and functions utilize the general specification of colours.

If a specified general colour has a nonsupported colour type, then the general colour defaults to colour type indirect and colour value one.

On true colour devices, direct colour is supported in hardware and indirect colour is mapped to direct colour by SunPHIGS. Cursor colours and PET colours may look different.

On 8-bit devices, the underlying hardware uses indirect colour and not direct colour. Direct colour can be simulated on these devices, but it may change the appearance and performance of PHIGS application, and is, therefore, controlled by the workstation type attribute PHIGS_COLOUR_MODE (see WORKSTATION TYPE SET (3P)).

If PHIGS_COLOUR_MODE is *index*, some SunPHIGS Extension features are not supported, or they have different functionality. General colour with type INDIRECT can be used. Direct colour is not supported and defaults to colour type INDIRECT, colour index 1. Lighting and depth-cueing are not supported. SunPHIGS Extension primitives with INDIRECT colour data may be shaded. This shading is not specified by PHIGS PLUS, but is a SunPHIGS Extension that linearly interpolates colour indices between the specified vertex colours. For example, if a line has colour index 3 on one vertex and colour index 7 on the other, then it will be shaded with colour indices 4, 5, and 6 in between. The appearance of these colour indices is controlled by the workstation's colour table, which can be set by the application (see SET COLOUR REPRESENTATION).

Enabling simulation has several effects on 8-bit devices. Direct colour is supported. All primitives are rendered using simulation, which means that

standard PHIGS primitives may have a different appearance. The rendering of primitives may be slower. The effect of setting a colour representation does not appear until the next traversal of the posted structures. (In the workstation description table the dynamic modification flag for SET COLOUR REPRESENTATION will be IRG.) Cursor colours and PET colours may look different.

The simulation approximates the colour specified by an application using a special colour map and dithering. The colour map uses 228 entries of the hardware colour map, so the mouse may need to be in the PHIGS graphical output area for the correct colours to appear.

Parametric Curve Attributes

Parametric curves are affected by the same attributes that affect polylines. In addition, the rendering accuracy of curves is controlled by the curve approximation criteria. An approximation criteria consists of an approximation type and approximation value. The approximation value is interpreted differently for each approximation type. The defined approximation types are listed in the manual entry for the function SET CURVE APPROXIMATION CRITERIA.

Parametric Surface Attributes

Parametric surfaces are affected by the same attributes that affect fill area sets. In addition, the rendering of surfaces is controlled by the surface approximation criteria. A surface approximation criteria consists of an approximation type and two approximation values. The approximation values are interpreted differently for each approximation type. The defined approximation types are listed in the manual entry for the SET SURFACE APPROXIMATION CRITERIA function.

Parametric surfaces are also affected by parametric surface characteristics. The parametric surface characteristics can be used to further specify the appearance of parametric surfaces beyond those aspects associated with fill area sets, such as isoparametric and level curves. The parametric surface characteristics aspect consists of a type and a data record. The content of the data record is unique for each type. The defined surface characteristics types are listed in the manual entry for the SET PARAMETRIC SURFACE CHARACTERISTICS function.

Trimming curves are treated as edges of a surface and they are rendered using edge attributes. In addition, the tessellation of trimming curves is controlled by the trimming curve approximation criteria, which is analogous to the curve approximation criteria, with the following exception; the approximation type can be only one of the following: implementation-dependent, workstation-dependent, constant isoparametric subdivision, or constant isoparametric subdivision between knots.

When the edge flag is OFF, no trimming curves are displayed. When the edge visibility flags are specified as part of the primitive definition and the edge flag is set to ON, the edge visibility flags determine whether the corresponding trimming curve is actually rendered.

New Functions

The following functions are implemented as SunPHIGS Extensions:

ADD TRIMMING CURVE†
add_trimming_curve (3PP)

CELL ARRAY 3 PLUS
cell_array_3_plus (3PP)

COMPUTE FILL AREA SET GEOMETRIC NORMAL
compute_fill_area_set_geometric_normal (3PP)

COPY ELEMENT RANGE
copy_element_range (3PP)

COPY ELEMENTS BETWEEN LABELS
copy_elements_between_labels (3PP)

FILL AREA SET 3 WITH DATA
fill_area_set_3_with_data (3PP)

INQUIRE COLOUR MAPPING FACILITIES
inquire_colour_mapping_facilities (3PP)

INQUIRE COLOUR MAPPING METHOD FACILITIES
inquire_colour_mapping_method_facilities (3PP)

INQUIRE COLOUR MAPPING REPRESENTATION
inquire_colour_mapping_representation (3PP)

INQUIRE CURVE AND SURFACE FACILITIES
inquire_curve_and_surface_facilities (3PP)

INQUIRE DEPTH CUE FACILITIES
inquire_depth_cue_facilities (3PP)

INQUIRE DEPTH CUE REPRESENTATION
inquire_depth_cue_representation (3PP)

INQUIRE DIRECT COLOUR MODEL FACILITIES
inquire_direct_colour_model_facilities (3PP)

INQUIRE DYNAMICS OF WORKSTATION ATTRIBUTES PLUS
inquire_dynamics_of_workstation_attributes_plus (3PP)

INQUIRE EDGE REPRESENTATION PLUS
inquire_edge_representation_plus (3PP)

INQUIRE INTERIOR FACILITIES PLUS
inquire_interior_facilities_plus (3PP)

INQUIRE INTERIOR REPRESENTATION PLUS
inquire_interior_representation_plus (3PP)

INQUIRE LIGHT SOURCE FACILITIES
inquire_light_source_facilities (3PP)

† FORTRAN only.

INQUIRE LIGHT SOURCE REPRESENTATION
inquire_light_source_representation (3PP)

INQUIRE LIST OF COLOUR MAPPING INDICES
inquire_list_of_colour_mapping_indices (3PP)

INQUIRE LIST OF DEPTH CUE INDICES
inquire_list_of_depth_cue_indices (3PP)

INQUIRE LIST OF LIGHT SOURCE INDICES
inquire_list_of_light_source_indices (3PP)

INQUIRE PATTERN REPRESENTATION PLUS
inquire_pattern_representation_plus (3PP)

INQUIRE POLYLINE FACILITIES PLUS
inquire_polyline_facilities_plus (3PP)

INQUIRE POLYLINE REPRESENTATION PLUS
inquire_polyline_representation_plus (3PP)

INQUIRE POLYMARKER REPRESENTATION PLUS
inquire_polymarker_representation_plus (3PP)

INQUIRE PREDEFINED COLOUR MAPPING REPRESENTATION
inquire_predefined_colour_mapping_representation (3PP)

INQUIRE PREDEFINED DEPTH CUE REPRESENTATION
inquire_predefined_depth_cue_representation (3PP)

INQUIRE PREDEFINED EDGE REPRESENTATION PLUS
inquire_predefined_edge_representation_plus (3PP)

INQUIRE PREDEFINED INTERIOR REPRESENTATION PLUS
inquire_predefined_interior_representation_plus (3PP)

INQUIRE PREDEFINED LIGHT SOURCE REPRESENTATION
inquire_predefined_light_source_representation (3PP)

INQUIRE PREDEFINED PATTERN REPRESENTATION PLUS
inquire_predefined_pattern_representation_plus (3PP)

INQUIRE PREDEFINED POLYLINE REPRESENTATION PLUS
inquire_predefined_polyline_representation_plus (3PP)

INQUIRE PREDEFINED POLYMARKER REPRESENTATION PLUS
inquire_predefined_polymarker_representation_plus (3PP)

INQUIRE PREDEFINED TEXT REPRESENTATION PLUS
inquire_predefined_text_representation_plus (3PP)

INQUIRE RENDERING COLOUR MODEL FACILITIES
inquire_rendering_colour_model_facilities (3PP)

INQUIRE TEXT REPRESENTATION PLUS
inquire_text_representation_plus (3PP)

INQUIRE WORKSTATION STATE TABLE LENGTHS PLUS
inquire_workstation_state_table_lengths_plus (3PP)

NON-UNIFORM B-SPLINE CURVE
non-uniform_b-spline_curve (3PP)

NON-UNIFORM B-SPLINE SURFACE
non-uniform_b-spline_surface (3PP)

OPEN XPHIGS
open_xphigs (3PP)

POLYLINE SET 3 WITH DATA
polyline_set_3_with_data (3PP)

QUADRILATERAL MESH 3 WITH DATA
quadrilateral_mesh_3_with_data (3PP)

SET BACK INTERIOR COLOUR
set_back_interior_colour (3PP)

SET BACK INTERIOR REFLECTANCE EQUATION
set_back_interior_reflectance_equation (3PP)

SET BACK INTERIOR SHADING METHOD
set_back_interior_shading_method (3PP)

SET BACK INTERIOR STYLE
set_back_interior_style (3PP)

SET BACK INTERIOR STYLE INDEX
set_back_interior_style_index (3PP)

SET BACK REFLECTANCE PROPERTIES
set_back_reflectance_properties (3PP)

SET COLOUR MAPPING INDEX
set_colour_mapping_index (3PP)

SET COLOUR MAPPING REPRESENTATION
set_colour_mapping_representation (3PP)

SET CURVE APPROXIMATION CRITERIA
set_curve_approximation_criteria (3PP)

SET DEPTH CUE INDEX
set_depth_cue_index (3PP)

SET DEPTH CUE REPRESENTATION
set_depth_cue_representation (3PP)

SET EDGE COLOUR
set_edge_colour (3PP)

SET EDGE REPRESENTATION PLUS
set_edge_representation_plus (3PP)

SET ELEMENT POINTER AT PICK IDENTIFIER
set_element_pointer_at_pick_identifier (3PP)

SET FACE CULLING MODE
set_face_culling_mode (3PP)

SET FACE DISTINGUISHING MODE
set_face_distinguishing_mode (3PP)

SET INTERIOR COLOUR
set_interior_colour (3PP)

SET INTERIOR REPRESENTATION PLUS
set_interior_representation_plus (3PP)

SET INTERIOR SHADING METHOD
(set_interior_shading_method3PP)

SET LIGHT SOURCE REPRESENTATION
set_light_source_representation (3PP)

SET LIGHT SOURCE STATE
set_light_source_state (3PP)

SET OF FILL AREA SET 3 WITH DATA
set_of_fill_area_set_3_with_data (3PP)

SET PARAMETRIC SURFACE CHARACTERISTICS
set_parametric_surface_characteristics (3PP)

SET PATTERN REPRESENTATION PLUS
set_pattern_representation_plus (3PP)

SET POLYLINE COLOUR
set_polyline_colour (3PP)

SET POLYLINE REPRESENTATION PLUS
set_polyline_representation_plus (3PP)

SET POLYLINE SHADING METHOD
set_polyline_shading_method (3PP)

SET POLYMARKER COLOUR
set_polymarker_colour (3PP)

SET POLYMARKER REPRESENTATION PLUS
set_polymarker_representation_plus (3PP)

SET REFLECTANCE EQUATION
set_reflectance_equation (3PP)

SET REFLECTANCE PROPERTIES
set_reflectance_properties (3PP)

SET SURFACE APPROXIMATION CRITERIA
set_surface_approximation_criteria (3PP)

SET TEXT COLOUR
set_text_colour (3PP)

SET TEXT REPRESENTATION PLUS
set_text_representation_plus (3PP)

TRIANGLE STRIP 3 WITH DATA
triangle_strip_3_with_data (3PP)

Changed Functions

The following functions have been extended to incorporate SunPHIGS Extensions primitives and attributes. The new versions are fully compatible with the standard PHIGS versions. The updated manual pages for these functions are included in the *SunPHIGS Reference Manual*.

ELEMENT SEARCH
INQUIRE CURRENT ELEMENT CONTENT
INQUIRE CURRENT ELEMENT TYPE AND SIZE
INQUIRE ELEMENT CONTENT
INQUIRE ELEMENT TYPE AND SIZE
SET INDIVIDUAL ASF
WORKSTATION TYPE SET

Limitations

Maximum B-spline Order

The maximum B-spline order supported by any SunPHIGS workstation type is ten. The maximum trimming curve order supported by any SunPHIGS workstation type is six. Primitives with greater orders are displayed as described in the manual pages for the functions NON-UNIFORM B-SPLINE CURVE and NON-UNIFORM B-SPLINE SURFACE.

Supported Curve and Surface Approximation Types

SunPHIGS Extensions supports the following approximation types: workstation-dependent and constant parametric subdivision between knots for B-spline curves, B-spline surfaces, and trimming curves. Specification of other types map to the workstation-dependent type during structure traversal. The workstation-dependent type is constant parametric subdivision between knots for all SunPHIGS workstation types.

In addition to the above non-adaptive curve and surface approximation types, SunPHIGS Extensions supports an adaptive approximation type for parametric curve and surfaces. Use of this adaptive type adjusts the displayed representation of parametric curve and surface primitives according to modelling and viewing transformations. As the display space used to view the primitive is increased, as when zooming in, the curve or surface is more finely tessellated. As the the display space used to view the primitive is reduced, the primitive is more coarsely tessellated. This helps the application maintain the smooth appearance of a curve or surface primitive as it is displayed with different modelling and viewing transforms.

This adaptive approximation type is applied to parametric curves when the

current curve approximation type is either Metric in NPC or Chordal Deviation in NPC. It is applied to parametric surfaces when the current surface approximation type is either Metric in NPC or Planar Deviation in NPC. The approximation value specifies the maximum size, as a percentage of the workstation viewport, of each tessellated portion of a parametric curve or surface primitive. The primitive is also tessellated at positions corresponding to the knot values to eliminate distortion of the general shape and other features of a curve or surface. Among other things, this ensures that any specified discontinuities are reflected in the displayed representation.

Major changes to the PHIGS PLUS curve and surface approximation types are currently under consideration by the ANSI and ISO PHIGS PLUS committees. Future releases of SunPHIGS will include those changes, which may require changes to, or removal of, the currently supported approximation types.

CGM Support on PHIGS PLUS Output Primitives and Attributes:

During generation of CGM, SunPHIGS ignores vertex normals, vertex colours, facet normals, and facet colours, but recognizes vertex coordinates and edge flags of the primitives. CGM workstations also ignore attributes that control lighting, shading, and depth-cueing.

Binding Notes

The SunPHIGS Extensions C and FORTRAN bindings follow the conventions used in the International Standard PHIGS bindings. Since PHIGS PLUS is not yet standardized, there are no standard bindings for PHIGS PLUS. When the bindings become available, SunPHIGS Extensions may replace the bindings listed here with new ones. The syntax descriptions of the functions listed in this reference manual are subject to change.

SEE ALSO

COLOUR (7P)
PHIGS TRAVERSAL STATE LIST (7P)
PHIGS WORKSTATION DESCRIPTION TABLE (7P)
INTRO (3P)
INTRO PHIGS (3P)
INTRO (7P)

NAME

ADD TRIMMING CURVE – add trimming curve parameters to the trim curve parameter lists

SYNOPSIS

C Syntax

There is no C binding for this function.

FORTRAN Syntax

SUBROUTINE paddtc (NEWLP, CVIS, CORDER, CRAT, CNKNTS, CKNOTS, CNCPTS, CCPTS, CTMIN, CTMAX, NLOOPS, LOOPS, VIS, ORDER, TRATS, NTKNTS, TKSTRT, TKNOTS, NTCPTS, TCSTRT, TCPTS, TMIN, TMAX, ERRIND)

INTEGER	NEWLP	*new loop flag*
INTEGER	CVIS	*trim curve visibility*
INTEGER	CORDER	*trim curve order*
INTEGER	CRAT	*rationality of trim curve*
INTEGER	CNKNTS	*number of knots for trim curve*
REAL	CKNOTS(CNKNTS)	*trim curve knots*
INTEGER	CNCPTS	*number of control points for trim curve*
REAL	CCPTS(3,CNCPTS)	*control points*
REAL	CTMIN	*lower parameter limit for trim curve*
REAL	CTMAX	*upper parameter limit for trim curve*
INTEGER	NLOOPS	*OUT number of trim loops†*
INTEGER	LOOPS(*)	*OUT number of trim curves in each loop†*
INTEGER	VIS(nt)	*OUT visibility flags for all trim curves‡*
INTEGER	ORDER(nt)	*OUT curve orders for all trim curves‡*
INTEGER	TRATS(nt)	*OUT curve types for all trim curves‡*
INTEGER	NTKNTS(nt)	*OUT number of knots in each curve‡*
INTEGER	TKSTRT(nt)	*OUT start indices of knot lists in TKNOTS‡*
REAL	TKNOTS(*)	*OUT knots, for all trim curves‡*
INTEGER	NTCPTS(nt)	*OUT number of control points in each curve‡*
INTEGER	TCSTRT(nt)	*OUT start indices of control point lists in TCPTS‡*
REAL	TCPTS(3,*)	*OUT control points, for all trim curves‡*
REAL	TMIN(nt), TMAX(nt)	*OUT parameter limits for all trim curves‡*
INTEGER	ERRIND	*OUT error indicator‡*

† The parameters below are exactly as passed to pnubss.

‡ In the array dimensions below, "nt" means the total number of trim curves.

Required PHIGS Operating States

(*, *, *, *)

DESCRIPTION

Purpose

ADD TRIMMING CURVE builds the lists of trimming curve parameters to be passed to NON-UNIFORM B-SPLINE SURFACE in FORTRAN. It is called once for each trimming curve. The output arrays must be preallocated to the required size or larger.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

NEWLP New loop flag. One means this trimming curve starts a new loop. Zero means this trimming curve is a continuation of the current loop.

CVIS Visibility flag for this trimming curve.

CORDER
Order of this trimming curve.

CRAT Rationality of this trimming curve. Valid values are:

PRAT *Rational*
PNRAT *Non-rational*

CNKNTS
Number of knots for this trimming curve.

CKNOTS(CNKNTS)
Knots for this trimming curve.

CNCPTS
Number of control points for this trimming curve.

CCPTS(3,CNCPTS)
Control points for this trimming curve. The point coordinates are stored in each column of the array as:

CCPTS(1,t) = *u Coordinate*
CCPTS(2,t) = *v Coordinate*
CCPTS(3,t) = *w Coordinate*

The *w* coordinate is not used if the curve is *Non-rational*.

CTMIN Lower parameter limit for this trimming curve.

CTMAX Upper parameter limit for this trimming curve.

FORTRAN Output Parameters

NLOOPS
Number of trimming curve loops. This variable must be initialized to zero prior to adding the first trimming curve and should subsequently be changed only by this function.

LOOPS(*)
Number of trimming curves in each loop.

VIS(nt) Array of visibility flags, one entry for each trimming curve.

ORDER(nt)
Array of curve orders, one entry for each trimming curve.

TRATS(nt)
Array of trimming curve rationalities, one entry for each trimming curve. Each entry is an integer specifying the rationality and hence the dimensionality of the

control points.

NTKNTS(nt)
Number of knots in each curve, one entry for each trimming curve.

TKSTRT(nt)
Start positions of the knot lists in the array *TKNOTS*, one entry for each trimming curve.

TKNOTS(*)
Array of knots for all trimming curves. Each curve uses a contiguous segment of the array. The index of the head of the array segment used by a curve is specified in the *TKSTRT* array.

NTCPTS(nt)
Number of control points in each curve, one entry for each trimming curve.

TCSTRT(nt)
Start positions of the control point lists in the array *TCPTS* , one entry for each trimming curve.

*TCPTS(3, *)*
Array of control points for all trimming curves. Each curve uses a contiguous segment of the array. The index of the head of the array segment used by a curve is specified in the *TCSTRT* array.

TMIN(nt), TMAX(nt)
Arrays of parameter limits for the trimming curves, one entry in each array for each trimming curve.

ERRIND
The error number of any error detected by this function.

Execution

ADD TRIMMING CURVE is a utility function to build the lists of trimming curve parameters passed to NON-UNIFORM B-SPLINE SURFACE in FORTRAN. ADD TRIMMING CURVE is called once per trimming curve. It adds the curve parameters to the appropriate places in the lists and increments the curve and loop counts as required. After this function has been called for all trimming curves the output parameter lists can be passed directly to NON-UNIFORM B-SPLINE SURFACE. See NON-UNIFORM B-SPLINE SURFACE for a complete description of all the parameters.

The output arrays must be declared or allocated large enough to contain all the trim curve data. No bounds checking is done on the length of these arrays and no memory allocation is performed by this function.

ERRORS

None

SEE ALSO

NON-UNIFORM B-SPLINE CURVE (3PP)
INQUIRE CURVE AND SURFACE FACILITIES (3PP)
INTRO (3PP)

NAME

CELL ARRAY 3 PLUS – create a 3D extended cell array element

SYNOPSIS
C Syntax

void
pcell_array3_plus (paral, dim, colr_type, colrs)

Pparal	*paral;	*cell parallelogram: [0]=P; [1]=Q; [2]=R*
Pint_size	*dim;	*dimension of cell: DX and DY*
Pint	colr_type;	*colour type*
Pcoval	*colrs;	*colour array*

SYNOPSIS
FORTRAN Syntax

SUBROUTINE pca3p (COLTYP, CPXA, CPYA, CPZA, DIMX, DIMY, ISC, ISR, DX, DY, COLIA, COLA)

INTEGER	COLTYPE	*colour type*
REAL	CPXA(3), CPYA(3), CPZA(3)	*cell parallelogram (P, Q, R) (MC)*
INTEGER	DIMX, DIMY	*dimensions of COLIA containing the cell array*
INTEGER	ISC, ISR	*indices of start column, start row*
INTEGER	DX, DY	*number of columns, number of rows*
INTEGER	COLIA(DIMX, DIMY)	*colour index array*
REAL	COLA(3,DIMX*DIMY)	*direct colour array*

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION
Purpose

CELL ARRAY 3 PLUS creates a 3D cell array made of cells filled with general colour in each cell. Its area is defined by three points, P, Q, and R, in Modelling Coordinates giving the corners of a parallelogram as P, Q, R and (Q+R-P).

C Input Parameters

paral A pointer to an array of three Ppoint3 structures giving three corners of the parallelogram that defines the area of the CELL ARRAY 3 PLUS element. Pparal is defined in phigs.h as follows:

```
typedef struct {
        Ppoint3     p;      /* point p */
        Ppoint3     q;      /* point q */
        Ppoint3     r;      /* point r */
} Pparal;
```

colr_type

The colour type for specified cell colours.

0	PINDIRECT	*Colour Index Specified*
1	PMODEL_RGB	*Red, Green, Blue Colour Model*

	2	PMODEL_CIELUV	*Commission Internationale de l'Eclairage Colour Model*
	3	PMODEL_HSV	*Hue, Saturation, Value Colour Model*
	4	PMODEL_HLS	*Hue, Lightness, Saturation Colour Model*

C Output Parameters

dim A pointer to a Pdim structure specifying the dimensions of the colour cells array. Pint_size is defined in phigs.h as follows:

```
typedef struct {
        Pint    size_x;  /* dimension (number of divisions) along X */
        Pint    size_y;  /* dimension (number of divisions) along Y */
} Pint_size;
```

colrs A pointer to an array of Pcoval of dimension defined by *dim*, containing the colour values specifying cell colours. Pcoval is defined in phigs.h as follows:

```
typedef union {
        Pint        ind;        /* index in workstation colour
                                   bundle table */
        Pcolr_rep   direct;     /* direct colour components */
} Pcoval;
```

Pcolr_rep is defined as:

```
typedef union {
        Prgb        rgb;        /* Red, Green, Blue colour specification */
        Pcieluv     cieluv;     /* CIE L*U*V*
                                   colour specification */
        Phls        hls;        /* Hue, Lightness, Saturation colour
                                   specification */
        Phsv        hsv;        /* Hue, Saturation, Value colour
                                   specification */
        Pdata       unsupp;     /* Colour in unsupported colour
                                   model */
} Pcolr_rep
```

Prgb is defined in phigs.h as follows:

```
typedef struct {
        Pfloat      red;        /* red, hue, and so on */
        Pfloat      green;      /* green, saturation, lightness, and so on */
        Pfloat      blue;       /* blue, value, saturation, and so on */
} Prgb;
```

Pcieluv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          cieluv_x;       /* x coefficient */
        Pfloat          cieluv_y;       /* y coefficient */
```

```
        Pfloat          cieluv_y_lum;   /* y luminance */
} Pcieluv;
```

Phsv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          satur;          /* saturation */
        Pfloat          value;          /* value */
} Phsv;
```

Phls is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          lightness;      /* lightness */
        Pfloat          satur;          /* saturation */
} Phls;
```

Pdata is defined in phigs.h as follows:

```
typedef struct {
        size_t          size;           /* size of data */
        char            *data           /* pointer to data */
} Pdata;
```

FORTRAN Input Parameters

COLTYPE

The colour type, one of the following values:

0	PINDIR	*Colour Index Specified*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

CPXA The *x* coordinates of the three corners of the parallelogram that define the area of the CELL ARRAY 3 element.

CPYA The *y* coordinates of the three corners of the parallelogram that define the area of the CELL ARRAY 3 element.

CPZA The *z* coordinates of the three corners of the parallelogram that define the area of the CELL ARRAY 3 element.

DIMX The *x* dimension, or number of columns, of the entire COLIA array.

DIMY The *y* dimension, or number of rows, of the entire COLIA array.

ISC The application may specify a portion of the COLIA array by indicating a starting position and number of rows and columns of the subarray. ISC is the *x* coordinate, or start column, of the subarray. To pass the entire array, this value should be 1.

ISR The *y* coordinate, or start row, of the subarray. To pass the entire array, this

value should be 1.

DX The *x* dimension, or number of columns, of the subarray. To pass the entire array, this value should be the same as that of DIMX.

DY The *y* dimension, or number of rows, of the subarray. To pass the entire array, this value should be the same as that of DIMY.

COLIA(DIMX, DIMY)
An array of integers containing the colour indices specifying colours for each cell. This array is only used when COLTYP is given as PINDIR, and is ignored otherwise.

*COLA(3, DIMX*DIMY)*
A two-dimensional array containing the direct colours for each cell. This array is used if COLTYP is PRGB, PCIE, PHSV, PHLS, and is ignored if COLTYP is PINDIR.

COLA(1,v) = Red, Hue, and so on
COLA(2,v) = Green, Saturation, Lightness, and so on
COLA(3,v) = Blue, Saturation, and so on

Execution

When the structure is traversed, the CELL ARRAY 3 PLUS element draws the transformed boundaries of the cell parallelogram, using the polyline attributes currently in effect. This is the minimal simulation for CELL ARRAY 3 PLUS.

Attributes Applied

The attributes listed below are used to display the CELL ARRAY 3 PLUS primitive when the structure is traversed. The Aspect Source Flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

polyline colour	polyline colour index ASF
linewidth scale factor	linewidth scale factor ASF
linetype	linetype ASF
polyline shading method	polyline shading method ASF
polyline index	
depth cue index	
name set	

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

117 Ignoring function, one of the dimensions of the colour index array is less than zero

SEE ALSO

CELL ARRAY 3 (3P)

NAME

COMPUTE FILL AREA SET GEOMETRIC NORMAL – compute geometric normal of the fill area set

SYNOPSIS

C Syntax

void
pcomp_fill_area_set_gnorm (vflag, nfa, vdata, error_ind, normal)

Pint	vflag;	*what vertex data is available*
Pint	nfa;	*number of fill areas*
Pfacet_vdata_list3	*vdata;	*fill area data*
Pint	*error_ind;	*OUT error indicator*
Pvec3	*normal;	*OUT geometric normal*

FORTRAN Syntax

SUBROUTINE pcfasn (NFA, FARR, NV, COORDS, ERRIND, GNORM)

INTEGER	NFA	*number of fill areas*
INTEGER	FARR(NFA)	*array of end indices per facet*
INTEGER	NV	*number of vertices*
REAL	COORDS(3,NV)	*array of vertex coordinates*
INTEGER	ERRIND	*OUT error indicator*
REAL	GNORM(3)	*OUT geometric normal*

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

COMPUTE FILL AREA SET GEOMETRIC NORMAL computes the geometric normal of the fill area set.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

All of the following data types are predefined in phigs.h.

vflag The data per vertex flag specifies the available data for each vertex of the primitive.

0	PVERT_COORD	*Coordinates Specified*
1	PVERT_COORD_COLOUR	*Coordinates and Vertex Colour Specified*
2	PVERT_COORD_NORMAL	*Coordinates and Vertex Normal Specified*
3	PVERT_COORD_COLOUR_NORMAL	*Coordinates, Vertex Colour, and Vertex Normal Specified*

nfa The number of fill areas in the set.

vdata A pointer, to a list of nfa Pfacet_vdata_list3 structures, that specifies the fill area set and optionally associated colour information. Pfacet_vdata_list3 is defined as

follows:

```
typedef struct {
        Pint                num_vertices;   /* number of vertices */
        Pfacet_vdata_arr3   vertex_data;    /* array of facet vertex data */
} Pfacet_vdata_list3;
```

Pfacet_vdata_arr3 is defined as follows:

```
typedef union {
        Ppoint3         *points;        /* point */
        Pptco3          *ptcolrs;       /* point and color */
        Pptnorm3        *ptnorms;       /* point and normal */
        Pptconorm3      *ptconorms;     /* point, color, and normal */
        /* implementation-dependent types can go here */
} Pfacet_vdata_arr3;
```

Ppoint3 is defined as follows:

```
typedef struct {
        Pfloat    x;    /* x coordinate */
        Pfloat    y;    /* y coordinate */
        Pfloat    z;    /* z coordinate */
} Ppoint3;
```

Pptco3 is defined as follows:

```
typedef struct {
        Ppoint3    point;    /* point coordinates */
        Pcoval     colr;     /* colour */
} Pptco3;
```

Ppoint3 is defined above. Pcoval is defined as follows:

```
typedef union {
        Pint          ind;          /* index in workstation colour
                                       bundle table */
        Pcolr_rep     direct;       /* direct colour components */
} Pcoval;
```

Pcolr_rep is defined as follows:

```
typedef union {
        Prgb          rgb;          /* Red, Green, Blue colour specification */
        Pcieluv       cieluv;       /* CIE L*U*V* colour specification */
        Phls          hls;          /* Hue, Lightness, Saturation colour
                                       specification */
        Phsv          hsv;          /* Hue, Saturation, Value colour
                                       specification */
        Pdata         unsupp;       /* Colour in unsupported colour model */
```

```
} Pcolr_rep;
```

Prgb is defined in phigs.h as follows:

```
typedef struct {
        Pfloat      red;        /* red, hue, and so on */
        Pfloat      green;      /* green, saturation, lightness, and so on */
        Pfloat      blue;       /* blue, value, saturation, and so on */
} Prgb;
```

Pcieluv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          cieluv_x;       /* x coefficient */
        Pfloat          cieluv_y;       /* y coefficient */
        Pfloat          cieluv_y_lum;   /* y luminance */
} Pcieluv;
```

Phsv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          satur;          /* saturation */
        Pfloat          value;          /* value */
} Phsv;
```

Phls is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          lightness;      /* lightness */
        Pfloat          satur;          /* saturation */
} Phls;
```

Pdata is defined in phigs.h as follows:

```
typedef struct {
        size_t          size;           /* size of data */
        char            *data           /* pointer to data */
} Pdata;
```

Pptnorm3 is defined as follows:

```
typedef struct {
        Ppoint3 point;      /* point coordinates */
        Pvec3   normal;     /* normal */
} Pptnorm3;
```

Ppoint3 is defined above. Pvec3 is defined as follows:

```
typedef struct {
        Pfloat          delta_x;        /* x magnitude */
        Pfloat          delta_y;        /* y magnitude */
```

```
        Pfloat      delta_z;    /* z magnitude */
    } Pvec3;
```

Pptconorm3 is defined as follows:

```
    typedef struct {
        Ppoint3     point;      /* point coordinates */
        Pcoval      colr;       /* colour */
        Pvec3       norm;       /* normal */
    } Pptconorm3;
```

Ppoint3, Pcoval, and Pvec3 are defined above.

C Output Parameters

error_ind
A pointer to the location to store the error number.

normal A pointer to the location to store the computed geometric normal. (Pvec3 is defined above.)

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

NFA The number of fill areas in the fill area set.

FARR(NFA)
An array of integers containing the end indices in the COORDS array for each point list in the fill area set.

NV The number of points used to define the fill area set.

COORDS(3,NV)
An array of reals containing the *x*, *y*, and *z* coordinates of the fill area set.

COORDS(1,v) = *x Coordinate*
COORDS(2,v) = *y Coordinate*
COORDS(3,v) = *z Coordinate*

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

GNORM(3)
An array of reals containing the computed geometric normal.

GNORM(1) = *x Magnitude*
GNORM(2) = *y Magnitude*
GNORM(3) = *z Magnitude*

Execution

The geometric normal to the fill area set defined by the vertex coordinates is returned in the unit normal vector. The geometric normal is computed by first selecting three points: *A*, *B*, and *C*. Point *A* is the first point in the first list of vertices. Point *B* is the next point in that list that is noncoincident with *A*. Point *C* is the next point in that list, after *B*, that is non-colinear with *A* and *B*. The geometric normal is the cross product of the vector extending from *A* to *B* with the vector extending from *A* to *C*.

If it is not possible to find three such points in the first list of vertices, then the rest of the lists are searched in order to select three appropriate points from a single list. In case the search fails in all lists, then it is determined that the fill area set is degenerate.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)

612 Warning, the fill area is degenerate

SEE ALSO

FILL AREA SET 3 WITH DATA (3PP)
INTRO (3PP)

NAME

COPY ELEMENT RANGE – copy a block of elements from the source structure to the open structure

SYNOPSIS

C Syntax

```
void
pcopy_elem_range ( struct_id, ep1_value, ep2_value )
Pint    struct_id;      source structure identifier
Pint    ep1_value;      element pointer 1 value
Pint    ep2_value;      element pointer 2 value
```

FORTRAN Syntax

```
SUBROUTINE pcelra ( STRID, EP1, EP2 )
INTEGER    STRID        source structure identifier
INTEGER    EP1, EP2     element pointer range
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

COPY ELEMENT RANGE copies all structure elements in the source structure between two specified element numbers, inclusive, to the open structure.

C Input Parameters

struct_id
Specifies the structure from which elements will be copied.

ep1_value
Specifies the beginning of the element range to be copied.

ep2_value
Specifies the end of the element range to be copied.

FORTRAN Input Parameters

STRID Specifies the structure from which elements will be copied.

EP1 Specifies the beginning of the element range to be copied.

EP2 Specifies the end of the element range to be copied.

Execution

The COPY ELEMENT RANGE subroutine copies structure elements from the source structure between and including the lower and the higher of the element positions specified by the element pointer 1 value and element pointer 2 value into the open structure. The elements are copied into the open structure following the element pointed to by the element pointer. The element pointer is updated to point to the element last inserted. The current edit mode has no effect on this function.

When an element position is less than 1, the range of elements copied starts from element position 0. When an element position is greater than the number of elements in the open structure, the range of elements copied ends at the last element of the source structure.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

COPY ALL ELEMENTS FROM STRUCTURE (3P)
COPY ELEMENTS BETWEEN LABELS (3PP)

NAME

COPY ELEMENTS BETWEEN LABELS – copy all elements in the source structure between specified labels to the open structure

SYNOPSIS

C Syntax

void pcopy_elems_labels (struct_id, label1_id, label2_id)

```
Pint    struct_id;    source structure identifier
Pint    label1_id;    label 1 identifier
Pint    label2_id;    label 2 identifier
```

FORTRAN Syntax

SUBROUTINE pcellb (STRID, LABEL1, LABEL2)

```
INTEGER    STRID              source structure identifier
INTEGER    LABEL1, LABEL2     label range
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

COPY ELEMENTS BETWEEN LABELS copies structure elements in the source structure between two specified labels to the open structure. The elements containing the labels are not copied.

C Input Parameters

struct_id
The subroutine copies elements from this structure.

label1_id
The subroutine begins copying elements with the structure element immediately following this label.

label2_id
The subroutine ends copying elements with the structure element immediately preceding this label.

FORTRAN Input Parameters

STRID The subroutine copies elements from this structure.

LABEL1
The subroutine begins copying elements with the structure element immediately following this label.

LABEL2
The subroutine ends copying elements with the structure element immediately preceding this label.

Execution

The COPY ELEMENTS BETWEEN LABELS subroutine copies all the elements in the source structure between the two specified labels into the open structure. The elements are copied into the open structure following the element pointed to by the element pointer. The element pointer is updated to point to the element last inserted. The current edit mode has no effect on this function.

If the source structure identifier is the open structure, the subroutine searches for the specified labels from the current position of the element pointer in the structure to the end of the structure. The next occurrence of an element containing label 1 identifier is first found; then, starting from the element containing label 1 identifier, the next occurrence of an element containing label 2 identifier is found.

If the source structure identifier is not the open structure, the subroutine searches for the specified labels from the first element in the structure to end of the structure. The first occurrence of an element containing label 1 identifier is first found; then, starting from the element containing label 1 identifier, the next occurrence of an element containing label 2 identifier is found.

If either of the label identifiers cannot be found, no elements are copied and an error is generated.

ERRORS

005	Ignoring function, function requires state (PHOP, *, STOP, *)
206	Ignoring function, one or both of the labels does not exist in the open structure between the element pointer and the end of the structure
-176	Ignoring function, the starting label does not exist in the structure
-177	Ignoring function, the label does not exist in the structure between the starting label and the end of the structure

SEE ALSO

COPY ALL ELEMENTS FROM STRUCTURE (3P)
COPY ELEMENT RANGE (3PP)

NAME

FILL AREA SET 3 WITH DATA – creates a 3D fill area set structure element that includes colour and shading data

SYNOPSIS

C Syntax

void
pfill_area_set3_data (fflag, eflag, vflag, colr_type, fdata, nfa, edata, vdata)

Pint	fflag;	*what data is specified per facet*
Pint	eflag;	*what data is specified per edge*
Pint	vflag;	*what data is specified per vertex*
Pint	colr_type;	*colour type*
Pfacet_data3	*fdata;	*facet data*
Pint	nfa;	*number of fill areas in the set*
Pedge_data_list	*edata;	*edge data list*
Pfacet_vdata_list3	*vdata;	*facet vertex data list*

FORTRAN Syntax

SUBROUTINE pfasd3 (FFLAG, EFLAG, VFLAG, COLTYP, IFCOLR, FCOLR, FNORM, NFA, FARR, NV, EDARR, COORDS, IVCOLR, VCOLR, VNORM, IMPL)

INTEGER	FFLAG	*data per facet flag*
INTEGER	EFLAG	*data per edge flag*
INTEGER	VFLAG	*data per vertex flag*
INTEGER	COLTYP	*colour type*
INTEGER	IFCOLR	*indirect facet colour*
REAL	FCOLR(3)	*array of facet colour*
REAL	FNORM(3)	*array of facet normals*
INTEGER	NFA	*number of fill areas*
INTEGER	FARR(NFA)	*array of fill area end indices*
INTEGER	NV	*number of vertices*
INTEGER	EDARR(NV)	*array of edge data (PON/POFF)*
REAL	COORDS(3,NV)	*array of vertex coordinates*
INTEGER	IVCOLR(NV)	*array of indirect vertex colours*
REAL	VCOLR(3,NV)	*array of vertex colours*
REAL	VNORM(3,NV)	*array of vertex normals*
REAL	IMPL(*)	*implementation-defined vertex data*

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

FILL AREA SET 3 WITH DATA creates a 3D fill area set primitive with colour and shading data.

The FILL AREA SET 3 WITH DATA is a set of closed polygonal areas defined by a series of three-dimensional Modelling Coordinate points. You can specify a geometric normal for the entire fill area, and specify colour / normal information for each vertex *in addition* to point coordinates. The added data is used when applying lighting, shading, face

distinguishing, and face culling.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

fflag The data per facet flag specifies the available data for the entire fill area set.

0	PFACET_NONE	*No Facet Data Specified*
1	PFACET_COLOUR	*Facet Colours Specified*
2	PFACET_NORMAL	*Facet Normal Specified*
3	PFACET_COLOUR_NORMAL	*Facet Normal and Colours Specified*

eflag The data per edge flag specifies the available information for each edge.

0	PEDGE_NONE	*No Edge Data Specified*
1	PEDGE_VISIBILITY	*Edge Visibility Flags Specified*

vflag The data per vertex flag specifies the available data for each vertex of the primitive.

0	PVERT_COORD	*Coordinates Specified*
1	PVERT_COORD_COLOUR	*Coordinates and Vertex Colour Specified*
2	PVERT_COORD_NORMAL	*Coordinates and Vertex Normal Specified*
3	PVERT_COORD_COLOUR_NORMAL	*Coordinates, Vertex Colour, and Vertex Normal Specified*

colr_type

The colour type for specified facet and / or vertex colours.

0	PINDIRECT	*Colour Index Specified*
1	PMODEL_RGB	*Red, Green, and Blue*
2	PMODEL_CIELUV	*CIE Colour Model*
3	PMODEL_HSV	*Hue, Saturation, and Value*
4	PMODEL_HLS	*Hue, Lightness, and Saturation*

fdata A pointer to a Pfacet_data3 structure that specifies the fill area and optionally associated colour information. Pfacet_data3 is defined as follows:

```
typedef union {
        Pcoval          colr;           /* colour */
        Pvec3           normal;         /* unit normal*/
        Pconorm3        conorm;         /* colour and normal */
} Pfacet_data3;
```

coval is defined as:

```
typedef union {
        Pint        ind;        /* index in workstation colour
                                   bundle table */
        Pcolr_rep   direct;     /* direct colour components */
} Pcoval;
```

Pcolr_rep is defined as:

```
typedef union {
        Prgb        rgb;        /* Red, Green, Blue colour specification */
        Pcieluv     cieluv;     /* CIE L*U*V* colour specification */
        Phls        hls;        /* Hue, Lightness, Saturation colour
                                   specification */
        Phsv        hsv;        /* Hue, Saturation, Value colour
                                   specification */
        Pdata       unsupp;     /* Colour in unsupported colour
                                   model */
} Pcolr_rep;
```

Prgb is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          red;            /* red, hue, and so on */
        Pfloat          green;          /* green, saturation, lightness,
                                           and so on */
        Pfloat          blue;           /* blue, value, saturation,
                                           and so on */
} Prgb;
```

Pcieluv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          cieluv_x;       /* x coefficient */
        Pfloat          cieluv_y;       /* y coefficient */
        Pfloat          cieluv_y_lum;   /* y luminance */
} Pcieluv;
```

Phsv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          satur;          /* saturation */
        Pfloat          value;          /* value */
} Phsv;
```

Phls is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          lightness;      /* lightness */
        Pfloat          satur;          /* saturation */
} Phls;
```

Pdata is defined in phigs.h as follows:

```
typedef struct {
        size_t          size;           /* size of data */
```

```
        char            *data           /* pointer to data */
} Pdata;
```

Pvec3 is defined in phigs.h as follows:

```
typedef struct  {
        Pfloat          delta_x;        /* x magnitude */
        Pfloat          delta_y;        /* y magnitude */
        Pfloat          delta_z;        /* z magnitude */
} Pvec3;
```

Pconorm3 is defined in phigs.h as follows:

```
typedef struct  {
        Pcoval          colr;           /* colour */
        Pvec3           norm;           /* normal */
} Pconorm3;
```

Pcoval and Pvec3 are defined above.

nfa The number of fill areas in the set.

edata A pointer to an array (nfa) of Pedge_data_list structures that specifies the edge data. Pedge_data_list is defined as:

```
typedef struct {
        Pint                num_edges;      /* number of edges */
        Pedge_data_arr      edgedata;       /* edge data */
} Pedge_data_list;
```

Pedge_data_arr is defined as:

```
typedef union {
        Pedge_flag          *edges;         /* array of edge flags */
        /* implementation-dependent data */
} Pedge_data_arr;
```

Pedge_flag is defined as:

```
typedef enum {
        PEDGE_OFF,
        PEDGE_ON
} Pedge_flag;
```

vdata A pointer to an array (nfa) of Pfacet_vdata_list3 structures that specifies each fill area in the set and optionally associated colour information. Pfacet_vdata_list3 is defined as:

```
typedef struct {
        Pint                num_vertices;   /* number of vertices */
        Pfacet_vdata_arr3   vertex_data;    /* facet vertex data */
} Pfacet_vdata_list3;
```

Pfacet_vdata_arr3 is defined as:

```
typedef union {
        Ppoint3         *points;        /* array of points */
        Pptco3          *ptcolrs;       /* array of points and colours */
        Pptnorm3        *ptnorms;       /* array of points and normals */
        Pptconorm3      *ptconorms;     /* array of points, colours, and
                                           normals */
        /* implementation-dependent types can go here */
} Pfacet_vdata_arr3;
```

Ppoint3 is defined as:

```
typedef struct {
        Pfloat          x;              /* x coordinate */
        Pfloat          y;              /* y coordinate */
        Pfloat          z;              /* z coordinate */
} Ppoint3;
```

Pptco3 is defined as:

```
typedef struct {
        Ppoint3         point;          /* point coordinates */
        Pcoval          colr;           /* colour */
} Pptco3;
```

Ppoint3 anc Pcoval are defined above. Pptnorm3 is defined in phigs.h as follows:

```
typedef struct {
        Ppoint3         point;          /* point coordinates */
        Pvec3           norm;           /* normal */
} Pptnorm3;
```

Ppoint3 and Pvec3 are defined above. Pptconorm3 is defined in phigs.h as follows:

```
typedef struct {
        Ppoint3         point;          /* point coordinates */
        Pcoval          colr;           /* colour */
        Pvec3           norm;           /* normal */
} Pptconorm3;                           /* point with colour and
                                           normal */
```

Ppoint3, Pcoval, and Pvec3 are defined above.

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

FFLAG The data per facet flag indicates the information specified for the whole fill area set. The possible values are:

0	PFNONE	*No Facet Data Specified*
1	PFCLR	*Facet Colour Specified*

2 PFNORM *Facet Normal Specified*
3 PFCLRN *Facet Colour and Normal Specified*

EFLAG The data per edge flag indicates the information specified for each edge. The possible values are:

0 PEDNO *No Edge Data Specified*
1 PEDVIS *Edge Visibility Flags Specified*

VFLAG The data per vertex flag indicates the information specified with each vertex of the primitive. The possible values are:

0 PVCOOR *Coordinates Only*
1 PVCLR *Coordinates and Colours*
2 PVNORM *Coordinates and Normals*
3 PVCLRN *Coordinates, Colours, and Normals*

COLTYP

The colour type. If the data per facet flag (FFLAG) indicates that a facet colour is specified, or if the data per vertex flag (VFLAG) indicates that vertex colours are specified, then COLTYP is set to one of the following:

0 PINDIR *Colour Index Specified*
1 PRGB *Red, Green, and Blue*
2 PCIE *CIE Colour Model*
3 PHSV *Hue, Saturation, and Value*
4 PHLS *Hue, Lightness, and Saturation*

IFCOLR

An integer value, which is an index into the workstation colour table for determining the colour of the facet. This value is unused if *FFLAG* is set to either PFNONE or PFNORM or if *COLTYP* is greater than zero.

FCOLR(3)

An array of reals containing the colours for the facet. This array is unused if *FFLAG* is set to either PFNONE or PFNORM, or *COLTYP* is equal to zero.

FCOLR(1) = *Red, Hue, and so on*
FCOLR(2) = *Green, Saturation, Lightness, and so on*
FCOLR(3) = *Blue, Value, Saturation, and so on*

FNORM(3)

An array of reals containing the unit normals for the facet. This array is unused if *FFLAG* is set to either PFNONE or PFCLR.

FNORM(1) = *x Magnitude*
FNORM(2) = *y Magnitude*
FNORM(3) = *z Magnitude*

NFA The number of fill areas in the fill area set.

FARR(NFA)

An array of integers containing the end indices the *COORDS* array for each polygon in the fill area set.

NV The number of points used to define the fill area set.

EDARR(NV)

An array of integers containing edge visibility data. The possible values for each element of the array are:

0 *POFF*
1 *PON*

COORDS(3,NV)

An array of reals containing the *x*, *y* and *z* coordinates of the fill area set.

COORDS(1,v) = *x Coordinate*
COORDS(2,v) = *y Coordinate*
COORDS(3,v) = *z Coordinate*

IVCOLR(NV)

An array of integers containing an index to the workstation colour table for each vertex. This array is unused if *VFLAG* is set to either PVCOOR or PVNORM or if *COLTYP* is greater than zero.

VCOLR(3,NV)

An array of reals containing the colours for each vertex. This array is unused if *VFLAG* is set to either PVCOOR or PVNORM, or if *COLTYP* is equal to zero.

VCOLR(1,v) = *Red, Hue, and so on*
VCOLR(2,v) = *Green, Saturation, Lightness, and so on*
VCOLR(3,v) = *Blue, Value, Saturation, and so on*

VNORM(3,NV)

An array of reals containing the unit normals for each vertex. This array is unused if *VFLAG* is set to either PVCOOR or PVCLR.

VNORM(1,v) = *x Magnitude*
VNORM(2,v) = *y Magnitude*
VNORM(3,v) = *z Magnitude*

IMPL(*)

This implementation-defined data type is not currently used.

Execution

Depending on the edit mode, a FILL AREA SET 3 WITH DATA element is inserted into the open structure after the element pointer, or replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the newly created FILL AREA SET 3 WITH DATA element.

When the structure is traversed, FILL AREA SET 3 WITH DATA draws an implicitly closed polygonal area with multiple boundaries. SunPHIGS Extensions implicitly closes the area by extending each fill area boundary from the last point to the first point. Fill areas can be empty, hollow, shaded, or filled with a color, pattern, or hatch. The interior of the primitive is defined by the odd winding rule. The FILL AREA SET 3 WITH DATA element can specify the facet color and normal vector as well as colours and normal vectors for each vertex. These colours and normal vectors are used in conjunction with the current lighting and depth cueing attributes to colour and shade the primitive. Note that

unexpected lighting effects result if the supplied normals are not of unit length. See INTRO (3PP) for a complete description of these values.

The application should insure that the fill area set points are coplanar. Display of non-coplanar fill area set is device-dependent. FILL AREA SET 3 WITH DATA elements containing fewer than three vertices can be created, but are ignored during structure traversal.

Attributes Applied

The attributes listed below are used to display the FILL AREA SET 3 WITH DATA primitive when the structure is traversed. The Aspect Source Flags (ASFs) describe where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

interior colour	interior colour index ASF
back interior colour	back interior colour ASF
interior style	interior style ASF
back interior style	back interior style ASF
interior style index	interior style index ASF
back interior style index	back interior style index ASF
interior shading method	interior shading method ASF
back interior shading method	back interior shading method ASF
interior reflectance characteristics	interior reflectance characteristics ASF
back interior reflectance characteristics	back interior reflectance characteristics ASF
reflectance properties	reflectance properties ASF
back reflectance properties	back reflectance properties ASF
interior index	
edge colour	edge colour index ASF
edge flag	edge flag ASF
edgetype	edgetype ASF
edgewidth scale factor	edgewidth scale factor ASF
edge index	
face distinguishing mode	
face culling mode	
depth cue index	
light source state	
name set	

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

900 Storage overflow has occurred in PHIGS

2004 FORTRAN-specific Binding Error — Ignoring function, input parameter size out of range

SEE ALSO

INTRO (3PP)
SET OF FILL AREA SET 3 WITH DATA (3PP)
FILL AREA SET 3 (3P)

NAME

INQUIRE COLOUR MAPPING FACILITIES – inquire for colour mapping facilities for the specified workstation type

SYNOPSIS

C Syntax

void
pinq_colr_map_facs (type, length, start, error_ind, colr_map_facs, total_length)

Pint	type;	*workstation type*
Pint	length;	*length of application list*
Pint	start;	*starting position*
Pint	*error_ind;	*OUT error indicator*
Pcolr_map_facs	*colr_map_facs;	*OUT colour mapping facilities*
Pint	*total_length;	*OUT total length of list*

FORTRAN Syntax

SUBROUTINE pqcmf (WTYPE, N, ERRIND, NMETH, METH, NPI)

INTEGER	WTYPE	*workstation type*
INTEGER	N	*element of list of available colour mapping methods*
INTEGER	ERRIND	*OUT error indicator*
INTEGER	NMETH	*OUT number of available colour mapping methods*
INTEGER	METH	*OUT Nth available colour mapping method*
INTEGER	NPI	*OUT number of predefined indices*

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE COLOUR MAPPING FACILITIES returns the available colour mapping facilities for a specified workstation type.

C Input Parameter

type The workstation type.

length The number of Pint entries in the array the application has allocated and assigned to *colr_map_facs→methods.ints*. If zero is specified, then the list of available colour mapping methods is not returned, but the number of available colour mapping methods is returned in the *total_length* argument.

start Starting position of inquiry. The elements in the list, beginning with the item number specified by *start*, are copied sequentially into *colr_map_facs→methods.ints* until *colr_maps_facs→methods.ints* is full or all the elements have been copied.

C Output Parameters

error_ind
A pointer to the location to store the error number of any error detected by this function.

colr_map_facs
A pointer to the variable in which to store the colour mapping facilities. Pcolr_map_facs is defined in phigs.h as:

```
typedef struct {
        Pint_list       meths;          /* available colour mapping
                                           methods */
        Pint            num_pred_inds;  /* number of predefined
                                           indices */
} Pcolr_map_facs;
```

meths returns the number and list of available colour mapping methods for the specified workstation type. Pint_list is defined in phigs.h as follows:

num_pred_ind returns the number of predefined colour mapping representations.

```
typedef struct {
        Pint            num_ints;       /* number of Pints in list */
        Pint            *ints;          /* list of integers */
} Pint_list;
```

The *num_ints* component specifies the number of elements in the list. The *ints* component is a pointer to a list *num_ints* long.

The application must allocate memory for *length* elements in the list of *ints*.

The predefined colour mapping methods in phigs.h are:

1 *PCOLR_MAP_TRUE*
2 *PCOLR_MAP_PSEUDO*
3 *PCOLR_MAP_PSEUDO_N*

total_length
A pointer to the integer in which to return the length of the list. This is the value required for *length* if all the items in the list are to be returned.

FORTRAN Input Parameters

WTYPE Workstation type

N Index of the desired element from the list of available colour mapping methods for this workstation type. A value of 0 can be used without error to inquire for the number of available colour mapping methods.

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

NMETH
The total length of the list of available colour mapping methods for this workstation type.

METH The Nth available colour mapping method.

NPI The number of predefined colour mapping representations.

Execution

If the inquired information is available, the error indicator is returned as zero and values are returned in the output parameters.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the following error numbers to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING (3P) is not invoked when an error is detected by this function.

Errors

002	Ignoring function, function requires state (PHOP,*,*,*)
051	Ignoring function, this information is not yet available for this generic workstation type; open a workstation of this type and use the specific workstation type
052	Ignoring function, workstation type not recognized by the implementation
059	Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
062	Ignoring function, this information is not available for this MO workstation type

SEE ALSO

SET COLOUR MAPPING REPRESENTATION (3PP)
SET COLOUR MAPPING INDEX (3PP)

NAME

INQUIRE COLOUR MAPPING METHOD FACILITIES – inquire for the available colour mapping facilities specific to a colour mapping method.

SYNOPSIS

C Syntax

```
void
pinq_colr_map_method_facs ( type, map_method, error_ind, map_st )
Pwstype         type;          workstation type
Pint            map_method;    mapping method
Pint            *error_ind;    OUT error indicator
Pcolr_map_st    *map_st;       OUT method-specific facilities
```

FORTRAN Syntax

```
SUBROUTINE pqcmmf ( WTYPE, METH, MLDR, ERRIND, LDR, DATREC )
INTEGER          WTYPE          workstation type
INTEGER          METH           colour mapping method of interest
INTEGER          MLDR           dimension of data record array
INTEGER          ERRIND         OUT error indicator
INTEGER          LDR            OUT number of array elements used in
                                data record
CHARACTER*80     DATREC(MLDR)   OUT data record
```

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE COLOUR MAPPING METHOD FACILITIES inquires for the available colour mapping facilities specific to a specified colour mapping method.

C Input Parameters

type The workstation type for which to get the colour mapping method facilities.

map_method

The colour mapping method of interest. The colour mapping methods defined in phigs.h are as follows:

1 PCOLR_MAP_TRUE
2 PCOLR_MAP_PSUEDO
3 PCOLR_MAP_PSEUDO_N

C Output Parameters

error_ind

A pointer to the location in which to store the error number of any error that this function detects.

map_st A pointer to a Pcolr_map_st structure in which to store the information. Pcolr_map_st is defined in phigs.h as:

```
typedef union {
        Pint        int_data;       for map_method 1 and 2
} Pcolr_map_st;
```

For colour mapping method PCOLR_MAP_TRUE, *int_data* will contain the number of available true colours. For colour mapping method PCOLR_MAP_PSEUDO, *int_data* will contain the maximum number of available pseudo colour entries. There is no information returned for colour mapping method PCOLR_MAP_PSEUDO_N.

FORTRAN Input Parameters

WTYPE The workstation type for which to return the colour mapping method facilities.

METH The colour mapping method of interest. The colour mapping methods defined in phigs77.h are as follows:

1 PCMAPTRUE
2 PCMAPPSEUDO
3 PCMAPPSEUDON

MLDR The dimension of the data record array, DATREC.

FORTRAN Output Parameters

ERRIND

The error number of any error detected by this function. The dimension of the data record array, DATREC.

DATREC(MLDR)

The data record array. This must subsequently be passed to UNPACK DATA RECORD (3P) to access its contents. The contents of the data record are determined by the colour mapping method:

IL = the number of integers, 0 or 1.

IA = For colour mapping method PCMAPTRUE, IA is an array of a single integer containing the number of available true colours. For colour mapping method PCMAPPSEUDO, IA is an array of a single integer containing the maximum number of pseudo colour entries. For colour mapping method PCMAPPSEUDON, IA is empty.

RL = the number of real values, 0

SL = the number of strings, 0

Execution

If the inquired information is available, then the error indicator is returned as zero, and the requested information is returned.

If the inquired information is not available, the values returned in the output parameters are undefined and the error indicator is set to one of the following error numbers to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING (3P) is not invoked when an error is detected by this function.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)

051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type

052 Ignoring function, workstation type not recognized by the implementation

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT or OUTIN)

062 Ignoring function, this information is not available for this MO workstation type

126 Ignoring function, the specified colour mapping method is not available on the specified workstation.

SEE ALSO

INQUIRE COLOUR MAPPING FACILITIES (3PP)
PHIGS WORKSTATION DESCRIPTION TABLE (7P)

NAME

INQUIRE COLOUR MAPPING REPRESENTATION – inquire a colour mapping representation for a given workstation

SYNOPSIS

C Syntax

```
void
pinq_colr_map_rep ( ws, index, type, store, error_ind, map_method, map_data )
Pint            ws;          workstation identifier
Pint            index;       colour mapping index
Pinq_type       type;        type of returned value
Pstore          store;       handle to Store object
Pint            *error_ind;  OUT error indicator
Pint            *map_method; OUT mapping method
Pcolr_map_data  **map_data;  OUT mapping method data record
```

FORTRAN Syntax

```
SUBROUTINE pqcmr ( WKID , CMRI, TYPE, MLDR, ERRIND, MMETH, LDR,
    DATREC )
INTEGER          WKID          workstation type
INTEGER          CMRI          predefined index
INTEGER          TYPE          type of returned value
INTEGER          MLDR          dimension of data record array
INTEGER          ERRIND        OUT error indicator
INTEGER          MMETH         OUT colour mapping method
INTEGER          LDR           OUT number of array elements used in
                               data record
CHARACTER*80     DATREC(MLDR)  OUT data record
```

Required PHIGS Operating States

(PHOP,WSOP, *, *)

DESCRIPTION

Purpose

INQUIRE COLOUR MAPPING REPRESENTATION returns a specified colour mapping table entry in a workstation's state list.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the *store* argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area such that there is sufficient memory for the specific inquiry. The data record within the store buffer is accessed by the pointer pointed to by *map_data*.

ws The workstation identifier.

index The colour mapping table index of the entry to return.

type An enumerated value specifying whether the values to be returned are those originally specified by the application (PINQ_SET), or those resulting after PHIGS mapped them to ones available on the workstation (PINQ_REALIZED). A Pinq_type structure is defined in phigs.h as:

```
typedef enum {
        PINQ_SET,
        PINQ_REALIZED
} Pinq_type;
```

store The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

C Output Parameters

error_ind

A pointer to the location to store the error number of any error detected by this function.

map_method

A pointer to the location to store the colour mapping method of the representation. The colour mapping methods are defined in phigs.h as follows:

1 PCOLR_MAP_TRUE
2 PCOLR_MAP_PSEUDO
3 PCOLR_MAP_PSEUDO_N

map_data

A pointer to a pointer to a Pcolr_map_data structure that contains the information specific to the data mapping method. PHIGS will allocate space for the structure in the Pstore object and return the pointer to that allocated space in this parameter. Pcolr_map_data is defined in phigs.h as follows:

```
typedef union {
        struct {
                Pint            colr_model;
                Pfloat_list     weights;
                Pcolr_rep_list  colrs;
        } method_r2
        struct {
                Pint            colr_model;
                Pfloat_list_list colr_lists;
        } method_r3
} Pcolr_map_data;
```

Pfloat_list is defined in phigs.h as follows:

```
typedef struct {
        Pint        num_floats;        /* number of Pfloats in list */
```

```
        Pfloat      *floats;        /* list of floats *
} Pfloat_list;
```

Pfloat_list_list is defined in phigs.h as follows:

```
typedef struct {
        Pint        num_lists;      /* number of lists in list */
        Pfloat_list *lists;         /* list of float lists */
} Pfloat_list_list;
```

Pcolr_rep_list is defined in phigs.h as follows:

```
typedef struct {
        Pint        num_colr_reps;  /* number of colours */
        Pcolr_rep   *colr_reps;     /* array of colours */
} Pcolr_rep_list;
```

Pcolr_rep is defined in phigs.h as follows:

```
typedef union {
        Prgb        rgb;            /* Red Green Blue colour
                                       specification */
        Pcieluv     cieluv;         /* CIE L*U*V* colour specification */
        Phls        hls;            /* Hue Lightness Saturation colour
                                       specification */
        Phsv        hsv;            /* Hue Saturation Value colour
                                       specification */
        Pdata       unsupp;         /* Colour in unsupported colour
                                       model */
} Pcolr_rep;
```

Prgb is defined in phigs.h as follows:

```
typedef struct {
        Pfloat      red;            /* red, hue, and so on */
        Pfloat      green;          /* green, saturation, lightness, and
                                       so on */
        Pfloat      blue;           /* blue, value, saturation, and
                                       so on */
} Prgb;
```

Pcieluv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat      cieluv_x;       /* x coefficient */
        Pfloat      cieluv_y;       /* y coefficient */
        Pfloat      cieluv_y_lum;   /* y luminance */
} Pcieluv;
```

Phsv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          satur;          /* saturation */
        Pfloat          value;          /* value */
} Phsv;
```

Phls is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          lightness;      /* lightness */
        Pfloat          satur;          /* saturation */
} Phls;
```

Pdata is defined in phigs.h as follows:

```
typedef struct {
        size_t          size;           /* size of data */
        char            *data           /*pointer to data */
} Pdata;
```

There is no data associated with mapping method PCOLR_MAP_TRUE. The data for method PCOLR_MAP_PSEUDO is a list of weights and a list of direct colours. *colr_model* specifies the type of colours given in the list of colours, *colrs*. The colour models are defined in phigs.h as follows:

1 PMODEL_RGB
2 PMODEL_CIELUV
3 PMODEL_HSV
4 PMODEL_HLS

The data for mapping method PCOLR_MAP_PSEUDO_N is *N* lists of direct colours. These colours may be of the same types as for PCOLR_MAP_PSEUDO.

FORTRAN Input Parameters

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which the system will place the contents of the device's input data record. The contents of the data record are subsequently extracted by the application with the function UNPACK DATA RECORD (3P). The allocated dimension of the character array is passed in the MLDR argument. The dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS returns the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values are returned. These are LDR, and the colour mapping method MMETH.

Error 2001 is returned when MLDR is too small; but not when it is zero.

WKID The identifier of the workstation for which capabilities are inquired.

CMRI Entry in the workstation table of predefined colour mapping representations to

be returned.

TYPE The type of returned value desired — either PSET or PREALI.

MLDR The dimension of the data record array, DATREC.

FORTRAN Output Parameters

ERRIND

The error number of any error detected by this function.

MMETH

The colour mapping method. This may be any of the following values defined in phigs77.h:

1 PCMAPTRUE
2 PCMAPPSEUDO
3 PCMAPPSEUDON

LDR The required dimension of the data record array, DATREC.

DATREC

The data record array. This must subsequently be passed to UNPACK DATA RECORD (3P) to access its content. The content of the data record is determined by the colour mapping method; see SET COLOUR MAPPING REPRESENTATION (3P) for a content description.

Execution

If the inquired information is available, the error indicator is returned as zero and values are returned in the output parameters.

If the specified colour mapping index is not present in the colour mapping table on the workstation and the specified type of returned value is *Realized*, the representation for the colour mapping index zero is returned.

If the inquired information is not available, the values returned in the output parameters are undefined and the error indicator is set to one of the following error numbers to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, the specified workstation is not open

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is neither OUTPUT nor OUTIN)

101 Ignoring function, the specified representation has not been defined

121 Ignoring function, the colour mapping index is less than zero

SEE ALSO

SET COLOUR MAPPING REPRESENTATION (3PP)
SET COLOUR MAPPING INDEX (3PP)
INQUIRE LIST OF COLOUR MAPPING INDICES (3PP)
INQUIRE PREDEFINED COLOUR MAPPING REPRESENTATION (3PP)

CREATE STORE (3P)
DELETE STORE (3P)
UNPACK DATA RECORD (3P)

NAME

INQUIRE CURVE AND SURFACE FACILITIES – inquire for the available curve and surface facilities of a workstation type

SYNOPSIS

C Syntax

void
pinq_curv_surf_facs (type, cat_len, cat_st, sat_len, sat_st, tcat_len, tcat_st, psc_len, psc_st, error_ind, facil, tot_cat_len, tot_sat_len, tot_tcat_len, tot_psc_len)

Pwstype	type;	*workstation type*
Pint	cat_len;	*length of curve approximation types list*
Pint	cat_st;	*starting position*
Pint	sat_len;	*length of surface approximation types list*
Pint	sat_st;	*starting position*
Pint	tcat_len;	*length of trim curve approximation types list*
Pint	tcat_st;	*starting position*
Pint	psc_len;	*length of parametric surface characteristics list*
Pint	psc_st;	*starting position*
Pint	*error_ind;	*OUT error indicator*
Pcurvsurf_facs	*facil;	*OUT curve and surface facilities*
Pint	*tot_cat_len;	*OUT total length of curve approximation types list*
Pint	*tot_sat_len;	*OUT total length of surface approximation types list*
Pint	*tot_tcat_len;	*OUT total length of trim curve approximation types list*
Pint	*tot_psc_len;	*OUT total length of parametric surface characteristics list*

FORTRAN Syntax

SUBROUTINE pqcsf (WKTYPE, ICAT, ISAT, ITCAT, IPSC, ERRIND, MBSORD, MTCORD, NCAT, CAT, NSAT, SAT, NTCAT, TCAT, NPSC, PSC)

INTEGER	WKTYPE	*workstation type*
INTEGER	ICAT	*list element of curve approximation type requested*
INTEGER	ISAT	*list element of surface approximation type requested*
INTEGER	ITCAT	*list element of trim curve approximation type requested*
INTEGER	IPSC	*list element of parametric surface characteristics requested*
INTEGER	ERRIND	*OUT error indicator*
INTEGER	MBSORD	*OUT maximum non-uniform rational B-spline order*
INTEGER	MTCORD	*OUT maximum trimming curve order*
INTEGER	NCAT	*OUT number of available curve approximation types*
INTEGER	CAT	*OUT ICAT-th element of list of available curve approximation types*
INTEGER	NSAT	*OUT number of available surface approximation types*
INTEGER	SAT	*OUT ISAT-th element of list of available surface approximation types*
INTEGER	NTCAT	*OUT number of available trim curve approximation types*
INTEGER	TCAT	*OUT ITCAT-th element of list of available trim curve approximation types*

	INTEGER	NPSC	*OUT number of available parametric surface characteristics types*
	INTEGER	PSC	*OUT IPSC-th element of list of available parametric surface characteristics types*

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE CURVE AND SURFACE FACILITIES returns the available curve and surface facilities associated with the specified workstation type.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

All of the following data types are predefined in phigs.h.

type The workstation type in question.

cat_len The number of entries to return from the list of curve approximation types.

cat_st The starting position of the portion of the list to return.

sat_len The number of entries to return from the list of surface approximation types.

sat_st The starting position of the portion of the list to return.

tcat_len The number of entries to return from the list of trimming curve approximation types.

tcat_st The starting position of the portion of the list to return.

psc_len The number of entries to return from the list of parametric surface characteristics.

psc_st The starting position of the portion of the list to return.

C Output Parameters

error_ind

A pointer to the location to store the error number of any error detected by this function.

facil A pointer to a Pcurvsurf_facs structure, which returns the requested information. Pcurvsurf_facs is defined as:

```
typedef struct {
        Pint        max_bsp_order;    /* maximum B-spline order */
        Pint        max_tc_order;     /* maximum trim curve order */
        Pint_list   cat_types;        /* list of curve approximation types */
        Pint_list   sat_types;        /* list of surface approximation types */
        Pint_list   tcat_types;       /* list of trim curve approximation
                                         types */
        Pint_list   psc_types;        /* list of parametric surface
                                         characteristics types */
} Pcurvsurf_facs;
```

Pint_list is defined as:

```
typedef struct {
        Pint    num_ints;  /* number of Pints in list */
        Pint    *ints;     /* list of integers */
} Pint_list;
```

Prior to calling this function, the integers field of the Pint_list structures must contain a pointer to an array of integers. This array must be at least as large as the corresponding *len* parameter.

tot_cat_len
A pointer to an integer, which returns the total length of the curve approximation type list as stored in the workstation description table.

tot_sat_len
A pointer to an integer, which returns the total length of the surface approximation type list as stored in the workstation description table.

tot_tcat_len
A pointer to an integer, which returns the total length of the trimming curve approximation type list as stored in the workstation description table.

tot_psc_len
A pointer to an integer, which returns the total length of the parametric surface characteristics list as stored in the workstation description table.

FORTRAN Input Parameters

WKTYPE
The workstation type in question.

ICAT The index of the entry desired from the curve approximation type list.

ISAT The index of the entry desired from the surface approximation type list.

ITCAT The index of the entry desired from the trimming curve approximation type list.

IPSC The index of the entry desired from the parametric surface characteristics list.

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

MBSORD
The maximum B-spline order supported.

MTCORD
The maximum trimming curve order supported.

NCAT The number of entries in the list of available curve approximation types.

CAT The requested entry of the curve approximation type list.

NSAT The number of entries in the list of available surface approximation types.

SAT The requested entry of the surface approximation type list.

NTCAT The number of entries in the list of available trimming curve approximation

types.

TCAT The requested entry of the trimming curve approximation type list.

NPSC The number of entries in the list of available parametric surface characteristics types.

PSC The requested entry of the parametric surface characteristics list.

Execution

If the inquired information is available, the error indicator is returned as zero and values are returned in the output parameters.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the error numbers listed below to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)

051 Ignoring function, this information is not yet available for this generic workstation type; open a workstation of this type and use the specific workstation type

052 Ignoring function, workstation type not recognized by the implementation

062 Ignoring function, this information is not available for this MO workstation type

SEE ALSO

PHIGS WORKSTATION DESCRIPTION TABLE (7P)
NON-UNIFORM B-SPLINE CURVE (3PP)
NON-UNIFORM B-SPLINE SURFACE (3PP)
SET CURVE APPROXIMATION CRITERIA (3PP)
SET PARAMETRIC SURFACE CHARACTERISTICS (3PP)
SET SURFACE APPROXIMATION CRITERIA (3PP)

NAME

INQUIRE DEPTH CUE FACILITIES – inquire for depth cue facilities for specified workstation type

SYNOPSIS

C Syntax

```
void
pinq_dcue_facs ( type, error_ind, facilities )
Pwstype    type;          workstation type
Pint       *error_ind;    OUT error indicator
Pint       *facilities;   OUT depth cue facilities
```

FORTRAN Syntax

```
SUBROUTINE pqdcf ( WKTYPE, ERRIND, NPDCI )
INTEGER    WKTYPE     workstation type
INTEGER    ERRIND     OUT error indicator
INTEGER    NPDCI      OUT number of predefined indices
```

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE DEPTH CUE FACILITIES returns the attribute values of a specified entry in the workstation table of depth cue facilities.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameter

wstype The workstation type.

C Output Parameters

error_ind
A pointer to the location to store the error number of any error detected by this function.

npdci A pointer to the location to store the number of predefined indices.

FORTRAN Input Parameter

WKTYPE
The identifier of the workstation type.

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

NPDCI The number of predefined depth cue indices.

Execution

If the inquired information is available, the error indicator is returned as zero and values are returned in the output parameters.

If the inquired information is not available, then the values returned in the output parameters are undefined and the error indicator is set to one of the following error numbers to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)

051 Ignoring function, this information is not yet available for this generic workstation type; open a workstation of this type and use the specific workstation type

052 Ignoring function, workstation type not recognized by the implementation

062 Ignoring function, this information is not available for this MO workstation type

SEE ALSO

SET DEPTH CUE REPRESENTATION (3PP)
SET DEPTH CUE INDEX (3PP)
PHIGS WORKSTATION DESCRIPTION TABLE (7P)

NAME

INQUIRE DEPTH CUE REPRESENTATION – inquire for depth cue representation for specified workstation

SYNOPSIS

C Syntax

void
pinq_dcue_rep (ws, index, type, error_ind, rep)

Pint	ws;	*workstation identifier*
Pint	index;	*polyline index*
Pinq_type	type;	*type of returned value*
Pint	*error_ind;	*OUT error indicator*
Pdcue_bundle	*rep;	*OUT depth cue representation*

FORTRAN Syntax

SUBROUTINE pqdcr (WKID, IDEPTH, ITYPE, ERRIND, MODE, BPLANE, FPLANE, BSCALE, FSCALE, COLMOD, IDCOLR, DCOLR)

INTEGER	WKID	*workstation identifier*
INTEGER	IDEPTH	*depth cue index*
INTEGER	ITYPE	*inquiry type (PSET, PREALI)*
INTEGER	ERRIND	*OUT error indicator*
INTEGER	MODE	*OUT depth cue mode*
REAL	BPLANE, FPLANE	*OUT back and front depth planes*
REAL	BSCALE, FSCALE	*OUT back and front scale values*
INTEGER	COLMOD	*OUT colour model*
INTEGER	IDCOLR	*OUT indirect depth cue colour*
REAL	DCOLR(3)	*OUT array of depth cue colour*

Required PHIGS Operating States

(PHOP,WSOP, *, *)

DESCRIPTION

Purpose

INQUIRE DEPTH CUE REPRESENTATION returns the attribute values of a specified entry in the workstation table of depth cue representations. For a full description of the contents of the bundle table entry and a description of the depth cue representation table, see SET DEPTH CUE REPRESENTATION .

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

ws The workstation identifier.

index The depth cue index.

type An enumerated value specifying whether the values to be returned are those originally specified by the application (PINQ_SET), or are those resulting after PHIGS mapped them to ones available on the workstation (PINQ_REALIZED). A Pinq_type structure is defined as:

typedef enum {

```
        PINQ_SET,
        PINQ_REALIZED
    } Pinq_type;
```

C Output Parameters

error_ind

A pointer to the location to store the error number of any error detected by this function.

rep A pointer to a Pdcue_bundle data structure containing attribute values defining a depth cue representation. A Pdcue_bundle structure is defined as:

```
typedef struct {
    Pdcue_mode    mode;          /* depth cue mode */
    Pfloat        ref_planes[2]; /* depth cue reference planes */
    Pfloat        scaling[2];    /* depth cue scaling */
    Pgcolr        colr;          /* depth cue colour */
} Pdcue_bundle;
```

Pdcue_mode is defined as:

```
typedef enum {
    PSUPPRESSED,
    PALLOWED
} Pdcue_mode;
```

ref_planes and *scaling* entries of 0 are back reference plane and corresponding scaling factors. *ref_planes* and *scaling* entries of 1 are front reference plane and corresponding scaling factors. These scaling factors define the portion of the primitive colour that should be combined with the depth cue colour, as a function of *z* in NPC.

Pgcolr is defined as:

```
typedef struct {
    Pint          type;     /* indirect, RGB, CIE, HSV, HLS */
    union {
      Pint        ind;      /* index in workstation colour
                               bundle table */
      struct {
        Pfloat    x;        /* red, hue, and so on */
        Pfloat    y;        /* green, saturation, lightness,
                               and so on */
        Pfloat    z;        /* blue, value, saturation, and so on */
      } general;
    } val;
} Pgcolr;
```

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

WKID The identifier of the workstation for which the depth cue representation is being inquired.

IDEPTH

The depth cue index into the workstation depth cue table.

ITYPE An enumerated value specifying whether the values to be returned are those originally specified by the application (*Set*), or those resulting after PHIGS mapped them to ones available on the workstation (*Realized*). Valid values are:

0	PSET	*Set*
1	PREALI	*Realized*

FORTRAN Output Parameters

ERRIND

The error number of any error detected by this function.

MODE

The depth cue MODE controls depth cue disabling or enabling. The depth cue mode contains one of the following values:

0	PSUPPR	*Disable Depth Cueing*
1	PALLOW	*Enable Depth Cueing*

BPLANE,FPLANE

The back and front depth planes, *z* values in NPC. BPLANE is the back depth plane, which defines a back reference plane for depth cueing. FPLANE is the front depth plane, which defines a front reference plane for depth cueing.

BSCALE,FSCALE

The back and front scale values. BSCALE is the back scale value, between 0.0 and 1.0, that corresponds to the *z* value of the back reference plane. FSCALE is a front scale value, between 0.0 and 1.0, that corresponds to the *z* value of the front depth cue reference plane. These values define the portion of the primitive colour that should be combined with the depth cue colour as a function of *z* in NPC.

COLMOD

COLMOD is the colour model that determines the type of colour to be used. If this value is PINDIR or zero, then IDCOLR will contain an indexed colour value. If COLMOD is greater than zero, then DCOLR will contain the depth cue colour. The valid values for COLMOD are:

0	PINDIR	*Colour Index Specified*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

IDCOLR

IDCOLR is an index into the workstation colour table. This integer value determines the depth cue colour. This value is not used if COLMOD is greater than

zero.

DCOLR(3)

DCOLR(3) is an array of reals containing the depth cue colour. This array is not used when COLMOD is equal to zero. The reals are stored as:

DCOLR(1) = *Red, Hue, and so on*
DCOLR(2) = *Green, Saturation, Lightness, and so on*
DCOLR(3) = *Blue, Value, Saturation, and so on*

Execution

If the inquired information is available, the error indicator is returned as zero and values are returned in the output parameters.

If the specified depth cue index is not present in the depth cue table on the workstation and the specified type of returned value is *Realized*, the representation for the depth cue index zero is returned.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the following error numbers to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, specified workstation is not open

059 Ignoring function, specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

101 Ignoring function, specified representation has not been defined

119 Ignoring function, depth cue index is less than zero

SEE ALSO

SET DEPTH CUE REPRESENTATION (3PP)
SET DEPTH CUE INDEX (3PP)
INQUIRE LIST OF DEPTH CUE INDICES (3PP)
INQUIRE PREDEFINED DEPTH CUE REPRESENTATION (3PP)

NAME

INQUIRE DIRECT COLOUR MODEL FACILITIES – inquire for list of workstation direct colour facilities

SYNOPSIS

C Syntax

void
pinq_direct_colr_model_facs (type, length, start, error_ind, total_length, models)

Pwstype	type;	*workstation type*
Pint	length;	*length of application list*
Pint	start;	*starting position*
Pint	*error_ind;	*OUT error indicator*
Pint_list	*models;	*OUT list of colour models*
Pint	*total_length;	*OUT length of list in PHIGS*

FORTRAN Syntax

SUBROUTINE pqdcmf (WKTYPE, ICM, ERRIND, NICM, CMI)

INTEGER	WKTYPE	*workstation type*
INTEGER	ICM	*index into the colour model list*
INTEGER	ERRIND	*OUT error indicator*
INTEGER	NICM	*OUT number of colour models*
INTEGER	CMI	*ICM-th element of colour models*

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE DIRECT COLOUR MODEL FACILITIES inquires for a list of workstation direct colour facilities.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

All of the following data types are predefined in phigs.h.

type The workstation type.

length The number of *ints* in the *models* output parameter for which the application has allocated memory. *length* is the number of list elements that the system can return in *models→ints*. If a value of 0 is used here, no data is returned in the *models→ints* list, but the total number of elements is returned in *total_length*.

start Starting position of inquiry. The elements in the list, beginning with the item number specified by *start*, are copied sequentially into *models→ints* until *models→ints* is full or all the elements have been copied.

C Output Parameters

error_ind
A pointer to the location for storing the error number of any error detected by this function.

models A pointer to a Pint_list that returns the portion of the list of available colour

models from the workstation description table, starting with *start*. Pint_list is defined in phigs.h as:

```
typedef struct {
        Pint    num_ints;   /* number of integers */
        Pint    *ints;      /* list of integers */
} Pint_list;
```

Constants defined for colour model are:

1 *PMODEL_RGB*
2 *PMODEL_CIELUV*
3 *PMODEL_HSV*
4 *PMODEL_HLS*

The pointer to the list of integers must be initialized to an array of *length* Pint elements.

total_length
A pointer to an integer in which the system returns the total number of elements in the list. This is the value required for *length* if all elements in the list are to be returned.

FORTRAN Input Parameters

WKTYPE
The workstation type.

ICM The index of the entry desired from the list of colour models.

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

NICM The number of colour models.

CMI The requested entry from the list of colour types. The valid values are:

0	PINDIR	*Indirect*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

Execution

If the inquired information is available; then the error indicator is returned as zero, and values are returned in the output parameters.

If the inquired information is not available; then the values returned in the output parameters are undefined, and the error indicator is set to one of the following error numbers to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)

051 Ignoring function, this information unavailable for this workstation type

052 Ignoring function, workstation type not recognized by the implementation

062 Ignoring function, this information is not available for this MO workstation type

SEE ALSO

INQUIRE COLOUR MODEL FACILITIES (3P)
PHIGS WORKSTATION DESCRIPTION TABLE (7P)

NAME

INQUIRE DYNAMICS OF WORKSTATION ATTRIBUTES PLUS – inquire for the dynamics of PHIGS PLUS workstation attributes for a workstation type

SYNOPSIS

C Syntax

```
void
pinq_dyns_ws_attrs_plus ( type, error_ind, attr )
Pint                  type;        workstation type
Pint                  *error_ind;  OUT error indicator
Pdyns_ws_attrs_plus   *attr;       OUT attribute dynamics
```

FORTRAN Syntax

```
SUBROUTINE pqdwap ( WTYPE, ERRIND, LIGHTR, DCUER, CMAPR )
INTEGER   WTYPE    workstation type
INTEGER   ERRIND   OUT error indicator
INTEGER   LIGHTR   OUT light source representation changeable
INTEGER   DCUER    OUT depth cue representation changeable
INTEGER   CMAPR    OUT colour mapping representation changeable
```

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE DYNAMICS OF WORKSTATION ATTRIBUTES PLUS determines the dynamics of PHIGS PLUS workstation attributes for a specified workstation type.

C Input Parameter

type The workstation type whose capabilities are inquired.

C Output Parameters

error_ind

A pointer to the location to store the error number of any error that this function detects.

attr A pointer to a structure in which the system returns the dynamic modification accepted values for various workstation attributes. Pdyns_ws_attrs_plus is defined in phigs.h as follows:

```
typedef struct {
        Pdyn_mod    light_src_rep;    /* light source representation */
        Pdyn_mod    depth_cue_rep;    /* depth cue representation */
        Pdyn_mod    colr_map_rep;     /* colour mapping
                                         representation */
} Pdyns_ws_attrs_plus;
```

The Pdyn_mod is an enumeration defined in phigs.h as follows:

```
typedef enum {
        PDYN_IRG,     /* modification leads to an Implicit Regeneration */
        PDYN_IMM,     /* modification is performed Immediately, without an
```

```
                                    implicit regeneration */
        PDYN_CBS                /* modification Can Be Simulated before an implicit
                                    regeneration is permitted */
    } Pdyn_mod;
```

FORTRAN Input Parameter

WTYPE The workstation type whose capabilities are being inquired.

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

LIGHTR
The dynamic modification accepted value for light source representation changes. This argument can return any of the following values defined in phigs77.h:

- *PIRG* modification leads to *Implicit Regeneration*.
- *PIMM* modification is performed *Immediately*, without implicit regeneration.
- *PCBS* modification *Can Be Simulated* before an implicit regeneration is permitted.

DCUER
The dynamic modification accepted value for depth cue representation changes. The value returned may be any of PIRG, PIMM, or PCBS, defined above.

CMAPR
The dynamic modification accepted value for colour mapping representation changes. The value returned can be any of PIRG, PIMM, or PCBS, defined above.

Execution

If the inquired information is available, the error indicator is returned as zero and values are returned in the output parameters.

If the inquired information is not available, the values returned in the output parameters are undefined and the error indicator is set to one of the following error numbers to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

ERRORS

- 002 Ignoring function, function requires state (PHOP, *, *, *)
- 051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type
- 052 Ignoring function, workstation type not recognized by the implementation
- 059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT or OUTIN)
- 062 Ignoring function, this information is not available for this MO workstation type

SEE ALSO

INQUIRE DYNAMICS OF WORKSTATION ATTRIBUTES (3P)
PHIGS WORKSTATION DESCRIPTION TABLE (7P)

NAME

INQUIRE EDGE REPRESENTATION PLUS – inquire for extended edge representation on a workstation

SYNOPSIS

C Syntax

void
pinq_edge_rep_plus (ws, index, type, error_ind, rep)

Pint	ws;	*workstation identifier*
Pint	index;	*edge index*
Pinq_type	type;	*type of returned value*
Pint	*error_ind;	*OUT error indicator*
Pedge_bundle_plus	*rep;	*OUT edge representation*

FORTRAN Syntax

SUBROUTINE pqeer (WKID, INDEX, ITYPE, ERRIND, EFLAG, ETYPE, EWIDTH, COLMOD, IECOLR, ECOLR)

INTEGER	WKID	*workstation identifier*
INTEGER	INDEX	*edge bundle index*
INTEGER	ITYPE	*type of returned values (PSET, PREALI)*
INTEGER	ERRIND	*OUT error indicator*
INTEGER	EFLAG	*OUT edge flag (PON/POFF)*
INTEGER	ETYPE	*OUT edge type*
REAL	EWIDTH	*OUT edge width scale factor*
INTEGER	COLMOD	*OUT colour model*
INTEGER	IECOLR	*OUT indirect colour*
REAL	ECOLR(3)	*OUT array of colour*

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

INQUIRE EDGE REPRESENTATION PLUS inquires for an extended edge representation on a workstation.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

ws The workstation identifier.

index The edge index.

type An enumerated value specifying whether the values to be returned are those originally specified by the application (PINQ_SET), or those resulting after PHIGS mapped them to ones available on the workstation (PINQ_REALIZED). A Pinq_type structure is defined as:

```
typedef enum {
        PINQ_SET,
        PINQ_REALIZED
```

```
} Pinq_type;
```

C Output Parameters

error_ind
A pointer to the location to store the error number of any error detected by this function.

rep A pointer to a Pedge_bundle_plus structure which returns the specified extended edge representation. Pedge_bundle_plus is defined as:

```
typedef struct {
        Pedge_flag      flag;       /* edge flag */
        Pint            type;       /* edgetype */
        Pfloat          width;      /* edgewidth scale factor */
        Pgcolr          colr;       /* edge colour */
} Pedge_bundle_plus;
```

Pedge_flag is defined in phigs.h as:

```
typedef enum {
        PEDGE_OFF,
        PEDGE_ON
} Pedge_flag;
```

Pgcolr is defined as:

```
typedef struct {
        Pint    type;   /* indirect, RGB, CIE, HSV, HLS */
        union {
                Pint    ind;    /* index in workstation colour
                                   bundle table */
                struct {
                        Pfloat  x;      /* red, hue, and so on */
                        Pfloat  y;      /* green, saturation, lightness,
                                           and so on */
                        Pfloat  z;      /* blue, value, saturation, and
                                           so on */
                } general;
        } val;
} Pgcolr;
```

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

WKID The identifier of the workstation for which the extended edge representation is being inquired.

INDEX The index into the specified workstation edge bundle table.

ITYPE An enumerated value specifying whether the values to be returned are those originally specified by the application (*Set*), or those resulting after PHIGS mapped them to ones available on the workstation (*Realized*). Valid values are:

0	PSET	*Set*
1	PREALI	*Realized*

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

EFLAG The edge flag.

ETYPE The edge type.

EWIDTH
The edge width scale factor.

COLMOD
The colour model.

IECOLR
An integer containing the index into the workstation colour table for the edge colour.

ECOLR(3)
An array of reals containing the edge colour.

Execution

If the inquired information is available, the error indicator is returned as zero and values are returned in the output parameters.

If the specified edge index is not present in the edge bundle table on the workstation and the specified type of returned value is *Realized*, the representation for edge index one is returned.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the following error numbers to indicate the reason for non-availability.

Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

ERRORS

003	Ignoring function, function requires state (PHOP, WSOP, *, *)
054	Ignoring function, the specified workstation is not open
059	Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
100	Ignoring function, the bundle index value is less than one
101	Ignoring function, the specified representation has not been defined

SEE ALSO

INQUIRE EDGE REPRESENTATION (3P)
SET EDGE REPRESENTATION PLUS (3PP)
INQUIRE PREDEFINED EDGE REPRESENTATION PLUS (3PP)

NAME

INQUIRE INTERIOR FACILITIES PLUS – inquire for extended interior facilities of specified workstation type

SYNOPSIS

C Syntax

void
pinq_int_facs_plus (type, is_length, is_start, hs_length, hs_start, re_length, re_start, sh_length, sh_start, error_ind, facil, tot_is_length, tot_hs_length, tot_re_length, tot_sh_length)

Pint	type;	*workstation type*
Pint	is_length;	*length of application's interior style list*
Pint	is_start;	*starting position*
Pint	hs_length;	*length of application's hatch style list*
Pint	hs_start;	*starting position*
Pint	re_length;	*length of application's reflectance equation list*
Pint	re_start;	*starting position*
Pint	sh_length;	*length of application's shading method list*
Pint	sh_start;	*starting position*
Pint	*error_ind;	*OUT error indicator*
Pint_facs_plus	*facil;	*OUT extended interior facilities*
Pint	*tot_is_length;	*OUT total length of interior styles list*
Pint	*tot_hs_length;	*OUT total length of hatch styles list*
Pint	*tot_re_length;	*OUT total length of reflectance equation list*
Pint	*tot_sh_length;	*OUT total length of shading method list*

FORTRAN Syntax

SUBROUTINE pqeif (WKTYPE, ISTYLE, IHATCH, IREFEQ, ISHADE, ERRIND, NSTYLE, STYLE, NHATCH, HATCH, NREFEQ, REFEQ, NSHADE, SHADE, NPRE)

INTEGER	WKTYPE	*workstation type*
INTEGER	ISTYLE	*list element of interior styles to return*
INTEGER	IHATCH	*list element of hatch styles to return*
INTEGER	IREFEQ	*list element of reflectance equation*
INTEGER	ISHADE	*list element of shading method to return*
INTEGER	ERRIND	*OUT error indicator*
INTEGER	NSTYLE	*OUT number of available interior styles*
INTEGER	STYLE	*OUT ISTYLE-th element of list of available interior styles*
INTEGER	NHATCH	*OUT number of available hatch styles*
INTEGER	HATCH	*OUT IHATCH-th element of list of available hatch style indices*
INTEGER	NREFEQ	*OUT number of available reflectance eqns*
INTEGER	REFEQ	*OUT IREFEQ-th element of list of available reflectance equations*
INTEGER	NSHADE	*OUT number of available shading methods*
INTEGER	SHADE	*OUT ISHADE-th element of list of available shading methods*
INTEGER	NPRE	*OUT number of predefined interior indices*

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE INTERIOR FACILITIES PLUS returns the extended interior support available from a specified workstation type.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

All of the following data types are predefined in phigs.h.

type The workstation type in question.

is_length
The number of entries to return from the list of interior styles.

is_start Starting position of the portion of the list to return.

hs_length
The number of entries to return from the list of hatch styles.

hs_start Starting position of the portion of the list to return.

re_length
The number of entries to return from the list of reflectance equations.

re_start Starting position of the portion of the list to return.

sh_length
The number of entries to return from the list of interior shading method.

sh_start Starting position of the portion of the list to return.

C Output Parameters

error_ind
A pointer to the location in which to store the error number of any error that this function detects.

facil A pointer to a Pint_facs_plus structure, which returns the requested information. Pint_facs_plus is defined in phigs.h as follows:

```
typedef struct {
        Pint        num_int_styles;   /* number of interior styles */
        Pint_style  *int_styles;      /* list of available interior
                                         styles */
        Pint_list   hatch_styles;     /* list of available hatch styles */
        Pint        num_pred_inds;    /* number of predefined interior
                                         indices */
        Pint_list   refl_eqns;        /* list of available reflectance
                                         equations */
        Pint_list   shad_meths;       /* list of available shading
                                         methods */
} Pint_facs_plus;
```

Pint_style is defined in phigs.h as follows:

```
typedef enum {
        PSTYLE_HOLLOW,
        PSTYLE_SOLID,
        PSTYLE_PAT,
        PSTYLE_HATCH,
        PSTYLE_EMPTY,
} Pint_style;
```

Pint_list is defined in phigs.h as follows:

```
typedef struct {
        Pint    num_ints;    /* number of Pints in list */
        Pint    *ints;       /* list of integers */
} Pint_list;
```

Prior to calling this function, the *ints* field of the Pint_list structures and interiors must contain a pointer to an application supplied buffer. This buffer must be at least as large as the corresponding *length* parameter.

tot_is_length
A pointer to the location to store the total length of the interior style list in the workstation description table.

tot_hs_length
A pointer to the location to store the total length of the hatch style list in the workstation description table.

tot_re_length
A pointer to the location to store the total length of the reflectance equation list in the workstation description table.

tot_sh_length
A pointer to the location to store the total length of the interior shading method list in the workstation description table.

FORTRAN Input Parameters

WKTYPE
The workstation type in question.

ISTYLE The index of the entry desired from the interior style list.

IHATCH
The index of the entry desired from the hatch style list.

IREFEQ
The index of the entry desired from the reflectance equation list.

ISHADE
The index of the entry desired from the interior shading method list.

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

NSTYLE
The number of entries in the list of available interior styles.

STYLE The requested entry of the interior style list.

NHATCH
The number of entries in the list of available hatch styles.

HATCH The requested entry of the hatch style list.

NREFEQ
The number of entries in the list of available reflectance equations.

REFEQ The requested entry of the reflectance equation list.

NSHADE
The number of entries in the list of available interior shading methods.

SHADE The requested entry of the interior shading method list.

NPRE The number of predefined extended interior bundle table entries.

Execution

If the inquired information is available, the error indicator is returned as zero and the requested information is returned in the output parameters.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the error numbers below to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)

052 Ignoring function, workstation type not recognized by the implementation

051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type

062 Ignoring function, this information is not available for this MO workstation type

SEE ALSO

SET INTERIOR REPRESENTATION PLUS (3PP)
INQUIRE INTERIOR REPRESENTATION PLUS (3PP)
INQUIRE PREDEFINED INTERIOR REPRESENTATION PLUS (3PP)

NAME

INQUIRE INTERIOR REPRESENTATION PLUS – inquire for values of extended interior representation on workstation

SYNOPSIS

C Syntax

```
void
pinq_int_rep_plus ( ws, index, type, error_ind, rep )
Pint            ws;         workstation identifier
Pint            index;      interior index
Pinq_type       type;       type of returned value
Pint            *error_ind; OUT error indicator
Pint_bundle_plus *rep;      OUT interior representation
```

FORTRAN Syntax

SUBROUTINE pqeir (WKID, INDEX, TYPE, ERRIND, STYLE, ISTYLE, COLTYPE, ICOLR, COLR, REFEQ, SHADEM, AMCOEF, DICOEF, SPCOEF, SCOLM, ISCOLR, SCOLR, SPEXP, TRCOEF, APTYP, APVAL)

```
INTEGER  WKID       workstation identifier
INTEGER  INDEX      bundle index
INTEGER  TYPE       type of returned values (PSET, PREALI)
INTEGER  ERRIND     OUT error indicator
INTEGER  STYLE(2)   OUT interior styles
INTEGER  ISTYLE(2)  OUT interior style indexes
INTEGER  COLTYPE(2) OUT interior colour types
INTEGER  ICOLR(2)   OUT interior colour index
REAL     COLR(3,2)  OUT interior colour triples
INTEGER  REFEQ(2)   OUT reflectance equations
INTEGER  SHADEM(2)  OUT shading methods
REAL     AMCOEF(2)  OUT ambient reflectance coefficients
REAL     DICOEF(2)  OUT diffuse reflectance coefficients
REAL     SPCOEF(2)  OUT specular reflectance coefficients
INTEGER  SCOLM(2)   OUT specular colour models
INTEGER  ISCOLR(2)  OUT specular colour indexes
REAL     SCOLR(3,2) OUT specular colour triples
REAL     SPEXP(2)   OUT specular exponents
REAL     TRCOEF(2)  OUT transparency coefficients
INTEGER  APTYP      OUT surface approximation type
REAL     APVAL(2)   OUT surface approximation values, u and v
```

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

INQUIRE INTERIOR REPRESENTATION PLUS returns the values of a specified entry in a workstation extended interior bundle table.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

ws The workstation identifier.

index Index of the entry to be returned. If this entry is not present in the table, and the type of returned value parameter is PINQ_REALIZED, the representation for interior index one is returned.

type An enumerated value specifying whether the values to be returned are those originally specified by the application (PINQ_SET), or those resulting after PHIGS mapped them to ones available on the workstation (PINQ_REALIZED). A Pinq_type structure is defined as:

```
typedef enum {
        PINQ_SET,
        PINQ_REALIZED
} Pinq_type;
```

C Output Parameters

error_ind
A pointer to the location to store the error number of any error detected by this function.

rep A pointer to a Pint_bundle_plus structure that returns the specified extended interior representation. Pint_bundle_plus is defined as:

```
typedef struct {
        Pint_style      style;          /* interior style */
        Pint            style_ind;      /* interior style index */
        Pgcolr          colr;           /* interior colour */
        Pint            refl_eqn;       /* reflectance equation */
        Pint            shad_meth;      /* shading method */
        Prefl_props     refl_props;     /* area properties */
        Pint_style      back_style;     /* interior style */
        Pint            back_style_ind; /* interior style index */
        Pgcolr          back_colr;      /* interior colour */
        Pint            back_refl_eqn;  /* back reflectance equation */
        Pint            back_shad_meth; /* back shading method */
        Prefl_props     back_refl_props; /* back area properties */
        Pint            approx_type;    /* approximation method */
        Pfloat          approx_val[2];  /* approximation values, u and v */
} Pint_bundle_plus;
```

Pint_style is defined as:

```
typedef enum {
        PSTYLE_HOLLOW,
        PSTYLE_SOLID,
```

```
        PSTYLE_PAT,
        PSTYLE_HATCH,
        PSTYLE_EMPTY,
} Pint_style;
```

Pgcolr is defined as:

```
typedef struct {
    Pinttype;    /* indirect, RGB, CIE, HSV, HLS */
    union {
        Pint    ind;    /* index in workstation colour bundle table */
        struct {
                Pfloat  x;      /* red, hue, and so on */
                Pfloat  y;      /* green, saturation, lightness, and so on */
                Pfloat  z;      /* blue, value, saturation, and so on */
        } general;
    } val;
} Pgcolr;
```

Prefl_props is defined as:

```
typedef struct {
        Pfloat  ambient_coef;       /* ambient reflectance coefficient */
        Pfloat  diffuse_coef;       /* diffuse reflectance coefficient */
        Pfloat  specular_coef;      /* specular reflectance coefficient */
        Pgcolr  specular_colr;      /* specular colour */
        Pfloat  specular_exp;       /* specular exponent */
        Pfloat  transpar_coef;      /* transparency coefficient */
} Prefl_props;
```

The values for ambient, diffuse, specular, and transparency coefficients must all be in the range [0,1]. The specular exponent must be greater than zero.

Constants defined for colour type are:

0	PINDIRECT	*Indirect*
1	PMODEL_RGB	*Red, Green, Blue*
2	PMODEL_CIELUV	*CIE*
3	PMODEL_HSV	*Hue, Saturation, Value*
4	PMODEL_HLS	*Hue, Lightness, Saturation*

The index member of the *val* union is used for type PINDIRECT. The general member is used for the other types.

Predefined reflectance equation values are:

1	PREFL_NONE	*No Reflectance Calculation Performed*
2	PREFL_AMBIENT	*Use Ambient Term*
3	PREFL_AMB_DIFF	*Use Ambient and Diffuse Terms*
4	PREFL_AMB_DIFF_SPEC	*Use Ambient, Diffuse, and Specular Terms*

Predefined constants for interior shading method are:

1	PSD_NONE	*No Shading*
2	PSD_COLOUR	*Colour Interpolation Shading*
3	PSD_DOT_PRODUCT	*Dot Product Interpolation Shading*
4	PSD_NORMAL	*Normal Interpolation Shading*

Constants defined for surface approximation type are:

1 PSURF_WS_DEP
2 PSURF_CONSTANT_PARAMETRIC_BETWEEN_KNOTS
3 PSURF_CHORDAL_SIZE_WC
4 PSURF_CHORDAL_SIZE_NPC
5 PSURF_CHORDAL_SIZE_DC
6 PSURF_PLANAR_DEVIATION_WC
7 PSURF_PLANAR_DEVIATION_NPC
8 PSURF_PLANAR_DEVIATION_DC

For a description of their use, see SET SURFACE APPROXIMATION CRITERIA.

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

WKID Workstation identifier.

INDEX Index of the entry to be returned. If this entry is not present in the table and the type of returned value parameter is PREALI, then the representation for interior index one is returned.

TYPE An enumerated value specifying whether the values to be returned are those originally specified by the application (*Set*), or those resulting after PHIGS mapped them to ones available on the workstation (*Realized*). Valid values are:

0	PSET	*Set*
1	PREALI	*Realized*

FORTRAN Output Parameters

ERRIND

The error number of any error detected by this function.

STYLE(2)

The front and back interior styles. Valid values are defined as:

0	PHOLLO	*Hollow*
1	PSOLID	*Solid*
2	PPATTR	*Patterned*
3	PHATCH	*Hatched*
4	PISEMP	*Empty*
5	PGENER	*General*

ISTYLE(2)

The front and back interior style indexes, which select entries in the workstation *Pattern*, *Hatch*, or *General* interior style tables.

COLTYPE(2)

The front and back interior colour types. Valid values are defined as:

0	PINDIR	*Indirect*
1	PRGB	*Red, Green, Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, Value*
4	PHLS	*Hue, Lightness, Saturation*

ICOLR(2)

The front and back colour indexes. ICOLR is used only when the corresponding value in COLTYPE is PINDIR.

COLR(3,2)

The front and back colour triples. COLR(3,*) is used only when the corresponding value in COLTYPE is not PINDIR. The triples are stored as:

ICOLR(1,c) = *x Colour*
ICOLR(2,c) = *y Colour*
ICOLR(3,c) = *z Colour*

REFEQ(2)

The front and back reflectance equations. Predefined reflectance equation values are:

1	PRNONE	*No Reflectance Calculation Performed*
2	PRAMB	*Use Ambient Term*
3	PRAMBD	*Use Ambient and Diffuse Terms*
4	PRAMDS	*Use Ambient, Diffuse, and Specular Terms*

SHADEM(2)

The front and back interior shading methods. Predefined constants are:

1	PSDNON	*No Shading*
2	PSDCOL	*Colour Interpolation Shading*
3	PSDDOT	*Dot Product Interpolation Shading*
4	PSDNOR	*Normal Interpolation Shading*

AMCOEF(2)

The front and back ambient reflectance coefficients, in the range [0,1].

DICOEF(2)

The front and back diffuse reflectance coefficients, in the range [0,1].

SPCOEF(2)

The front and back specular reflectance coefficients, in the range [0,1].

SCOLM(2)

The front and back specular colour models.

ISCOLR(2)

The front and back specular colour indexes. *ISCOLR(2)* is used only when the corresponding value in SCOLM PINDIR.

SCOLR(3,2)

The front and back specular colour triples. *SCOLR(3,2)* is used only when the corresponding value in SCOLM is not PINDIR.

SPEXP(2)
The front and back specular exponents, greater than zero.

TRCOEF(2)
The front and back transparency coefficients, in the range [0,1].

APTYP The surface approximation type. Predefined values are:

1	PSAWSD	*Workstation-dependent*
2	PSAEPK	*Constant Parametric Subdivision Between Knots*
3	PSACSW	*Chordal Size in WC*
4	PSACSN	*Chordal Size in NPC*
5	PSACSD	*Chordal Size in DC*
6	PSAPDW	*Planar Deviation in WC*
7	PSAPDN	*Planar Deviation in NPC*
8	PSAPDD	*Planar Deviation in DC*

For a description of their use, see SET SURFACE APPROXIMATION CRITERIA.

APVAL(2)
The surface approximation values. For a description of their use, see SET SURFACE APPROXIMATION CRITERIA.

Execution

If the inquired information is available, the error indicator is returned as zero and the requested information is returned in the output parameters.

If the specified interior index is not present in the interior bundle table on the workstation and the specified type of returned value is *Realized*, the representation for interior index one is returned.

If the inquired information is not available, the values returned in the output parameters are undefined and the error indicator is set to one of the below error numbers to indicate the reason for nonavailability.

The extended interior bundle table contains bundled entries of the PHIGS and SunPHIGS Extension interior attributes. During traversal, the attribute values of these bundles are used when the corresponding aspect source flag in the workstation state list is set to BUNDLED.

The PHIGS interior bundle table is a portion of the extended table. The INQUIRE INTERIOR REPRESENTATION function returns the following attributes from the extended table: interior style, interior style index, and colour index.

Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, the specified workstation is not open

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

100 Ignoring function, the bundle index value is less than one

101 Ignoring function, the specified representation has not been defined

SEE ALSO

SET INTERIOR REPRESENTATION PLUS (3PP)
INQUIRE INTERIOR REPRESENTATION (3P)
INQUIRE PREDEFINED INTERIOR REPRESENTATION PLUS (3PP)

NAME

INQUIRE LIGHT SOURCE FACILITIES – inquire for list of workstation light source facilities for specified workstation type

SYNOPSIS

C Syntax

void
pinq_light_src_facs (type, length, start, error_ind, facilities, total_length)

Pwstype	type;	*workstation type*
Pint	length;	*length of application list*
Pint	start;	*starting position*
Pint	*error_ind;	*OUT error indicator*
Plight_src_facs	*facilities;	*OUT light source facilities*
Pint	*total_length;	*OUT length of list in PHIGS*

FORTRAN Syntax

SUBROUTINE pqlsf (WKTYPE, ILS, ERRIND, NLSI, LSI, NPDEF)

INTEGER	WKTYPE	*workstation type*
INTEGER	ILS	*index into the light source list*
INTEGER	ERRIND	*OUT error indicator*
INTEGER	NLSI	*OUT number of light sources indices*
INTEGER	LSI	*OUT ILS-th element of light sources*
INTEGER	NPDEF	*OUT number of predefined indices*

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE LIGHT SOURCE FACILITIES inquires for a list of workstation light source facilities.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

type The workstation type for which the light source facilities are being inquired.

length The number of *ints* in the *facilities→types* output parameter for which the application has allocated memory. *length* is the number of list elements that the system can return in *facilites→types.ints*. If a value of 0 is used here, no data will be returned in the *facilites→types.ints* list, but the total number of elements will be returned in *total_length.*

start Starting position of inquiry. The elements in the list, beginning with the item number specified by *start,* are copied sequentially into *facilites→types.ints* until *facilites→types.ints* is full or all the elements have been copied.

C Output Parameters

error_ind
A pointer to the location to store the error number.

facilities
A pointer to the location in which to return the light source facilities.

Plight_src_facs is defined as:

```
typedef struct {
        Pint_list   types;          /* types of light source supported */
        Pint        max;            /* maximum number of simultaneously
                                       active lights */
        Pint        num_pred_inds   /* number of predefined bundles */
} Plight_src_facs;
```

types is a pointer to the types of light sources that are supported:

1 PLIGHT_AMBIENT
2 PLIGHT_DIRECTIONAL
3 PLIGHT_POSITIONAL
4 PLIGHT_SPOT

Pint_list is defined in phigs.h as follows:

```
typedef struct {
        Pint        num_ints;       /* number of Pints in list */
        Pint        *ints;          /* list of integers */
} Pint_list;
```

The *num_ints* component specifies the number of elements in the list. The *ints* component is a pointer to a list *num_ints* long.

Prior to calling this function, the *ints* field of the Pint_list structure must contain a pointer to an application-supplied buffer. This buffer must be at least as large as the corresponding length parameter.

The light types supported are:

1	Ambient	*A light source that affects all surface primitives uniformly. Ambient light sources have only a color attribute.*
2	WCS_Vector	*A light source that is specified in world coordinates with a color and a direction vector.*
3	WCS_Point	*A light source that is specified in world coordinates with a color, a position, and two attenuation coefficients.*
4	WCS_Spot	*A light source that is specified in world coordinates with a color, a position, a direction vector, a concentration exponent, two attenuation coefficients, and a spread angle.*

total_length

A pointer to an integer in which the system returns the total number of elements in the list. This is the value required for *length* if all elements in the list are to be returned.

FORTRAN Input Parameters

WKTYPE
The workstation type for which the light source facilities are being inquired.

ILS The index of the entry desired from the list of light source indices.

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

NLSI The number of light source types supported.

LSI The requested element from the defined list of light source indices.

NPDEF The number of predefined light source type indices.

Execution

If the inquired information is available, the error indicator is returned as zero, and values are returned in the output parameters.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the following error numbers to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)

051 Ignoring function, this information unavailable for this workstation type

052 Ignoring function, workstation type not recognized by the implementation

062 Ignoring function, this information is not available for this MO workstation type

SEE ALSO

SET LIGHT SOURCE REPRESENTATION (3PP)
INQUIRE LIST OF LIGHT SOURCE INDICES (3PP)
INQUIRE LIGHT SOURCE REPRESENTATION (3PP)
INQUIRE PREDEFINED LIGHT SOURCE REPRESENTATION (3PP)

NAME

INQUIRE LIGHT SOURCE REPRESENTATION – inquire for light source representation for specified workstation

SYNOPSIS

C Syntax

void
pinq_light_src_rep (ws, index, type, error_ind, rep)

Pint	ws;	*workstation identifier*
Pint	index;	*light source index*
Pinq_type	type;	*type of returned value*
Pint	*error_ind;	*OUT error indicator*
Plight_src_bundle	*rep;	*OUT light source representation*

FORTRAN Syntax

SUBROUTINE pqlsr (WKID, ILS, ITYPE, MLDR, ERRIND, LSTYPE, LDR, DATREC)

INTEGER	WKID	*workstation identifier*
INTEGER	ILS	*index into the light source list*
INTEGER	ITYPE	*type of returned values (PSET, PREALI)*
INTEGER	MLDR	*dimension of data record array*
INTEGER	ERRIND	*OUT error indicator*
INTEGER	LSTYPE	*OUT light source type*
INTEGER	LDR	*OUT actual size of the data record*
CHARACTER*80	DATREC(LDR)	*OUT data record*

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

INQUIRE LIGHT SOURCE REPRESENTATION inquires for a light source representation for a given workstation.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

ws The workstation identifier.

index The light source index.

type An enumerated value specifying whether the values to be returned are those originally specified by the application (PINQ_SET), or those resulting after PHIGS mapped them to ones available on the workstation (PINQ_REALIZED).

A Pinq_type structure is defined as:

```
typedef enum {
        PINQ_SET,
        PINQ_REALIZED
} Pinq_type;
```

C Output Parameters

error_ind

A pointer to the location to store the error number of any error detected by this function.

rep A pointer to the location in which to return the light source representation. The Plight_src_bundle data structure contains values needed to define the specified type of light source representation. Plight_src_bundle is defined as:

```
typedef struct {
        Pint            type;           /* light source type */
        Plight_src_rec  rec;            /* light source data record */
} Plight_src_bundle;
```

Plight_src_rec is defined as:

```
typedef union {
        Pamb_light_src_rec      ambient;
        Pdir_light_src_rec      directional;
        Ppos_light_src_rec      positional;
        Pspot_light_src_rec     spot;
} Plight_src_rec;
```

Pamb_light_src_rec is defined as:

```
typedef struct {
        Pgcolr  colr;   /* light source colour */
} Pamb_light_src_rec;
```

Pgcolr is defined as:

```
typedef struct {
    Pinttype;   /* indirect, RGB, CIE, HSV, HLS */
    union {
        Pint    ind;    /* index in workstation colour bundle table */
        struct {
                Pfloat  delta_x;  /* red, hue, and so on */
                Pfloat  delta_y;  /* green, saturation, lightness, and so on */
                Pfloat  delta_z;  /* blue, value, saturation, and so on */
        } general;
    } val;
} Pgcolr;
```

Pdir_light_src_rec is defined as:

```
typedef struct {
        Pgcolr  colr;   /* light source colour */
        Pvec3   dir;    /* light source direction */
} Pdir_light_src_rec;
```

Pvec3 is defined as:

```
typedef struct {
        Pfloat    delta_x;    /* x magnitude */
        Pfloat    delta_y;    /* y magnitude */
        Pfloat    delta_z;    /* z magnitude */
} Pvec3;
```

Ppos_light_src_rec is defined as:

```
typedef struct {
        Pgcolr      colr;       /* light source colour */
        Ppoint3     pos;        /* light source position */
        Pfloat      coef[2];    /* attenuation coefficients */
} Ppos_light_src_rec;
```

Ppoint3 is defined as:

```
typedef struct {
        Pfloat    x;        /* x coordinate */
        Pfloat    y;        /* y coordinate */
        Pfloat    z;        /* z coordinate */
} Ppoint3;
```

Pspot_light_src_rec is defined as:

```
typedef struct {
        Pgcolr      colr;       /* light source colour */
        Ppoint3     pos;        /* light source position */
        Pvec3       dir;        /* light source direction */
        Pfloat      exp;        /* concentration exponent */
        Pfloat      coef[2];    /* attenuation coefficients */
        Pfloat      angle;      /* half-cone spread angle */
} Pspot_light_src_rec;
```

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

WKID The identifier of the workstation for which the light source indices list is being inquired.

ILS An index into the light source table of the workstation state list.

ITYPE An enumerated value specifying whether the values to be returned are those originally specified by the application (*Set*), or those resulting after PHIGS mapped them to ones available on the workstation (*Realized*). Valid values are:

0	PSET	*Set*
1	PREALI	*Realized*

MLDR The dimension of the data record array, *DATREC*.

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

LSTYPE
The light source type.

LDR The actual size, in elements, returned in the data record.

DATREC(LDR)
The data record that must be unpacked by UNPACK DATA RECORD. The contents depend on the light source type. See SET LIGHT SOURCE REPRESENTATION for a description of its contents.

Execution

If the specified light source index is not present in the light source table on the workstation, and the specified type of returned value is *Realized*, the representation for light source index one is returned.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the following error numbers to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, the specified workstation is not open

059 Ignoring function, the specified workstation does not have output capability

101 Ignoring function, the specified representation has not been defined

614 Ignoring function, the light source index is less than one

SEE ALSO

SET LIGHT SOURCE STATE (3PP)
SET LIGHT SOURCE REPRESENTATION (3PP)
INQUIRE LIST OF LIGHT SOURCE INDICES (3PP)
INQUIRE LIGHT SOURCE FACILITIES (3PP)
INQUIRE PREDEFINED LIGHT SOURCE REPRESENTATION (3PP)

NAME

INQUIRE LIST OF COLOUR MAPPING INDICES – inquire for the list of defined colour mapping indices on a workstation

SYNOPSIS

C Syntax

void
pinq_list_colr_map_inds (ws, length, start, error_ind, colr_map_inds, total_length)

Pint	ws;	*workstation identifier*
Pint	length;	*length of application list*
Pint	start;	*starting position*
Pint	*error_ind;	*OUT error indicator*
Pint_list	*colr_map_inds;	*OUT list of polyline indices*
Pint	*total_length;	*OUT length of list in PHIGS*

FORTRAN Syntax

SUBROUTINE pqecmi (WKID, N, ERRIND, NCMI, CMI)

INTEGER	WKID	*workstation identifier*
INTEGER	N	*list element requested*
INTEGER	ERRIND	*OUT error indicator*
INTEGER	NCMI	*OUT number of colour mapping indices*
INTEGER	CMI	*OUT Nth element of list of colour mapping indices*

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

INQUIRE LIST OF COLOUR MAPPING INDICES inquires for a list of the currently defined workstation colour mapping indices.

C Input Parameters

ws The workstation identifier of the workstation whose colour mapping indices are to be returned.

length The number of *ints* in the *colr_map_inds* output parameter for which the application has allocated memory. *length* is the number of list elements that the system can return in *colr_map_inds→ints*. If a value of 0 is used here, then no data is returned in the *colr_map_inds→ints* list, but the total number of elements is returned in *total_length*.

start Starting position of inquiry. The elements in the list, beginning with the item number specified by *start*, are copied sequentially into *colr_map_inds→ints* until *colr_map_inds→ints* is full or all the elements have been copied.

C Output Parameters

error_ind
A pointer to the location to store the error number of any error detected by this function.

colr_map_inds
A pointer to a Pint_list data structure containing the list of colour mapping

indices. A Pint_list structure is defined as:

```
typedef struct {
        Pint    num_ints;    /* number of Pints in list */
        Pint    *ints;       /* list of integers */
} Pint_list;
```

The *num_ints* component specifies the number of elements in the list. The *ints* component is a pointer to a list *num_ints* long.

The application must allocate memory for *length* elements in the list of *ints*.

total_length
A pointer to an integer, which returns the total length of the list. This is the value required for *length* if all items in the list are to be returned.

FORTRAN Input Parameters

WKID The workstation identifier of the workstation whose colour mapping indices are to be returned.

N The index of the element requested from the list of colour mapping indices. If a value of 0 is used, then no data is returned in CMI, but the number of available indices is returned in NCMI.

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

NCMI The number of colour mapping indices defined on the specified workstation.

CMI The Nth element of the list of colour mapping indices.

Execution

If the inquired information is available, the error indicator is returned as zero and values are returned in the output parameters.

If the inquired information is not available, the values returned in the output parameters are undefined and the error indicator is set to one of the following error numbers to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, the specified workstation is not open

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT or OUTIN)

SEE ALSO

SET COLOUR MAPPING REPRESENTATION (3PP)
SET COLOUR MAPPING INDEX (3PP)
INQUIRE COLOUR MAPPING REPRESENTATION (3PP)

NAME

INQUIRE LIST OF DEPTH CUE INDICES – inquire for the list of depth cue indices defined on a workstation

SYNOPSIS

C Syntax

```
void
pinq_list_dcue_inds ( ws, length, start, error_ind, indices, total_length )
Pint        ws;             workstation identifier
Pint        length;         length of application list
Pint        start;          starting position
Pint        *error_ind;     OUT error indicator
Pint_list   *indices;       OUT list of polyline indices
Pint        *total_length;  OUT length of list in PHIGS
```

FORTRAN Syntax

```
SUBROUTINE pqdci ( WKID, N, ERRIND, NDCI, DCI )
INTEGER   WKID    workstation identifier
INTEGER   N       list element requested
INTEGER   ERRIND  OUT error indicator
INTEGER   NDCI    OUT number of depth cue indices
INTEGER   DCI     OUT Nth depth cue index
```

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

INQUIRE LIST OF DEPTH CUE INDICES inquire for a list of the currently-defined workstation depth cue indices. Objects are depth cued toward a specified depth cue colour. Depth cueing applies to all primitives.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

ws The workstation identifier.

length The number of *ints* in the *indices* output parameter for which the application has allocated memory. *length* is the number of list elements that the system can return in *indices→ints*. If a value of 0 is used here, then no data is returned in the *indices→ints* list, but the total number of elements is returned in *total_length*.

start Starting position of inquiry. The elements in the list, beginning with the item number specified by *start*, are copied sequentially into *indices→ints* until *indices→ints* is full or all the elements have been copied.

C Output Parameters

error_ind
A pointer to the location in which to store the error number of any error that this function detects.

indices A pointer to a Pint_list data structure containing the list of depth cue indices. A

Pint_list structure is defined as:

```
typedef struct {
        Pint    num_ints;    /* number of Pints in list */
        Pint    *ints;       /* list of integers */
} Pint_list;
```

The *num_ints* component specifies the number of elements in the list. The *ints* component is a pointer to a list *num_ints* long.

The application must allocate memory for *length* elements in the list of *ints*.

total_length
A pointer to an integer in which the system returns the total number of elements in the list. This is the value required for *length* if all elements in the list are to be returned.

FORTRAN Input Parameters

WKID The identifier of the workstation for which the list of depth cue indices is desired.

N The requested element from the list of depth cue indices.

FORTRAN Output Parameters

ERRIND
A pointer to the location in which to store the error number of any error that this function detects.

NDCI The number of depth cue indices in the list.

DCI The Nth element of the list of depth cue indices.

Execution

If the inquired information is available, then the error indicator is returned as zero and values are returned in the output parameters.

If the inquired information is not available, then the values returned in the output parameters are undefined and the error indicator is set to one of the following error numbers to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, the specified workstation is not open

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

SEE ALSO

SET DEPTH CUE REPRESENTATION (3PP)
SET DEPTH CUE INDEX (3PP)
INQUIRE DEPTH CUE REPRESENTATION (3PP)

NAME

INQUIRE LIST OF LIGHT SOURCE INDICES – inquire for list of light source indices defined on workstation

SYNOPSIS

C Syntax

```
void
pinq_list_light_src_inds ( ws, length, start, error_ind, indices, total_length )
Pint        ws;             workstation identifier
Pint        length;         length of application list
Pint        start;          starting position
Pint        *error_ind;     OUT error indicator
Pint_list   *indices;       OUT list of polyline indices
Pint        *total_length;  OUT length of list in PHIGS
```

FORTRAN Syntax

```
SUBROUTINE pqlsi ( WKID, ILS, ERRIND, NLSI, LSI )
INTEGER   WKID    workstation identifier
INTEGER   ILS     index into the light source list
INTEGER   ERRIND  OUT error indicator
INTEGER   NLSI    OUT number of light sources indices
INTEGER   LSI     OUT ILS-th element of light sources
```

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

INQUIRE LIST OF LIGHT SOURCE INDICES inquires for a list of light source indices defined on a workstation.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

ws The workstation identifier.

length The number of *ints* in the *indices* output parameter for which the application has allocated memory. *length* is the number of list elements that the system can return in *indices→ints*. If a value of 0 is used here, then no data is returned in the *indices→ints* list, but the total number of elements is returned in *total_length*.

start Starting position of inquiry. The elements in the list, beginning with the item number specified by *start*, are copied sequentially into *indices→ints* until *indices→ints* is full or all the elements have been copied.

C Output Parameters

error_ind
A pointer to the location to store the error number of any error detected by this function.

indices This is a pointer to a Pint_list structure. This structure specifies both an integer value indicating the number of light source representations and a pointer to a list

specifying the index numbers of the light source representations. Pint_list is defined in phigs.h as follows:

```
typedef struct {
        Pint    num_ints;    /* number of light sources */
        Pint    *ints;       /* list of light source indices */
} Pint_list;
```

The *num_ints* component specifies the number of elements in the list. The *ints* component is a pointer to a list *num_ints* long.

The application must allocate memory for *length* elements in the list of *ints*.

total_length
A pointer to an integer, which returns the total number of elements in the list of light source indices. This is the value required for *length* if all elements in the list are to be returned.

FORTRAN Input Parameters

WKID The workstation identifier.

ILS The index of the entry desired from the list of light source indices.

FORTRAN Output Parameters

ERRIND
A pointer to the location in which to store the error number of any error that this function detects

NLSI The number of indices in the light source list.

LSI The requested element from the list of defined light source indices.

Execution

If the inquired information is available, then the error indicator is returned as zero and values are returned in the output parameters.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the following error numbers to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, the specified workstation is not open

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

SEE ALSO

INQUIRE LIGHT SOURCE REPRESENTATION (3PP)
SET LIGHT SOURCE REPRESENTATION (3PP)

NAME

INQUIRE PATTERN REPRESENTATION PLUS – inquire for extended pattern representation on workstation

C Syntax

void
pinq_pat_rep_plus (ws, index, type, store, error_ind, rep)

Pint	ws;	*workstation identifier*
Pint	index;	*pattern index*
Pinq_type	type;	*type of returned value*
Pstore	store;	*OUT pointer to buffer*
Pint	*error_ind;	*OUT error indicator*
Ppat_rep_plus	**rep;	*OUT pattern representation*

FORTRAN Syntax

SUBROUTINE pqepr (WKID, INDEX, ITYPE, PSIZ, ERRIND, PBUF, ACTSIZ, PROWS, PCOLS, COLMOD, IPCOLR, PCOLR)

INTEGER	WKID	*workstation identifier*
INTEGER	INDEX	*pattern index*
INTEGER	ITYPE	*type of returned values (PSET, PREALI)*
INTEGER	PSIZ	*buffer size dimension*
INTEGER	ERRIND	*OUT error indicator*
CHARACTER*80	PBUF(PSIZ)	*OUT pattern buffer*
INTEGER	ACTSIZ	*OUT actual buffer size*
INTEGER	PROWS	*OUT pattern dimensions along y-axis*
INTEGER	PCOLS	*OUT pattern dimensions along x-axis*
INTEGER	COLMOD	*OUT colour model*
INTEGER	IPCOLR	*OUT indirect colour*
REAL	PCOLR(3)	*OUT array of colour*

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

INQUIRE PATTERN REPRESENTATION PLUS inquires for an extended pattern representation on a workstation.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the *store* argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area so that there is sufficient memory for the specific inquiry. The data record within the store buffer is accessed by the pointer pointed to by *rep*.

ws The workstation identifier.

index The pattern index.

type An enumerated value specifying whether the values to be returned are those originally specified by the application (PINQ_SET), or those resulting after PHIGS mapped them to ones available on the workstation (PINQ_REALIZED). A Pinq_type structure is defined as:

```
typedef enum {
        PINQ_SET,
        PINQ_REALIZED
} Pinqtype;
```

store The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

C Output Parameters

error_ind
A pointer to the location to store the error number of any error detected by this function.

rep A pointer to a pointer to a Ppat_rep_plus structure, which returns the specified extended pattern representation. Ppat_rep_plus is defined as:

```
typedef struct {
        Pint_size   dims;       /* pattern's dimensions */
        Pint        type;       /* colour model */
        Pcoval      *colr_array; /* array of colours */
} Ppat_rep_plus;
```

See SET PATTERN REPRESENTATION PLUS for a full description of the contents of this structure.

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

WKID The identifier of the workstation for which the extended pattern representation is being inquired.

INDEX The index into the specified workstation pattern bundle table.

ITYPE An enumerated value specifying whether the values to be returned are those originally specified by the application (*Set*), or those resulting after PHIGS mapped them to ones available on the workstation (*Realized*). Valid values are:

0	PSET	*Set*
1	PREALI	*Realized*

PSIZ The size, in number of bytes, of the application-supplied buffer in which the requested pattern representation will be stored.

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

PBUF A character array in which to return the requested pattern representation.

ACTSIZ The actual size, in bytes, filled in the pattern buffer PBUF.

PROWS The pattern dimensions along the y-axis.

PCOLS The pattern dimensions along the x-axis.

COLMOD
The colour model.

IPCOLR
An integer containing the index into the workstation colour table for the pattern colour.

PCOLR(3)
An array of reals containing the pattern colour.

Execution

If the inquired information is available, the error indicator is returned as zero and values are returned in the output parameters.

If the specified pattern index is not present in the pattern bundle table on the workstation and the specified type of returned value is *Realized*, the representation for pattern index one is returned.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the following error numbers to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, the specified workstation is not open

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

101 Ignoring function, the specified representation has not been defined

109 Ignoring function, interior style PATTERN is not supported on the workstation

112 Ignoring function, the pattern index value is less than one

SEE ALSO

SET PATTERN REPRESENTATION PLUS (3PP)
INQUIRE PATTERN REPRESENTATION (3P)
INQUIRE PREDEFINED PATTERN REPRESENTATION PLUS (3PP)

NAME

INQUIRE POLYLINE FACILITIES PLUS – inquire for workstation type extended polyline facilities

SYNOPSIS

C Syntax

void
pinq_line_facs_plus (type, lt_length, lt_start, sm_length, sm_start, error_ind, tot_lt_length, tot_sm_length, facilities)

Pint	type;	*workstation type*
Pint	lt_length;	*number of line types to return*
Pint	lt_start;	*starting position*
Pint	sm_length;	*number of shading methods to return*
Pint	sm_start;	*starting position*
Pint	*error_ind;	*OUT error indicator*
Pint	*tot_lt_length;	*OUT length of line type list in PHIGS*
Pint	*tot_sm_length;	*OUT length of shading method list in PHIGS*
Pline_facs_plus	*facilities;	*OUT extended polyline facilities*

FORTRAN Syntax

SUBROUTINE pqeplf (WKTYPE, ILT, ISM, ERRIND, NLT, LT, NLW, NOMLW, MINLW, MAXLW,NSM, SM, NPPLI)

INTEGER	WKTYPE	*workstation type*
INTEGER	ILT	*index into linetype list*
INTEGER	ISM	*index into shading method list*
INTEGER	ERRIND	*OUT error indicator*
INTEGER	NLT	*OUT number of available linetypes*
INTEGER	LT	*OUT ILT entry from linetype list*
INTEGER	NLW	*OUT number of available linewidths*
REAL	NOMLW	*OUT nominal linewidth (DC)*
REAL	MINLW, MAXLW	*OUT range of linewidths (DC)*
INTEGER	NSM	*OUT number of available shading methods*
INTEGER	SM	*OUT ISM entry from shading methods list*
INTEGER	NPPLI	*OUT number of predefined polyline indices*

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE POLYLINE FACILITIES PLUS returns the extended polyline support available from a specified workstation type.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

type Get the polyline facilities for this workstation type.

lt_length

The number of *ints* in the *facilities→types.ints* output parameter for which the

application has allocated memory. *length* is the number of list elements that the system can return in *facilities→types.ints*. If a value of 0 is used here, then no data is returned in the *facilities→types.ints* list, but the total number of elements is returned in *tot_lt_length*.

lt_start Starting position of inquiry. The elements in the list, beginning with the item number specified by *start*, are copied sequentially into *facilities→types.ints* until *facilities→types.ints* is full or all the elements have been copied.

sm_length
The number of *ints* in the *facilities→shads.ints* output parameter for which the application has allocated memory. *length* is the number of list elements that the system can return in *facilities→shads.ints*. If a value of 0 is used here, then no data is returned in the *facilities→shads.ints* list, but the total number of elements is returned in *tot_sm_length*.

sm_start
Starting position of inquiry. The elements in the list, beginning with the item number specified by *sm_start*, are copied sequentially into *facilities→shads.ints* until *facilities→shads.ints* is full or all the elements have been copied.

C Output Parameters

error_ind
A pointer to the location to store the error number of any error detected by this function.

tot_lt_length
A pointer to an integer in which to return the total length of the line type list. This is the value required for *lt_length* if all the items in the list are to be returned.

tot_sm_length
A pointer to an integer in which to return the total length of the shading method list. This is the value required for *sm_length* if all the items in the list are to be returned.

facilities
A pointer to a structure in which the system returns the portion of the list of polyline facilities from the workstation description table, starting with start. Pline_facs_plus is defined in phigs.h as:

```
typedef struct {
    Pint_list   types;          /* list of line types */
    Pint        widths;         /* number of available line widths */
    Pint        num_widths;     /* number of available line widths */
    Pfloat      nom_width;      /* nominal line width */
    Pfloat      min_width;      /* minimum line width */
    Pfloat      max_width;      /* maximum line width */
    Pint        num_pred_inds;  /* number of predefined bundles */
    Pint_list   shads;          /* list of shading methods */
} Pline_facs_plus;
```

And Pint_list is defined in phigs.h as:

```
typedef struct {
        Pint        num_ints;       /* number of integers */
        Pint        *ints;          /* list of integers */
} Pint_list;
```

Prior to calling this function, the *ints* field of the Pint_list structure must contain a pointer to an application-supplied buffer. This buffer must be at least as large as the corresponding *length* parameter.

FORTRAN Input Parameters

WKTYPE
The workstation type.

ILT The index of the entry desired from the list of available linetypes.

ISM The index of the entry desired from the list of available shading methods.

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

NLT The number of entries in the list of available linetypes.

LT The requested entry of the available linetype list.

NLW The number of available linewidths.

NOMLW
The nominal linewidth in Device Coordinates (DC).

MINLW The minimum linewidth in DC.

MAXLW
The maximum linewidth in DC.

NSM The number of entries in the list of available shading methods.

SM The requested entry of the available shading method list. Defined shading methods are:

1 PSDNON *No Shading*
2 PSDCOL *Colour Interpolation Shading*

NPPLI The number of predefined polyline indices.

Execution

If the inquired information is available, the error indicator is returned as zero and the requested information is returned in the output parameters.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the error numbers below to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)

052 Ignoring function, workstation type not recognized by the implementation

051 Ignoring function, this information is not yet available for this workstation type; open a workstation of this type and use the specific workstation type

062 Ignoring function, this information is not available for this MO workstation type

SEE ALSO

SET POLYLINE REPRESENTATION PLUS (3PP)
INQUIRE POLYLINE REPRESENTATION PLUS (3PP)
INQUIRE PREDEFINED POLYLINE REPRESENTATION PLUS (3PP)

NAME

INQUIRE POLYLINE REPRESENTATION PLUS – inquire for values of workstation's extended polyline representation

SYNOPSIS

C Syntax

```
void
pinq_line_rep_plus ( ws, index, type, error_ind, rep )
Pint                ws;          workstation identifier
Pint                index;       polyline index
Pinq_type           type;        type of returned value
Pint                *error_ind;  OUT error indicator
Pline_bundle_plus   *rep;        OUT polyline representation
```

FORTRAN Syntax

```
SUBROUTINE pqeplr ( WKID, INDEX, TYPE, ERRIND, LTYPE, LWIDTH,
    COLMOD, ICOLR, COLR, METHOD, ATYPE, AVALUE )
INTEGER   WKID      workstation identifier
INTEGER   INDEX     polyline index
INTEGER   TYPE      type of returned values (PSET, PREALI)
INTEGER   ERRIND    OUT error indicator
INTEGER   LTYPE     OUT linetype
REAL      LWIDTH    OUT linewidth scale factor
INTEGER   COLMOD    OUT colour model
INTEGER   ICOLR     OUT polyline colour index
REAL      COLR(3)   OUT colour values
INTEGER   METHOD    OUT polyline shading method
INTEGER   ATYPE     OUT curve approximation type
REAL      AVALUE    OUT curve approximation value
```

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

INQUIRE POLYLINE REPRESENTATION PLUS returns the attribute values of a specified entry in a workstation extended polyline bundle table. See SET POLYLINE REPRESENTATION PLUS for a full description of the contents of the bundle table entry and a description of the extended polyline bundle table.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

ws Workstation identifier.

index Index of the entry to be returned from the workstation extended polyline bundle table. If this entry is not present in the table, and the type of returned value parameter is PINQ_REALIZED, the representation for polyline index one is returned.

type An enumerated value specifying whether the values to be returned are those originally specified by the application (PINQ_SET), or those resulting after PHIGS mapped them to ones available on the workstation (PINQ_REALIZED). A Pinq_type structure is defined as:

```
typedef enum {
        PINQ_SET,
        PINQ_REALIZED
} Pinq_type;
```

C Output Parameters

error_ind
A pointer to the location to store the error number of any error detected by this function.

rep A pointer to a Pline_bundle_plus structure that returns the specified extended polyline representation. Pline_bundle_plus is defined as:

```
typedef struct {
        Pint    type;           /* line type */
        Pfloat  width;          /* linewidth scale factor */
        Pgcolr  colr;           /* polyline colour */
        Pint    shad_meth;      /* polyline shading method */
        Pint    approx_type;    /* curve approximation type */
        Pfloat  approx_val;     /* curve approximation value */
} Pline_bundle_plus;
```

See SET POLYLINE REPRESENTATION PLUS for a full description of the contents of this structure.

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

WKID Workstation identifier.

INDEX Index of the entry to be returned from the workstation extended polyline bundle table. If this entry is not present in the table and the type of returned value parameter is PREALI, then representation for polyline index one is returned.

TYPE An enumerated value specifying whether the values to be returned are those originally specified by the application (*Set*), or those resulting after PHIGS mapped them to ones available on the workstation (*Realized*). Valid values are:

0	PSET	*Set*
1	PREALI	*Realized*

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

LTYPE The line type.

LWIDTH
The linewidth scale factor.

COLMOD
The colour model. The predefined values are:

0	PINDIR	*Colour Index Specified*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

ICOLR An integer value which is an index into the workstation colour table for determining the polyline colour. This value is not used if *COLMOD* is greater than zero.

COLR(3)
An array of reals containing the colours for the polyline. This array is unused if *COLMOD* is equal to zero (PINDIR).

COLR(1) = *Red, Hue, and so on*
COLR(2) = *Green, Saturation, Lightness, and so on*
COLR(3) = *Blue, Value, Saturation, and so on*

METHOD
The polyline shading method.

ATYPE The curve approximation type.

AVALUE
The curve approximation value.

Execution

If the inquired information is available, then the error indicator is returned as zero, and the requested information is returned in the output parameters.

If the specified polyline index is not present in the polyline bundle table on the workstation, and the specified type of returned value is *Realized*, then the representation for polyline index one is returned.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the error numbers below to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

ERRORS

003	Ignoring function, function requires state (PHOP, WSOP, *, *)
054	Ignoring function, the specified workstation is not open
059	Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
100	Ignoring function, the bundle index value is less than one
101	Ignoring function, the specified representation has not been defined

SEE ALSO

SET POLYLINE REPRESENTATION PLUS (3PP)
INQUIRE POLYLINE REPRESENTATION (3P)
INQUIRE PREDEFINED POLYLINE REPRESENTATION PLUS (3PP)

NAME

INQUIRE POLYMARKER REPRESENTATION PLUS – inquire for extended polymarker representation on workstation

SYNOPSIS

C Syntax

```
void
pinq_marker_rep_plus ( wsid, index, type, error_ind, rep )
Pint                  wsid;       workstation identifier
Pint                  index;      polymarker index
Pinq_type             type;       type of returned value
Pint                  *error_ind; OUT error indicator
Pmarker_bundle_plus   *rep;       OUT polymarker representation
```

FORTRAN Syntax

```
SUBROUTINE pqepmr ( WKID, INDEX, ITYPE, ERRIND, MTYPE, MSIZE,
    COLMOD, IMCOLR, MCOLR )
INTEGER   WKID      workstation identifier
INTEGER   INDEX     polymarker bundle index
INTEGER   ITYPE     type of returned values (PSET, PREALI)
INTEGER   ERRIND    OUT error indicator
INTEGER   MTYPE     OUT polymarker type
REAL      MSIZE     OUT polymarker size scale factor
INTEGER   COLMOD    OUT colour model
INTEGER   IMCOLR    OUT indirect colour index
REAL      MCOLR(3)  OUT direct colour value
```

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

INQUIRE POLYMARKER REPRESENTATION PLUS returns the attribute values of a specified entry in a workstation extended polymarker bundle table.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

wsid Workstation identifier.

index Index of the entry to be returned from the workstation extended polymarker bundle table. If this entry is not present in the table, and the type of returned value parameter is PINQ_REALIZED, then the representation for polymarker index one is returned.

type An enumerated value specifying whether the values to be returned are those originally specified by the application (PINQ_SET), or those resulting after PHIGS mapped them to ones available on the workstation (PINQ_REALIZED). A Pinq_type structure is defined as:

```
typedef enum {
```

```
        PINQ_SET,
        PINQ_REALIZED
    } Pinq_type;
```

C Output Parameters

error_ind
A pointer to the location to store the error number of any error that this function detects.

rep A pointer to a Pmarker_bundle_plus structure, which returns the specified extended polymarker representation. Pmarker_bundle_plus is defined as:

```
typedef struct {
        Pint    type;   /* marker type */
        Pfloat  size;   /* marker size scale factor */
        Pgcolr  colr;   /* marker colour */
} Pmarker_bundle_plus;
```

Pgcolr is defined as:

```
typedef struct {
    Pint    type;   /* indirect, RGB, CIE, HSV, HLS */
    union {
        Pint  ind;  /* index in workstation colour bundle table */
        struct {
            Pfloat  x;      /* red, hue, and so on */
            Pfloat  y;      /* green, saturation, lightness, and so on */
            Pfloat  z;      /* blue, value, saturation, and so on */
        } general;
    } val;
} Pgcolr;
```

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

WKID The identifier of the workstation for which the polymarker representation is being inquired.

INDEX Index of the entry to be returned from the workstation extended polymarker bundle table. If this entry is not present in the table and the type of returned value parameter is PREALI, then representation for polymarker index one is returned.

ITYPE An enumerated value specifying whether the values to be returned are those originally specified by the application (*Set*), or those resulting after PHIGS mapped them to ones available on the workstation (*Realized*). Valid values are:

0	PSET	*Set*
1	PREALI	*Realized*

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

MTYPE The polymarker type.

MSIZE The polymarker size scale factor.

COLMOD
The colour model. The predefined values are:

0	PINDIR	*Indirect*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

IMCOLR
An integer containing the index into the workstation colour table for the polymarker colour. This value is not used if *COLMOD* is greater than zero.

MCOLR(3)
An array of reals containing the polymarker colour. This array is unused if *COLMOD* is equal to zero (PINDIR).

MCOLR(1) = *Red, Hue, and so on*
MCOLR(2) = *Green, Saturation, Lightness, and so on*
MCOLR(3) = *Blue, Value, Saturation, and so on*

Execution

If the inquired information is available, then the error indicator is returned as zero and values are returned in the output parameters.

If the specified polymarker index is not present in the polymarker bundle table on the workstation, and the specified type of returned value is *Realized*, then the representation for polymarker index one is returned.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the error numbers below to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

ERRORS

003	Ignoring function, function requires state (PHOP, WSOP, *, *)
054	Ignoring function, the specified workstation is not open
059	Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT,, OUTIN, or MO)
100	Ignoring function, the bundle index value is less than one
101	Ignoring function, the specified representation has not been defined

SEE ALSO

SET POLYMARKER REPRESENTATION PLUS (3PP)
INQUIRE POLYMARKER REPRESENTATION (3P)
INQUIRE PREDEFINED POLYMARKER REPRESENTATION PLUS (3PP)

NAME

INQUIRE PREDEFINED COLOUR MAPPING REPRESENTATION – inquire for a predefined colour mapping representation for the specified workstation type

SYNOPSIS

C Syntax

void
pinq_pred_colr_map_rep (type, index, store, error_ind, map_method, map_data)

Pint	type;	*workstation type*
Pint	index;	*predefined index*
Pstore	store;	*handle to store data*
Pint	*error_ind;	*OUT error indicator*
Pint	*map_method	*OUT colour mapping method*
Pcolr_map_data	**map_data	*OUT colour mapping data*

FORTRAN Syntax

SUBROUTINE pqpcmr (WTYPE, CMRI, MLDR, ERRIND, MMETH, LDR, DATREC)

INTEGER	WTYPE	*workstation type*
INTEGER	CMRI	*predefined index*
INTEGER	MLDR	*dimension of data record array*
INTEGER	ERRIND	*OUT error indicator*
INTEGER	MMETH	*OUT colour mapping method*
INTEGER	LDR	*OUT number of array elements used in data record*
CHARACTER*80	DATREC(MLDR)	*OUT data record*

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE PREDEFINED COLOUR MAPPING REPRESENTATION returns the values of a specified entry in the table of predefined colour mapping representations for the workstation type. Applications using the C binding must create a buffer for this function to use as a memory space for storing data associated with the device state. This buffer is passed as the *store* argument.

The store buffer is a data area that PHIGS manages. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area so that there is sufficient memory for the specific inquiry. The data record in the store buffer is accessed by the pointer pointed to by *map_data*.

type The type of workstation for which capabilities are inquired.

index Entry in the workstation table of predefined colour mapping representations to be returned.

store The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

C Output Parameters

error_ind

A pointer to the location in which to store the error number of any error that this function detects.

map_method

A pointer to a location in which to store the returned color mapping method.

map_data

A pointer to a pointer to a Pcolr_map_data structure, which contains information specific to the data mapping method. PHIGS allocates space for the structure in the Pstore object and returns the pointer to that allocated space in this parameter. Pcolr_map_data is defined in phigs.h as follows:

```
typedef union {
        struct {
                Pint            colr_model;
                Pfloat_list     weights;
                Pcolr_rep_list  colrs;
        } method_r2
        struct {
                Pint            colr_model;
                Pfloat_list_list colr_lists;
        } method_r3
} Pcolr_map_data;
```

Pfloat_list is defined in phigs.h as follows:

```
typedef struct {
        Pint            num_floats;     /* number of Pfloats in list */
        Pfloat          *floats;        /* list of floats *
} Pfloat_list;
```

Pfloat_list_list is defined in phigs.h as follows:

```
typedef struct {
        Pint            num_lists;      /* number of lists in list */
        Pfloat_list     *lists;         /* list of float lists */
} Pfloat_list_list;
```

Pcolr_rep_list is defined in phigs.h as follows:

```
typedef struct {
        Pint            num_colr_reps;  /* number of colours */
        Pcolr_rep       *colr_reps;     /* array of colours */
} Pcolr_rep_list;
```

Pcolr_rep is defined in phigs.h as follows:

```
typedef union {
```

```
        Prgb        rgb;        /* Red, Green, Blue colour specification */
        Pcieluv     cieluv;     /* CIE L*U*V* colour specification */
        Phls        hls;        /* Hue, Lightness, Saturation colour
                                   specification */
        Phsv        hsv;        /* Hue, Saturation, Value colour
                                   specification */
        Pdata       unsupp;     /* Colour in unsupported colour model */
} Pcolr_rep;
```

Prgb is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          red;            /* red, hue, and so on */
        Pfloat          green;          /* green, saturation, lightness,
                                           and so on */
        Pfloat          blue;           /* blue, value, saturation,
                                           and so on */
} Prgb;
```

Pcieluv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          cieluv_x;       /* x coefficient */
        Pfloat          cieluv_y;       /* y coefficient */
        Pfloat          cieluv_y_lum;   /* y luminance */
} Pcieluv;
```

Phsv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          satur;          /* saturation */
        Pfloat          value;          /* value */
} Phsv;
```

Phls is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          lightness;      /* lightness */
        Pfloat          satur;          /* saturation */
} Phls;
```

Pdata is defined in phigs.h as follows:

```
typedef struct {
        size_t          size;           /* size of data */
        char            *data           /*pointer to data */
} Pdata;
```

No data is associated with mapping method PCOLR_MAP_TRUE. The data for method PCOLR_MAP_PSEUDO is a list of weights and a list of direct colours. *colr_model* specifies the type of colours given in the list of colours, *colrs*. The colour models are defined in phigs.h as follows:

1 PMODEL_RGB
2 PMODEL_CIELUV
3 PMODEL_HSV
4 PMODEL_HLS

The data for mapping method PCOLR_MAP_PSEUDO_N is N lists of direct colours. These colours may be of the same types as for PCOLR_MAP_PSEUDO.

FORTRAN Input Parameters

Applications using the FORTRAN binding must supply a CHARACTER array to this function, into which the system will place the contents of the device's input data record. The contents of the data record are subsequently extracted by the application with the function UNPACK DATA RECORD (3P). The allocated dimension of the character array is passed in the MLDR argument. The dimension needed is returned in the LDR argument. The caller can determine the required dimension by calling this function with MLDR set to zero, in which case PHIGS returns the dimension needed in LDR.

Even if the dimension specified in MLDR is too small, including the case of its being zero, some values are returned. These are LDR, and the colour mapping method MMETH.

Error 2001 is returned when MLDR is too small; but not when it is zero.

WTYPE The type of workstation for which capabilities are inquired.

CMRI Entry in the workstation table of predefined colour mapping representations to be returned.

MLDR The dimension of the data record array, DATREC.

FORTRAN Output Parameters

ERRIND

The error number of any error detected by this function.

MMETH

The colour mapping method. This may be any of the following values defined in phigs77.h:

1 PCMAPTRUE
2 PCMAPPSEUDO
3 PCMAPPSEUDON

LDR The required dimension of the data record array, DATREC.

DATREC

The data record array. This must subsequently be passed to UNPACK DATA RECORD (3P) to access its content. The content of the data record is determined by the colour mapping method; see SET COLOUR MAPPING REPRESENTATION (3P) for a content description.

Execution

If the inquired information is available, then the error indicator is returned as zero and values are returned in the output parameters.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the following error numbers to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

ERRORS

002	Ignoring function, function requires state (PHOP, *, *, *)
051	Ignoring function, this information unavailable for this workstation type
052	Ignoring function, workstation type not recognized by the implementation
059	Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
062	Ignoring function, this information is not available for this MO workstation type
102	Ignoring function, specified representation has not been predefined on this workstation
121	Ignoring function, the colour mapping index is less than 0.

SEE ALSO

SET COLOUR MAPPING REPRESENTATION (3PP)
INQUIRE COLOUR MAPPING REPRESENTATION (3PP)
CREATE STORE (3P)
DELETE STORE (3P)
UNPACK DATA RECORD (3P)

NAME

INQUIRE PREDEFINED DEPTH CUE REPRESENTATION – inquire for predefined depth cue representation for specified workstation type

SYNOPSIS

C Syntax

```
void
pinq_pred_dcue_rep ( type, index, error_ind, bundle )
Pwstype         type;       workstation type
Pint            index;      predefined index
Pint            *error_ind; OUT error indicator
Pdcue_bundle    *bundle;    OUT predefined depth cue rep
```

FORTRAN Syntax

```
SUBROUTINE pqpdcr ( WKTYPE, PDCI, ERRIND, MODE, BPLANE, FPLANE,
    BSCALE, FSCALE, COLTYPE, IDCOLR, DCOLR )
INTEGER   WKTYPE          workstation type
INTEGER   PDCI            predefined depth cue index
INTEGER   ERRIND          OUT error indicator
INTEGER   MODE            OUT depth cue mode
REAL      BPLANE, FPLANE  OUT back and front depth planes
REAL      BSCALE, FSCALE  OUT back and front scale values
INTEGER   COLTYPE         OUT colour type
INTEGER   IDCOLR          OUT indirect depth cue colour
REAL      DCOLR(3)        OUT array of depth cue colour
```

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE PREDEFINED DEPTH CUE REPRESENTATION returns the attribute values of a specified entry in the table of predefined depth cue representations for that workstation type. See SET DEPTH CUE REPRESENTATION for a full description of the content of the bundle table entry and a description of the depth cue representation table.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

type The workstation type.

index The index of the predefined entry to be returned.

C Output Parameters

error_ind
A pointer to the location to store the error number of any error that this function detects.

bundle A pointer to a Pdcue_bundle data structure containing attribute values defining a predefined depth cue representation. A Pdcue_bundle structure is defined in phigs.h as follows:

```
typedef struct {
        Pdcue_mode      mode;           /* depth cue mode */
        Pfloat          ref_planes[2];  /* depth cue reference planes */
        Pfloat          scaling[2];     /* depth cue scaling */
        Pgcolr          colr;           /* depth cue colour */
} Pdcue_bundle;
```

Entry 0 in ref_planes is the back reference plane. Entry 0 in scaling is back scaling.

Pdcue_mode is defined as follows:

```
typedef enum {
        PSUPPRESSED,
        PALLOWED
} Pdcue_mode;
```

Pgcolr is defined as:

```
typedef struct {
        Pint    type;   /* indirect, RGB, CIE, HSV, HLS */
        union {
                Pint    ind;    /* index in workstation colour
                                   bundle table */
                struct {
                        Pfloat  x;      /* red, hue, and so on */
                        Pfloat  y;      /* green, saturation, lightness,
                                           and so on */
                        Pfloat  z;      /* blue, value, saturation,
                                           and so on */
                } general;
        } val;
} Pgcolr;
```

Constants defined for colour type are:

0	PINDIRECT	*Indirect*
1	PMODEL_RGB	*Red, Green, Blue*
2	PMODEL_CIELUV	*CIE*
3	PMODEL_HSV	*Hue, Saturation, Value*
4	PMODEL_HLS	*Hue, Lightness, Saturation*

The index member of the *val* union is used for type PINDIRECT. The general member is used for the other types.

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

WKTYPE

The workstation type.

PDCI

The predefined depth cue index.

FORTRAN Output Parameters

ERRIND

A pointer to the location in which to store the error number of any error that this function detects.

MODE

The depth cue MODE controls the depth cue disabling or enabling. The depth cue mode contains one of the following values:

0	PSUPPR	*Disable Depth Cueing*
1	PALLOW	*Enable Depth Cueing*

BPLANE,

The back and front depth planes. BPLANE is the back depth plane, a *z* value in NPC, that defines a back reference plane for depth cueing. FPLANE is the front depth plane, a *z* value in NPC, which defines a front reference plane for depth cueing.

BSCALE,

The back and front scale values. BSCALE is the back scale value, between 0.0 and 1.0, that corresponds to the *z* value of the back depth cue reference plane. This value defines the portion of the primitive and the depth cue colours that should be combined as a function of *z* in NPC. FSCALE is a front scale value, between 0.0 and 1.0, that corresponds to the *z* value of the front depth cue reference plane. This value defines the portion of the primitive and depth cue colours that should be combined as a function of *z* in NPC.

COLTYPE

The colour type. If this value is PINDIR or zero, *IDCOLR* will contain an indexed colour value. If *COLTYPE* is greater than zero, *DCOLR* will contain the depth cue colour. The valid values for *COLTYPE* are:

0	PINDIR	*Colour Index Specified*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

IDCOLR

An index into the workstation colour table. This integer value determines the depth cue colour. This value is not used if COLTYPE is greater than zero.

DCOLR(3)

An array of reals containing the depth cue colour. This array is not used if COLTYPE is equal to zero. The reals are stored as follows:

DCOLR(1) = *Red, Hue, and so on*
DCOLR(2) = *Green, Saturation, Lightness, and so on*
DCOLR(3) = *Blue, Value, Saturation, and so on*

Execution

If the inquired information is available, then the error indicator is returned as zero, and values are returned in the output parameters.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the error numbers below to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

ERRORS

002 Ignoring function, function requires state (PHOP *, *, *)

051 Ignoring function, this information unavailable for this workstation type

052 Ignoring function, workstation type not recognized by the implementation

062 Ignoring function, this information is not available for this MO workstation type

102 Ignoring function, specified representation has not been predefined on this workstation

119 Ignoring function, depth cue index is less than zero

SEE ALSO

SET DEPTH CUE REPRESENTATION (3PP)
INQUIRE DEPTH CUE REPRESENTATION (3PP)

NAME

INQUIRE PREDEFINED EDGE REPRESENTATION PLUS – inquire for predefined extended edge representation for specified workstation type

SYNOPSIS

C Syntax

```
void
pinq_pred_edge_rep_plus ( type, index, error_ind, bundle )
Pint                type;       workstation type
Pint                index;      predefined index
Pint                *error_ind; OUT error indicator
Pedge_bundle_plus   *bundle;    OUT predefined edge representation
```

FORTRAN Syntax

```
SUBROUTINE pqpeer ( WKTYPE, INDEX, ERRIND, EFLAG, ETYPE, EWIDTH,
   COLTYPE, IECOLR, ECOLR )
INTEGER  WKTYPE    workstation type
INTEGER  INDEX     predefined edge index
INTEGER  ERRIND    OUT error indicator
INTEGER  EFLAG     OUT edge flag (PON/POFF)
INTEGER  ETYPE     OUT edge type
REAL     EWIDTH    OUT edge width scale factor
INTEGER  COLTYPE   OUT colour type
INTEGER  IECOLR    OUT indirect colour
REAL     ECOLR(3)  OUT array of colour
```

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE PREDEFINED EDGE REPRESENTATION PLUS inquires for a predefined extended edge representation for a specified workstation type.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

type The workstation type.

index The index of the predefined entry to be returned.

C Output Parameters

error_ind

A pointer to the location to store the error number of any error detected by this function.

bundle

A pointer to a Pedge_bundle_plus structure that returns the requested representation. Pedge_bundle_plus is defined as:

```
typedef struct {
        Pedge_flag      flag;       /* edge flag */
```

```
        Pint        type;       /* edgetype */
        Pfloat      scale;      /* edgewidth scale factor */
        Pgcolr      colr;       /* edge colour */
} Pedge_bundle_plus;
```

Valid values for the Pedge_flag enumerated type are defined in phigs.h as:

```
typedef enum {
        PEDGE_OFF,
        PEDGE_ON
} Pedge_flag;
```

Pgcolr is defined as:

```
typedef struct {
    Pinttype;    /* indirect, RGB, CIE, HSV, HLS */
    union {
        Pint    ind;    /* index in workstation colour bundle table */
        struct {
                Pfloat  x;      /* red, hue, and so on */
                Pfloat  y;      /* green, saturation, lightness, and so on */
                Pfloat  z;      /* blue, value, saturation, and so on */
        } general;
    } val;
} Pgcolr;
```

Constants defined for colour type are:

0	PINDIRECT	*Indirect*
1	PMODEL_RGB	*Red, Green, Blue*
2	PMODEL_CIELUV	*CIE*
3	PMODEL_HSV	*Hue, Saturation, Value*
4	PMODEL_HLS	*Hue, Lightness, Saturation*

The index member of the val union is used for type PINDIRECT. The general member is used for the other types.

FORTRAN Input Parameters

WKTYPE
The workstation type.

INDEX Index of the predefined entry to be returned.

FORTRAN Output Parameters

ERRIND
The error number of any error that this function detects.

EFLAG The edge flag.

ETYPE The edge type.

EWIDTH
The edge width scale factor.

COLTYPE
The colour type.

IECOLR
An integer containing the index into the workstation colour table for the edge colour.

ECOLR(3)
An array of reals containing the edge colour.

Execution

If the inquired information is available, then the error indicator is returned as zero, and values are returned in the output parameters.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the error numbers below to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)

051 Ignoring function, this information unavailable for this workstation type

052 Ignoring function, workstation type not recognized by the implementation

062 Ignoring function, this information is not available for this MO workstation type

100 Ignoring function, the bundle index value is less than one

102 Ignoring function, the specified representation has not been predefined on this workstation

SEE ALSO

SET EDGE REPRESENTATION PLUS (3PP)
INQUIRE EDGE REPRESENTATION PLUS (3PP)
INQUIRE PREDEFINED EDGE REPRESENTATION (3P)

NAME

INQUIRE PREDEFINED INTERIOR REPRESENTATION PLUS – inquire for values of predefined extended interior representation for specified workstation type

SYNOPSIS

C Syntax

pinq_pred_int_rep_plus (type, index, error_ind, bundle)

Pint	type;	*workstation type*
Pint	index;	*interior index*
Pint	*error_ind;	*OUT error indicator*
Pint_bundle_plus	*bundle;	*OUT predefined interior representation*

FORTRAN Syntax

SUBROUTINE pqpeir (WKTYPE, INDEX, ERRIND, STYLE, ISTYLE, COLTYPE, ICOLR, COLR, REFEQ, SHADEM, AMCOEF, DICOEF, SPCOEF, SCOLM, ISCOLR, SCOLR, SPEXP, TRCOEF, APTYP, APVAL)

INTEGER	WKTYPE	*workstation type*
INTEGER	INDEX	*predefined interior index*
INTEGER	ERRIND	*OUT error indicator*
INTEGER	STYLE(2)	*OUT interior styles*
INTEGER	ISTYLE(2)	*OUT interior style indexes*
INTEGER	COLTYPE(2)	*OUT interior colour types*
INTEGER	ICOLR(2)	*OUT interior colour indexes*
REAL	COLR(3,2)	*OUT interior colour triples*
INTEGER	REFEQ(2)	*OUT reflectance equations*
INTEGER	SHADEM(2)	*OUT shading methods*
REAL	AMCOEF(2)	*OUT ambient reflectance coefficients*
REAL	DICOEF(2)	*OUT diffuse reflectance coefficients*
REAL	SPCOEF(2)	*OUT specular reflectance coefficients*
INTEGER	SCOLM(2)	*OUT specular colour models*
INTEGER	ISCOLR(2)	*OUT specular colour indexes*
REAL	SCOLR(3,2)	*OUT specular colour triples*
REAL	SPEXP(2)	*OUT specular exponents*
REAL	TRCOEF(2)	*OUT transparency coefficients*
INTEGER	APTYP	*OUT surface approximation type*
REAL	APVAL(2)	*OUT surface approximation values, u and v*

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE PREDEFINED INTERIOR REPRESENTATION PLUS returns the values of a specified entry in the workstation type's table of predefined interior representations.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

type The workstation type.

index Index of the predefined entry to be returned.

C Output Parameters

error_ind
A pointer to the location for storing the error number of any error this function detects.

bundle A pointer to a Pint_bundle_plus structure in which the system returns the specified predefined extended interior representation. Pint_bundle_plus is defined as:

```
typedef struct {
    Pint_style      style;          /* interior style */
    Pint            style_ind;      /* interior style index */
    Pgcolr          colr;           /* interior colour */
    Pint            refl_eqn;       /* reflectance equation */
    Pint            shad_meth;      /* shading method */
    Prefl_props     refl_props;     /* area properties */
    Pint_style      back_style;     /* interior style */
    Pint            back_style_ind; /* interior style index */
    Pgcolr          back_colr;      /* interior colour */
    Pint            back_refl_eqn;  /* back reflectance equation */
    Pint            back_shad_meth; /* back shading method */
    Prefl_props     back_refl_props; /* back area properties */
    Pint            approx_type;    /* approximation method */
    Pfloat          approx_val[2];  /* approximation values,
                                       u and v */
} Pint_bundle_plus;
```

Pint_style is defined as:

```
typedef enum {
        PSTYLE_HOLLOW,
        PSTYLE_SOLID,
        PSTYLE_PAT,
        PSTYLE_HATCH,
        PSTYLE_EMPTY
} Pint_style;
```

See SET INTERIOR STYLE for a description of each style.

Pgcolr is defined as:

```
typedef struct {
    Pinttype;    /* indirect, RGB, CIE, HSV, HLS */
    union {
        Pint     ind;      /* index in workstation colour bundle table */
        struct {
```

```
                Pfloat  x;      /* red, hue, and so on */
                Pfloat  y;      /* green, saturation, lightness, and so on */
                Pfloat  z;      /* blue, value, saturation, and so on */
            } general;
        } val;
    } Pgcolr;
```

Constants defined for colour type are:

0	PINDIRECT	*Indirect*
1	PMODEL_RGB	*Red, Green, Blue*
2	PMODEL_CIELUV	*CIE*
3	PMODEL_HSV	*Hue, Saturation, Value*
4	PMODEL_HLS	*Hue, Lightness, Saturation*

The index member of the *val* union is used for type PINDIRECT. The general member is used for the other types.

Prefl_props is defined as:

```
typedef struct {
    Pfloat  ambient_coef;   /* ambient reflectance coefficient */
    Pfloat  diffuse_coef;   /* diffuse reflectance coefficient */
    Pfloat  specular_coef;  /* specular reflectance coefficient */
    Pgcolr  specular_colr;  /* specular colour */
    Pfloat  specular_exp;   /* specular exponent */
    Pfloat  transpar_coef;  /* transparency coefficient */
} Prefl_props;
```

The values for ambient, diffuse, specular, and transparency coefficients must all be in the range [0,1]. The specular exponent must be greater than zero.

Defined reflectance equation values are:

1	PREFL_NONE	*No reflectance calculation performed*
2	PREFL_AMBIENT	*Use ambient term*
3	PREFL_AMB_DIFF	*Use ambient and diffuse terms*
4	PREFL_AMB_DIFF_SPEC	*Use ambient, diffuse, and specular terms*

Defined constants for interior shading method are:

1	PSD_NONE	*No shading*
2	PSD_COLOUR	*Colour interpolation shading*
3	PSD_DOT_PRODUCT	*Dot product interpolation shading*
4	PSD_NORMAL	*Normal interpolation shading*

Constants defined for surface approximation type are:

1	PSURF_WS_DEP
2	PSURF_CONSTANT_PARAMETRIC_BETWEEN_KNOTS
3	PSURF_CHORDAL_SIZE_WC
4	PSURF_CHORDAL_SIZE_NPC
5	PSURF_CHORDAL_SIZE_DC

6 PSURF_PLANAR_DEVIATION_WC
7 PSURF_PLANAR_DEVIATION_NPC
8 PSURF_PLANAR_DEVIATION_DC

See SET SURFACE APPROXIMATION CRITERIA for a description of the meaning of these constants.

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

WKTYPE
The workstation type. See *OPEN WORKSTATION* for a list of the available workstation types.

INDEX Index of the predefined entry to be returned from the extended interior bundle table of the workstation description table of that workstation type.

FORTRAN Output Parameters

ERRIND
The error number of any error that this function detects.

STYLE(2)
The front and back interior styles. Valid values are defined as:

0 PHOLLO *Hollow*
1 PSOLID *Solid*
2 PPATTR *Patterned*
3 PHATCH *Hatched*
4 PISEMP *Empty*

See SET INTERIOR STYLE for a description of each style.

ISTYLE(2)
The front and back interior style indices, which select entries in the workstation interior style tables.

COLTYPE(2)
The front and back colour types. The defined values are:

0 PINDIR *Colour Index Specified*
1 PRGB *Red, Green, and Blue*
2 PCIE *CIE Colour Model*
3 PHSV *Hue, Saturation, and Value*
4 PHLS *Hue, Lightness, and Saturation*

ICOLR(2)
The front and back colour indices. ICOLR is used only if the corresponding value in COLTYPE is PINDIR.

COLR(3,2)
The front and back colour triples. COLR is used only if the corresponding value in COLTYPE is not PINDIR. The triples are stored as:
ICOLR(1,c) = *x Colour*
ICOLR(2,c) = *y Colour*

ICOLR(3,c) = z *Colour*

REFEQ(2)
The front and back reflectance equations. Defined reflectance equation values are:

1	PRNONE	*No Reflectance Calculation Performed*
2	PRAMB	*Use Ambient Term*
3	PRAMBD	*Use Ambient and Diffuse Terms*
4	PRAMDS	*Use Ambient, Diffuse, and Specular Terms*

The front and back interior shading methods. Defined constants are:

1	PSDNON	*No Shading*
2	PSDCOL	*Colour Interpolation Shading*
3	PSDDOT	*Dot Product Interpolation Shading*
4	PSDNOR	*Normal Interpolation Shading*

AMCOEF(2)
The front and back ambient reflectance coefficients, in the range [0,1].

DICOEF(2)
The front and back diffuse reflectance coefficients, in the range [0,1].

SPCOEF(2)
The front and back specular reflectance coefficients, in the range [0,1].

SCOLM(2)
The front and back specular colour models.

ISCOLR(2)
The front and back specular colour indices. *ISCOLR* is used only if the corresponding value in SCOLM is PINDIR.

SCOLR(3,2)
The front and back specular colour triples. *SCOLR* is used only if the corresponding value in SCOLM is not PINDIR.

SPEXP(2)
The front and back specular exponents, greater than zero.

TRCOEF(2)
The front and back transparency coefficients, in the range [0,1].

APTYP The surface approximation type. Defined values are:

1	PSAWSD	*Workstation-dependent*
2	PSAEPK	*Constant Parametric Subdivision Between Knots*
3	PSACSW	*Chordal Size in WC*
4	PSACSN	*Chordal Size in NPC*
5	PSACSD	*Chordal Size in DC*
6	PSAPDW	*Planar Deviation in WC*
7	PSAPDN	*Planar Deviation in NPC*
8	PSAPDD	*Planar Deviation in DC*

For their descriptions, see SET SURFACE APPROXIMATION CRITERIA.

APVAL(2)
The surface approximation values. For their descriptions, see SET SURFACE APPROXIMATION CRITERIA.

Execution

If the inquired information is available, then the error indicator is returned as zero, and information is returned in the output parameters.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the error numbers below to indicate the reason for nonavailability.

The extended interior bundle table contains bundled entries of the PHIGS and SunPHIGS Extension interior attributes. During traversal, the attribute values of these bundles are used when the corresponding aspect source flag in the workstation state list is set to BUNDLED.

The PHIGS interior bundle table is a portion of the extended table. The function INQUIRE PREDEFINED INTERIOR REPRESENTATION returns the following attributes from the extended table: interior style, interior style index, and colour index.

Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)

051 Ignoring function, the information unavailable for this workstation type

052 Ignoring function, workstation type not recognized by the implementation

062 Ignoring function, this information is not available for this MO workstation type

100 Ignoring function, the bundle index value is less than one

102 Ignoring function, the specified representation has not been predefined on this workstation

SEE ALSO

SET INTERIOR REPRESENTATION PLUS (3PP)
INQUIRE INTERIOR REPRESENTATION PLUS (3PP)
INQUIRE INTERIOR FACILITIES PLUS (3PP)
INQUIRE PREDEFINED INTERIOR REPRESENTATION (3P)

NAME

INQUIRE PREDEFINED LIGHT SOURCE REPRESENTATION – inquire for predefined light source representation

SYNOPSIS

C Syntax

```
void
pinq_pred_light_src_rep ( type, index, error_ind, bundle )
Pwstype             type;       workstation type
Pint                index;      predefined index
Pint                *error_ind; OUT error indicator
Plight_src_bundle   *bundle;    OUT predefined light source representation
```

FORTRAN Syntax

```
SUBROUTINE pqplsr ( WKTYPE, ILS, MLDR, ERRIND, LSTYPE, LDR, DATREC )
INTEGER             WKTYPE      workstation type
INTEGER             ILS         index into predefined light source list
INTEGER             MLDR        dimension of data record array
INTEGER             ERRIND      OUT error indicator
INTEGER             LSTYPE      OUT light source type
INTEGER             LDR         OUT actual size of the data record
CHARACTER*80        DATREC(LDR) OUT data record
```

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE PREDEFINED LIGHT SOURCE REPRESENTATION inquires for an entry in the predefined light source table of a specified workstation type's workstation description table.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

type The workstation type.

index Index of the predefined entry to be returned.

C Output Parameters

error_ind

A pointer to the location in which to store the error number of any error that this function detects.

bundle A pointer to the location in which to return the light source representation. The Plight_src_bundle data structure contains values needed to define the specified type of light source representation. Plight_src_bundle is defined as:

```
typedef struct {
        Pint            type;   /* light source type */
        Plight_src_rec  rec;    /* light source data record */
} Plight_src_bundle;
```

Plight_src_rec is defined as:

```
typedef union {
        Pamb_light_src_rec      ambient;
        Pdir_light_src_rec      directional;
        Ppos_light_src_rec      positional;
        Pspot_light_src_rec     spot;
} Plight_src_rec;
```

Pamb_light_src_rec is defined as:

```
typedef struct {
        Pgcolr    colr;    /* light source colour */
} Pamb_light_src_rec;
```

Pdir_light_src_rec is defined as:

```
typedef struct {
        Pgcolr    colr;    /* light source colour */
        Pvec3     dir;     /* light source direction */
} Pdir_light_src_rec;
```

Ppos_light_src_rec is defined as:

```
typedef struct {
        Pgcolr      colr;       /* light source colour */
        Ppoint3     pos;        /* light source position */
        Pfloat      coef[2];    /* attenuation coefficients */
} Ppos_light_src_rec;
```

Pspot_light_src_rec is defined as:

```
typedef struct {
        Pgcolr      colr;       /* light source colour */
        Ppoint3     pos;        /* light source position */
        Pvec3       dir;        /* light source direction */
        Pfloat      exp;        /* concentration exponent */
        Pfloat      coef[2];    /* attenuation coefficients */
        Pfloat      angle;      /* spread angle */
} Pspot_light_src_rec;
```

Pgcolr is defined as:

```
typedef struct {
        Pint          type;         /* indirect, RGB, CIE, HSV, HLS */
        union {
           Pint       ind;          /* index in workstation colour
```

```
                                        bundle table */
                struct {
                    Pfloat x;           /* red, hue, and so on */
                    Pfloat y;           /* green, saturation, lightness, and
                                           so on */
                    Pfloat z;           /* blue, value, saturation, and so on */
                } general;
            } val;
        } Pgcolr;
```

Pvec3 is defined as:

```
typedef struct {
        Pfloat    delta_x;    /* x magnitude */
        Pfloat    delta_y;    /* y magnitude */
        Pfloat    delta_z;    /* z magnitude */
} Pvec3;
```

Ppoint3 is defined as:

```
typedef struct {
        Pfloat    x;    /* x coordinate */
        Pfloat    y;    /* y coordinate */
        Pfloat    z;    /* z coordinate */
} Ppoint3;
```

FORTRAN Input Parameters

WKTYPE
The workstation type for which the predefined light source representation is being inquired.

ILS The index of the predefined entry to be returned.

MLDR The dimension of the data record array, DATREC.

FORTRAN Output Parameters

ERRIND
A pointer to the location to store the error number of any error detected by this function.

LSTYPE
The light source type.

LDR The actual size, in elements, returned in the data record.

DATREC(LDR)
The data record that must be unpacked by UNPACK DATA RECORD. The contents depend on the light source type. See SET LIGHT SOURCE REPRESENTATION for a description of its contents.

Execution

If the inquired information is available, then the error indicator is returned as zero, and values are returned in the output parameters.

See SET LIGHT SOURCE REPRESENTATION for the defined data record formats that can be returned in the data record parameter.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the error numbers below to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)

051 Ignoring function, this information unavailable for this workstation type

052 Ignoring function, workstation type not recognized by the implementation

062 Ignoring function, this information is not available for this MO workstation type

102 Ignoring function, specified representation has not been predefined on this workstation

614 Ignoring function, the general interior index is less than one

SEE ALSO

SET LIGHT SOURCE REPRESENTATION (3PP)
SET LIGHT SOURCE STATE (3PP)
INQUIRE LIST OF LIGHT SOURCE INDICES (3PP)
INQUIRE LIGHT SOURCE REPRESENTATION (3PP)
INQUIRE LIGHT SOURCE FACILITIES (3PP)

NAME

INQUIRE PREDEFINED PATTERN REPRESENTATION PLUS – inquire for predefined extended pattern representation for specified workstation type

SYNOPSIS

C Syntax

```
void
pinq_pred_pat_rep_plus ( type, index, store, error_ind, rep )
Pint          type;       workstation type
Pint          index;      predefined index
Pstore        store;      handle to Store object
Pint          *error_ind; OUT error indicator
Ppat_rep_plus **rep;      OUT predefined pattern representation
```

FORTRAN Syntax

```
SUBROUTINE pqpenr ( WKTYPE, INDEX, PSIZ, ERRIND, PBUF, ACTSIZ,
    PROWS, PCOLS, COLMOD, IPCOLR, PCOLR )
INTEGER          WKTYPE                 workstation type
INTEGER          INDEX                  predefined pattern index
INTEGER          PSIZ                   buffer size dimension
INTEGER          ERRIND                 OUT error indicator
CHARACTER*80     PBUF(PSIZ)             OUT pattern buffer
INTEGER          ACTSIZ                 OUT actual buffer size
INTEGER          PROWS                  OUT pattern dimensions
                                        along y-axis
INTEGER          PCOLS                  OUT pattern dimensions
                                        along x-axis
INTEGER          COLMOD                 OUT colour model
INTEGER          IPCOLR(PROWS*PCOLS)    OUT indirect colour
REAL             PCOLR(3,PROWS*PCOLS)   OUT array of colour
```

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE PREDEFINED PATTERN REPRESENTATION PLUS inquires for a predefined extended pattern representation for a specified workstation type.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

Applications using the C binding must create a buffer to be used by this function as memory space for storing data associated with the device state. This buffer is passed as the *store* argument.

The store buffer is a data area managed by PHIGS. While the application is responsible for creating the initial buffer through a call to CREATE STORE, PHIGS manages this area to provide sufficient memory for the specific inquiry. The data record within the store buffer is accessed by the pointer pointed to by *rep*.

type The workstation type.

index Index of the predefined entry to be returned.

store The memory buffer PHIGS is to use for storing the information returned. This buffer must exist prior to calling this function (see CREATE STORE (3P)).

C Output Parameters

error_ind

A pointer to the location to store the error number of any error detected by this function.

rep A pointer to a pointer to a Ppat_rep_plus structure that returns the requested representation. Ppat_rep_plus is defined in phigs.h as follows:

```
typedef struct {
        Pint_size       dims;           /* pattern's dimensions */
        Pint            type;           /* colour type */
        Pcoval          *colr_array;    /* array of colours */
} Ppat_rep_plus;
```

Pint_size is defined in phigs.h as follows:

```
typedef struct {
        Pint            size_x;
        Pint            size_y;
} Pint_size;
```

Pcoval is defined in phigs.h as follows:

```
typedef union {
        Pint            ind;            /* index in workstation colour
                                           bundle table */
        Pcolr_rep       direct;         /* direct colour components */
} Pcoval;
```

Pcolr_rep is defined in phigs.h as follows:

```
typedef union {
        Prgb            rgb;            /* Red Green Blue colour
                                           specification */
        Pcieluv         cieluv;         /* CIE L*U*V* colour specification */
        Phls            hls;            /* Hue Lightness Saturation colour
                                           specification */
        Phsv            hsv;            /* Hue Saturation Value colour
                                           specification */
        Pdata           unsupp;         /* Colour in unsupported colour
                                           model */
} Pcolr_rep;
```

Prgb is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          red;            /* red, hue, and so on */
        Pfloat          green;          /* green, saturation, lightness,
                                           and so on */
        Pfloat          blue;           /* blue, value, saturation, and
                                           so on */
} Prgb;
```

Pcieluv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          cieluv_x;       /* x coefficient */
        Pfloat          cieluv_y;       /* y coefficient */
        Pfloat          cieluv_y_lum;   /* y luminance */
} Pcieluv;
```

Phsv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          satur;          /* saturation */
        Pfloat          value;          /* value */
} Phsv;
```

Phls is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          lightness;      /* lightness */
        Pfloat          satur;          /* saturation */
} Phls;
```

Pdata is defined in phigs.h as follows:

```
typedef struct {
        size_t          size;           /* size of data */
        char            *data           /*pointer to data */
} Pdata;
```

FORTRAN Input Parameters

WKTYPE
The workstation type.

INDEX Index of the predefined entry to be returned.

PSIZ The size, in number of bytes, of the application-supplied buffer in which the requested pattern representation is stored.

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

PBUF A character array in which to return the requested pattern representation.

ACTSIZ The actual size, in bytes, filled in the pattern buffer PBUF.

PROWS The pattern dimensions along the *y* axis.

PCOLS The pattern dimensions along the *x* axis.

COLMOD
The colour model.

IPCOLR(PROWS∗PCOLS)
An integer index into the workstation colour table for the pattern colour.

PCOLR(3,PROWS∗PCOLS)
An array of reals containing the pattern colour.

Execution

If the inquired information is available, then the error indicator is returned as zero, and values are returned in the output parameters.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the following error numbers to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)

051 Ignoring function, this information unavailable for this workstation type

052 Ignoring function, workstation type not recognized by the implementation

062 Ignoring function, this information is not available for this MO workstation type

100 Ignoring function, the bundle index value is less than one

102 Ignoring function, the specified representation has not been predefined on this workstation

SEE ALSO

SET PATTERN REPRESENTATION PLUS (3PP)
INQUIRE PATTERN REPRESENTATION PLUS (3PP)
INQUIRE PREDEFINED PATTERN REPRESENTATION (3P)

NAME

INQUIRE PREDEFINED POLYLINE REPRESENTATION PLUS – inquire for predefined extended polyline representation from specified workstation type

SYNOPSIS

C Syntax

void
pinq_pred_line_rep_plus (type, index, error_ind, bundle)

Pint	type;	*workstation type*
Pint	index;	*predefined index*
Pint	*error_ind;	*OUT error indicator*
Pline_bundle_plus	*bundle;	*OUT predefined polyline representation*

FORTRAN Syntax

SUBROUTINE pqpepr (WKTYPE, INDEX, ERRIND, LTYPE, LWIDTH, COLMOD, ICOLR, COLR, METHOD, ATYPE, AVALUE)

INTEGER	WKTYPE	*workstation type*
INTEGER	INDEX	*predefined polyline index*
INTEGER	ERRIND	*OUT error indicator*
INTEGER	LTYPE	*OUT linetype*
REAL	LWIDTH	*OUT linewidth scale factor*
INTEGER	COLMOD	*OUT colour model*
INTEGER	ICOLR	*OUT polyline colour index*
REAL	COLR(3)	*OUT colour values*
INTEGER	METHOD	*OUT polyline shading method*
INTEGER	ATYPE	*OUT curve approximation type*
REAL	AVALUE	*OUT curve approximation value*

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE PREDEFINED POLYLINE REPRESENTATION PLUS returns the attribute values of a specified predefined entry in the extended polyline bundle table of a workstation type's workstation description table. See SET POLYLINE REPRESENTATION PLUS for a full description of the contents of the bundle table entry and a description of the extended polyline bundle table.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

type The workstation type.

index Index of the predefined entry to be returned.

C Output Parameters

error_ind
A pointer to the location to store the error number of any error detected by this function.

bundle A pointer to a Pline_bundle_plus structure that returns the requested representation. Pline_bundle_plus is defined as:

```
typedef struct {
        Pint    type;           /* line type */
        Pfloat  width;          /* linewidth scale factor */
        Pgcolr  colr;           /* polyline colour */
        Pint    shad_meth;      /* polyline shading method */
        Pint    approx_type;    /* curve approximation criteria */
        Pfloat  approx_val;     /* curve approximation criteria */
} Pline_bundle_plus;
```

See SET POLYLINE REPRESENTATION PLUS for a full description of the contents of this structure.

FORTRAN Input Parameters

WKTYPE
The workstation type.

INDEX Index of the predefined entry to be returned.

FORTRAN Output Parameters

See SET POLYLINE REPRESENTATION PLUS for a full description of the below parameters.

ERRIND
The error number of any error detected by this function.

LTYPE The line type.

LWIDTH
The linewidth scale factor.

COLMOD
The colour model. The predefined values are:

0	PINDIR	*Indirect*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

ICOLR An integer containing the index into the workstation colour table for the polyline colour. This value is not used if COLMOD is greater than zero.

COLR(3)
An array of reals containing the polyline colour. This array is unused if COLMOD is equal to zero (PINDIR).

COLR(1) = *Red, Hue, and so on*
COLR(2) = *Green, Saturation, Lightness, and so on*
COLR(3) = *Blue, Value, Saturation, and so on*

METHOD
The polyline shading method.

ATYPE The curve approximation type.

AVALUE
The curve approximation value.

Execution

If the inquired information is available, then the error indicator is returned as zero, and the requested information is returned in the output parameters.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the error numbers below to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)

051 Ignoring function, this information unavailable for this workstation type

052 Ignoring function, workstation type not recognized by the implementation

062 Ignoring function, this information is not available for this MO workstation type

100 Ignoring function, the bundle index value is less than one

102 Ignoring function, the specified representation has not been predefined on this workstation

SEE ALSO

SET POLYLINE REPRESENTATION PLUS (3PP)
INQUIRE POLYLINE REPRESENTATION PLUS (3PP)
INQUIRE PREDEFINED POLYLINE REPRESENTATION (3P)

NAME

INQUIRE PREDEFINED POLYMARKER REPRESENTATION PLUS – inquire for predefined extended polymarker representation for specified workstation type

SYNOPSIS

C Syntax

```
void
pinq_pred_marker_rep_plus ( type, index, error_ind, bundle )
Pint                  type;       workstation type
Pint                  index;      predefined index
Pint                  *error_ind; OUT error indicator
Pmarker_bundle_plus   *bundle;    OUT predefined polymarker rep
```

FORTRAN Syntax

```
SUBROUTINE pqpemr ( WKTYPE, INDEX, ERRIND, MTYPE, MSIZE, COLMOD,
   IMCOLR, MCOLR )
INTEGER   WKTYPE     workstation type
INTEGER   INDEX      predefined polymarker index
INTEGER   ERRIND     OUT error indicator
INTEGER   MTYPE      OUT polymarker type
REAL      MSIZE      OUT polymarker size scale factor
INTEGER   COLMOD     OUT colour model
INTEGER   IMCOLR     OUT indirect colour index
REAL      MCOLR(3)   OUT direct colour value
```

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE PREDEFINED POLYMARKER REPRESENTATION PLUS returns the attribute values of a specified predefined entry in the extended polymarker bundle table of a workstation type's workstation description table. For a description of the contents of the bundle table entry and a description of the extended polymarker bundle table, see SET POLYMARKER REPRESENTATION PLUS.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

type The workstation type.

index The index of the predefined entry to be returned.

C Output Parameters

error_ind
A pointer to the location to store the error number of any error that this function detects.

bundle A pointer to a Pmarker_bundle_plus structure that returns the requested representation. Pmarker_bundle_plus is defined in phigs.h as:

```
typedef struct {
```

```
        Pint    type;   /* marker type */
        Pfloat  size;   /* marker size scale factor */
        Pgcolr  colr;   /* marker colour */
} Pmarker_bundle_plus;
```

For a description of the contents of this structure, see SET POLYMARKER REPRESENTATION PLUS.

FORTRAN Input Parameters

WKTYPE
The workstation type.

INDEX
The index of the predefined entry to be returned.

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

MTYPE The polymarker type.

MSIZE The polymarker size scale factor.

COLMOD
The colour model. The predefined values are:

0	PINDIR	*Indirect*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

IMCOLR
An integer containing the index into the workstation colour table for the polymarker colour. This value is not used if COLMOD is greater than zero.

MCOLR(3)
An array of reals containing the polymarker colour. This array is unused if COLMOD is equal to zero (PINDIR).

MCOLR(1) = *Red, Hue, and so on*
MCOLR(2) = *Green, Saturation, Lightness, and so on*
MCOLR(3) = *Blue, Value, Saturation, and so on*

Execution

If the inquired information is available, then the error indicator is returned as zero, and values are returned in the output parameters.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the error numbers below to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)

051 Ignoring function, this information unavailable for this workstation type

052 Ignoring function, workstation type not recognized by the implementation

062 Ignoring function, this information is not available for this MO workstation type

100 Ignoring function, the bundle index value is less than one

102 Ignoring function, the specified representation has not been predefined on this workstation

SEE ALSO

SET POLYMARKER REPRESENTATION PLUS (3PP)
INQUIRE POLYMARKER REPRESENTATION PLUS (3PP)
INQUIRE PREDEFINED POLYMARKER REPRESENTATION (3P)

NAME

INQUIRE PREDEFINED TEXT REPRESENTATION PLUS – inquire for predefined extended text representation for specified workstation type

SYNOPSIS

C Syntax

```
void
pinq_pred_text_rep_plus ( type, index, error_ind, bundle )
Pint                type;       workstation type
Pint                index;      predefined index
Pint                *error_ind; OUT error indicator
Ptext_bundle_plus   *bundle;    OUT predefined text rep
```

FORTRAN Syntax

```
SUBROUTINE pqpetr ( WKTYPE, INDEX, ERRIND, TFONT, TPREC, TCEXPF,
    TSPACE, COLMOD, ITCOLR, TCOLR )
INTEGER   WKTYPE    workstation type
INTEGER   INDEX     predefined text index
INTEGER   ERRIND    OUT error indicator
INTEGER   TFONT     OUT text font
INTEGER   TPREC     OUT text precision
REAL      TCEXPF    OUT text character expansion factor
REAL      TSPACE    OUT text character spacing
INTEGER   COLMOD    OUT colour model
INTEGER   ITCOLR    OUT indirect colour index
REAL      TCOLR(3)  OUT direct colour value
```

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE PREDEFINED TEXT REPRESENTATION PLUS inquires for a predefined extended text representation for a specified workstation type.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

type The workstation type.

index Index of the predefined entry to be returned.

C Output Parameters

error_ind
A pointer to the location to store the error number of any error detected by this function.

bundle A pointer to a Ptext_bundle_plus structure that returns the specified extended text representation. Ptext_bundle_plus is defined as:

```
typedef struct {
        Pint        font;       /* text font */
```

```
        Ptext_prec    prec;          /* text precision */
        Pfloat        char_expan;    /* character expansion factor */
        Pfloat        char_space;    /* character spacing */
        Pgcolr        colr;          /* text colour */
} Ptext_bundle_plus;
```

See SET TEXT REPRESENTATION PLUS for a description of the contents of this structure.

FORTRAN Input Parameters

WKTYPE
The workstation type.

INDEX
Index of the predefined entry to be returned.

FORTRAN Output Parameters

ERRIND
The error number of any error detected by this function.

TFONT The text font.

TPREC The text precision.

TCEXPF
The text character expansion factor.

TSPACE
The text character spacing.

COLMOD
The colour model. The predefined values are:

0	PINDIR	*Colour Index Specified*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

ITCOLR
An integer containing the index into the workstation colour table for the text colour. This value is not used if COLMOD is greater than zero.

TCOLR(3)
An array of reals containing the text colour. This array is unused if COLMOD is equal to zero (PINDIR).

TCOLR(1) = *Red, Hue, and so on*
TCOLR(2) = *Green, Saturation, Lightness, and so on*
TCOLR(3) = *Blue, Value, Saturation, and so on*

Execution

If the inquired information is available, then the error indicator is returned as zero, and values are returned in the output parameters.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the error numbers below to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when an error is detected by this function.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)

051 Ignoring function, this information unavailable for this workstation type

052 Ignoring function, workstation type not recognized by the implementation

062 Ignoring function, this information is not available for this MO workstation type

100 Ignoring function, the bundle index value is less than one

102 Ignoring function, the specified representation has not been predefined on this workstation

SEE ALSO

SET TEXT REPRESENTATION PLUS (3PP)
INQUIRE TEXT REPRESENTATION PLUS (3PP)
INQUIRE PREDEFINED TEXT REPRESENTATION (3P)

NAME

INQUIRE RENDERING COLOUR MODEL FACILITIES – inquire for list of available direct colour models

SYNOPSIS

C Syntax

void
pinq_rendering_colr_model_facs (type, length, start, error_ind, models, total_length)

Pint	type;	*workstation type*
Pint	length;	*length of application's list*
Pint	start;	*starting position*
Pint	*error_ind;	*OUT error indicator*
Pint_list	*models;	*OUT rendering colour models*
Pint	*total_length;	*OUT total number of rendering*

FORTRAN Syntax

SUBROUTINE pqrcmf (WTYPE, N, ERRIND, NRCM, RCM)

INTEGER	WTYPE	*workstation type*
INTEGER	N	*list element requested*
INTEGER	ERRIND	*OUT error indicator*
INTEGER	NRCM	*OUT number of rendering colour models*
INTEGER	RCM	*OUT Nth element of list of rendering colour models*

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE RENDERING COLOUR MODEL FACILITIES inquires for a list of colour model facilities available for rendering for the specified workstation type

C Input Parameters

type The workstation type.

length The number of *ints* in the *models* output parameter for which the application has allocated memory. *length* is the number of list elements that the system can return in *models→ints*. If a value of 0 is used here, no data is returned in the *models→ints* list, but the total number of elements is returned in *total_length*.

start Starting position of inquiry. The elements in the list, beginning with the item number specified by *start*, are copied sequentially into *models→ints* until *models→ints* is full or all the elements have been copied.

C Output Parameters

error_ind
A pointer to the location to store the error number of any error that this function detects.

models A pointer to a Pint_list that returns the portion of the list of available rendering colour models from the workstation description table, starting with *start*. Pint_list is defined in phigs.h as follows:

typedef struct {

```
        Pint        num_ints;    /* number of Pints in list */
        Pint        *ints;       /* list of integers */
    } Pint_list;
```

The *num_ints* component specifies the number of elements in the list. The *ints* component is a pointer to a list *num_ints* long.

The application must allocate memory for *length* elements in the list of *ints*.

total_length
A pointer to an integer in which to return the total length of the list. This is the value required for *length* if all the items in the list are to be returned.

FORTRAN Input Parameters

WTYPE The workstation type.

N Element of list of rendering colour models to return in output parameter RCM.

FORTRAN Output Parameters

ERRIND
The number of rendering colour models available for the specified workstation type.

RCM The Nth element of the list of rendering colour models available for the specified workstation type.

Execution

If the inquired information is available, then the error indicator is returned as zero, and values are returned in the output parameters.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the error numbers below to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

ERRORS

–153	List length is less than zero — zero will be used
002	Ignoring function, function requires state (PHOP, *, *, *)
052	Ignoring function, workstation type not recognized by the implementation
059	Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
062	Ignoring function, specified workstation category is not OUTPUT or OUTIN
2201	Start index out of range

SEE ALSO

SET RENDERING COLOUR MODEL (3P)

NAME

INQUIRE TEXT REPRESENTATION PLUS – inquire for extended text representation on workstation

SYNOPSIS

C Syntax

void
pinq_text_rep_plus (ws, index, type, error_ind, rep)

Pint	ws;	*workstation identifier*
Pint	index;	*text index*
Pinq_type	type;	*type of returned value*
Pint	*error_ind;	*OUT error indicator*
Ptext_bundle_plus	*rep;	*OUT extended text representation*

FORTRAN Syntax

SUBROUTINE pqetr (WKID, INDEX, ITYPE, ERRIND, TFONT, TPREC, TCEXPF, TSPACE, COLTYPE, ITCOLR, TCOLR)

INTEGER	WKID	*workstation identifier*
INTEGER	INDEX	*text bundle index*
INTEGER	ITYPE	*type of returned values (PSET, PREALI)*
INTEGER	ERRIND	*OUT error indicator*
INTEGER	TFONT	*OUT text font*
INTEGER	TPREC	*OUT text precision*
REAL	TCEXPF	*OUT text character expansion factor*
REAL	TSPACE	*OUT text character spacing*
INTEGER	COLTYPE	*OUT colour type*
INTEGER	ITCOLR	*OUT indirect colour index*
REAL	TCOLR(3)	*OUT direct colour value*

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

INQUIRE TEXT REPRESENTATION PLUS inquires for an extended text representation on a workstation.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

ws The workstation identifier.

index The text index.

type An enumerated value specifying whether the values to be returned are those originally specified by the application (PINQ_SET), or those resulting after PHIGS mapped them to ones available on the workstation (PINQ_REALIZED). A Pinq_type structure is defined in phigs.h as:

```
typedef enum {
        PINQ_SET,
```

```
        PINQ_REALIZED
    } Pinq_type;
```

C Output Parameters

error_ind

A pointer to the location to store the error number of any error that this function detects.

rep A pointer to a Ptext_bundle_plus structure, which returns the specified extended text representation. Ptext_bundle_plus is defined as:

```
typedef struct {
    Pint        font;       /* text font */
    Ptext_prec  prec;       /* text precision */
    Pfloat      char_expan; /* character expansion
                               factor */
    Pfloat      char_space; /* character spacing */
    Pgcolr      colr;       /* text colour */
} Ptext_bundle_plus;
```

Ptext_prec is defined as:

```
typedef enum {
    PPREC_STRING,
    PPREC_CHAR,
    PPREC_STROKE
} Ptext_prec;
```

Pgcolr is defined as:

```
typedef struct {
    Pint        type;   /* indirect, RGB, CIE, HSV, HLS */
    union {
        Pint    ind;    /* index in workstation colour bundle
                           table */
        struct {
            Pfloat  x;  /* red, hue, and so on */
            Pfloat  y;  /* green, saturation, lightness, and
                           so on */
            Pfloat  z;  /* blue, value, saturation, and so on */
        } general;
    } val;
} Pgcolr;
```

Constants defined for colour type are:

0	PINDIRECT	*Indirect*
1	PMODEL_RGB	*Red, Green, Blue*
2	PMODEL_CIELUV	*CIE*
3	PMODEL_HSV	*Hue, Saturation, Value*

4 PMODEL_HLS *Hue, Lightness, Saturation*

The index member of the *val* union is used for type PINDIRECT. The general member is used for the other types.

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

WKID The identifier of the workstation for which the text representation is being defined.

INDEX The index into the specified workstation text bundle table.

ITYPE An enumerated value specifying whether the values to be returned are those originally specified by the application (*Set*), or those resulting after PHIGS mapped them to ones available on the workstation (*Realized*). Valid values are:

0 PSET *Set*
1 PREALI *Realized*

FORTRAN Output Parameters

ERRIND
The error number of any error that this function detects.

TFONT The text font.

TPREC The text precision.

TCEXPF
The text character expansion factor.

TSPACE
The text character spacing.

COLTYPE
The colour type. The predefined values are:

0 PINDIR *Colour Index Specified*
1 PRGB *Red, Green, and Blue*
2 PCIE *CIE Colour Model*
3 PHSV *Hue, Saturation, and Value*
4 PHLS *Hue, Lightness, and Saturation*

ITCOLR
An integer value containing the index into the workstation colour table for determining the text colour. This value is not used when COLTYPE is greater than zero.

TCOLR(3)
An array of reals containing the text colour. This array is not used when COLTYPE is equal to zero (PINDIR).

TCOLR(1) = *Red, Hue, and so on*
TCOLR(2) = *Green, Saturation, Lightness, and so on*
TCOLR(3) = *Blue, Value, Saturation, and so on*

Execution

If the inquired information is available, then the error indicator is returned as zero, and values are returned in the output parameters.

If the specified text index is not present in the text bundle table on the workstation, and the specified type of returned value is *Realized*, then the representation for text index one is returned.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the error numbers below to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when this function detects an error.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, the specified workstation is not open

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

100 Ignoring function, the bundle index value is less than one

101 Ignoring function, the specified representation has not been defined

SEE ALSO

SET TEXT REPRESENTATION PLUS (3PP)
INQUIRE TEXT REPRESENTATION (3P)
INQUIRE PREDEFINED TEXT REPRESENTATION PLUS (3PP)

NAME

INQUIRE WORKSTATION STATE TABLE LENGTHS PLUS – inquire for the lengths of the extended workstation state tables for a specified workstation type

SYNOPSIS

C Syntax

```
void
pinq_ws_st_table_plus ( type, error_ind, lengths )
Pint              type;        workstation type
Pint              *error_ind;  OUT error indicator
Pws_tables_plus   *lengths;    OUT lengths of the workstation tables
```

FORTRAN Syntax

```
SUBROUTINE pqewtl ( WKTYPE, ERRIND, PLINE, PMARK, TEXT, INTER, EDGE,
   PAT, COLR, VIEW, GENINT, DCUE, LTSRC, CLRMAP, CULL )
INTEGER  WKTYPE   workstation type
INTEGER  ERRIND   OUT error indicator
INTEGER  PLINE    OUT polyline tables
INTEGER  PMARK    OUT polymarker tables
INTEGER  TEXT     OUT text tables
INTEGER  INTER    OUT interior tables
INTEGER  EDGE     OUT edge tables
INTEGER  PAT      OUT pattern tables
INTEGER  COLR     OUT colour tables
INTEGER  VIEW     OUT view tables
INTEGER  GENINT   OUT general interior tables
INTEGER  DCUE     OUT depth cue tables
INTEGER  LTSRC    OUT light source tables
INTEGER  CLRMAP   OUT colour mapping tables
INTEGER  CULL     OUT cull size tables
```

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

INQUIRE WORKSTATION STATE TABLE LENGTHS PLUS inquires for the lengths of the extended workstation state tables for a specified workstation type.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameter

type The workstation type in question.

C Output Parameters

error_ind
A pointer to the location to store the error number of any error that this function detects.

lengths A pointer to a Pws_tables_plus structure, which returns the lengths of the workstation's tables. Pws_tables_plus is defined in phigs.h as follows:

```
typedef struct {
        Pint    line_bundles;   /* polyline tables */
        Pint    mark_bundles;   /* polymarker tables */
        Pint    text_bundles;   /* text tables */
        Pint    int_bundles;    /* interior tables */
        Pint    edge_bundles;   /* edge tables */
        Pint    pat_reps;       /* pattern tables */
        Pint    colr_reps;      /* colour tables */
        Pint    view_reps;      /* view tables */
        Pint    dcue_rep;       /* depth cue tables */
        Pint    light_src_rep;  /* light source tables */
        Pint    colr_map_rep;   /* colour mapping tables */
} Pws_tables_plus;
```

FORTRAN Input Parameter

WKTYPE
The workstation type.

FORTRAN Output Parameters

ERRIND
The error number of any error that this function detects.

PLINE The number of extended polyline bundle table entries.

PMARK
The number of extended polymarker bundle table entries.

TEXT The number of extended text bundle table entries.

INTER The number of extended interior bundle table entries.

EDGE The number of extended edge bundle table entries.

PAT The number of extended pattern bundle table entries.

COLR The number of colour table entries.

VIEW The number of view table entries.

GENINT
The number of general interior table entries.

DCUE The number of depth cue table entries.

LTSRC The number of light source table entries.

CLRMAP
The number of colour mapping table entries.

CULL The number of cull size table entries.

Execution

If the inquired information is available, then the error indicator is returned as zero, and values are returned in the output parameters.

If the inquired information is not available, then the values returned in the output parameters are undefined, and the error indicator is set to one of the error numbers below to indicate the reason for nonavailability.

Since this is an inquiry function, ERROR HANDLING is not invoked when this function dectects an error.

ERRORS

002 Ignoring function, function requires state (PHOP, *, *, *)

051 Ignoring function, this information unavailable for this workstation type

052 Ignoring function, workstation type not recognized by the implementation

062 Ignoring function, this information is not available for this MO workstation type

SEE ALSO

INQUIRE WORKSTATION STATE TABLE LENGTHS (3P)
PHIGS WORKSTATION DESCRIPTION TABLE (7P)

NAME

NON-UNIFORM B-SPLINE CURVE – create a non-uniform B-spline curve structure element

SYNOPSIS

C Syntax

```
void
pnuni_bsp_curv ( order, knots, rationality, cpoints, min, max )
Pint            order;      spline order
Pfloat_list     *knots;     list of knots
Prational       rationality; rationality specifier
Ppoint_list34   *cpoints;   list of 3D or 4D control points
Pfloat          min, max;   parameter range
```

FORTRAN Syntax

```
SUBROUTINE pnubsc (ORDER, NKNOTS, KNOTS, RATSEL, NCPTS, CPTS,
    MIN, MAX )
INTEGER   ORDER          spline order
INTEGER   NKNOTS         number of knots
REAL      KNOTS(NKNOTS)  list of knots
INTEGER   RATSEL         rationality specifier (PRAT, PNRAT)
INTEGER   NCPTS          number of control points
REAL      CPTS(4,NCPTS)  control points (MC)
REAL      MIN, MAX       parameter range
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

NON-UNIFORM B-SPLINE CURVE creates a structure element containing the definition of a non-uniform B-spline curve. The curve can be rational or nonrational.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

order The order of the curve. The degree of the curve's basic functions is one less than *order*.

knots A pointer to a Pfloat_list structure containing a nondecreasing sequence of real numbers specifying the knot vector of the curve. Pfloat_list is defined in phigs.h as follows:

```
typedef struct {
        Pint        num_floats;     /* number of Pfloats in list */
        Pfloat      *floats;        /* list of floats */
} Pfloat_list;
```

rationality

The rationality of the curve. Prational is defined in phigs.h as follows:

```
typedef enum {
        PNON_RATIONAL = 0,
        PRATIONAL = 1
} Prational;
```

cpoints A pointer to a Ppoint_list34 structure containing the list of the curve's control points. The points are specified in Modelling Coordinates. Ppoint_list34 is defined in phigs.h as follows:

```
typedef struct {                                /* list of 3D or 4D points */
        Pint            num_points;             /* number of control points */
        union {
                Ppoint3 *point3d;               /* array of 3D points */
                Ppoint4 *point4d;               /* array of 4D points */
        } points;
} Ppoint_list34;
```

The point4d member of the points union is used if rationality is PRATIONAL; otherwise, the point3d member is used.

Ppoint3 is defined in phigs.h as follows:

```
typedef struct {
        Pfloat  x;      /* x coordinate */
        Pfloat  y;      /* y coordinate */
        Pfloat  z;      /* z coordinate */
} Ppoint3;
```

Ppoint4 is defined in phigs.h as follows:

```
typedef struct {
        Pfloat  x;      /* x coordinate */
        Pfloat  y;      /* y coordinate */
        Pfloat  z;      /* z coordinate */
        Pfloat  w;      /* w coordinate */
} Ppoint4;
```

min, max
The parameter limits specifying the range over which the curve is to be evaluated.

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

ORDER The order of the curve. The degree of the curve's basis functions is one less than ORDER.

NKNOTS
The number of knots in the curve's knot vector.

KNOTS(NKNOTS)
An array containing a nondecreasing sequence of real numbers specifying the curve's knot vector.

RATSEL
An integer specifying the rationality and hence the dimensionality of the control points. Valid values are defined as:

PRAT	*Rational*
PNRAT	*Nonrational*

NCPTS The number of control points.

CPTS(4,NCPTS)
The *x*, *y*, *z*, and *w* coordinates of the curve's control points. The points are specified in Modelling Coordinates (MC) and correspond to the array positions as follows:

CPTS(1,) = x Coordinate*
CPTS(2,) = y Coordinate*
CPTS(3,) = z Coordinate*
CPTS(4,) = w Coordinate*

The *w* coordinates are not used if the rationality is PNRAT.

MIN, MAX
The parameter limits specifying the range over which the curve is to be evaluated.

Execution

Depending on the edit mode, a NON-UNIFORM B-SPLINE CURVE element is either inserted into the open structure after the element pointer, or replaces the element to which the element pointer points. The element pointer is then updated to point to the NON-UNIFORM B-SPLINE CURVE structure element.

The spline order must be greater than zero. Curves of unsupported order will be displayed by drawing a polyline connecting the control points. Curves of order one are drawn as a polymarker using the current polymarker attributes.

The knots must form a nondecreasing sequence of numbers.

The rationality selector parameter may have the enumerated value *Rational* or *Nonrational*. When Rational is specified, the control points are specified as 4D homogeneous modelling coordinates. When Nonrational is specified, the control points are 3DPHIGS modelling coordinates.

The number of control points must be at least as large as the order. The number of control points plus the spline order must equal the number of knots.

The parameter range values, *min* and *max*, specify over what range the curve is evaluated. The value min must be less than max and greater than or equal to the *order*-th knot value. The value max must be less than or equal to the (*k*+1–*order*)-th knot value, where *k* is the

number of knots.

At structure traversal time, a non-uniform B-spline curve primitive is drawn using the extended polyline attributes. The curve approximation criteria is applied between MIN and MAX. See SET POLYLINE REPRESENTATION PLUS and SET CURVE APPROXIMATION CRITERIA for more information on display attributes.

Attributes Applied

The attributes listed below are used to display the NON-UNIFORM B-SPLINE CURVE primitive when the structure is traversed. The Aspect Source Flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

polyline colour	polyline colour index ASF
linewidth scale factor	linewidth scale factor ASF
linetype	linetype ASF
polyline shading method	polyline shading method ASF
curve approximation criteria	curve approximation criteria ASF
polyline index	
depth cue index	
name set	

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

600 Ignoring function, not enough control points for specified order

601 Ignoring function, knot sequence is not nondecreasing

602 Ignoring function, order is inconsistent with number of knots and control points specified

615 Ignoring function, parameter range is inconsistent with knots

SEE ALSO

ADD TRIMMING CURVE (3PP)
SET CURVE APPROXIMATION CRITERIA (3PP)
INQUIRE CURVE AND SURFACE FACILITIES (3PP)
SET POLYLINE REPRESENTATION PLUS (3PP)
INTRO (3PP)

NAME

NON-UNIFORM B-SPLINE SURFACE – create a non-uniform B-spline surface structure element

SYNOPSIS

C Syntax

pnuni_bsp_surf (uorder, vorder, uknots, vknots, rationality, grid, nloops, tloops)

Pint	uorder;	*U spline order*
Pint	vorder;	*V spline order*
Pfloat_list	*uknots;	*U knots*
Pfloat_list	*vknots;	*V knots*
Prational	rationality;	*rationality selector*
Ppoint_grid34	*grid;	*grid of 3D or 4D control points*
Pint	nloops;	*number of trim curve loops*
Ptrimcurve_list	*tloops;	*trim curve loops*

FORTRAN Syntax

SUBROUTINE pnubss (UORDER, VORDER, NUKNTS, NVKNTS, UKNOTS, VKNOTS, RATSEL, NUCPTS, NVCPTS, DIMU, USTART, VSTART, CPTS, NLOOPS, LOOPS, VIS, ORDER, TATYPE, TAVAL, TRATS, NTKNTS, TKSTRT, TKNOTS, NTCPTS, TCSTRT, TCPTS, TMIN, TMAX)

INTEGER	UORDER	*order in u dimension*
INTEGER	VORDER	*order in v dimension*
INTEGER	NUKNTS	*number of knots in u dimension*
INTEGER	NVKNTS	*number of knots in v dimension*
REAL	UKNOTS(NUKNTS)	*knots in u dimension*
REAL	VKNOTS(NVKNTS)	*knots in v dimension*
INTEGER	RATSEL	*surface rationality*
INTEGER	NUCPTS	*number of ctrl points in u dimension*
INTEGER	NVCPTS	*number of ctrl points in v dimension*
INTEGER	DIMU	*u dimension of control points array*
INTEGER	USTART, VSTART	*indices of control point's start location*
REAL	CPTS(4, DIMU, *)	*control points*
INTEGER	NLOOPS	*number of trim loops*
INTEGER	LOOPS(NLOOPS)	*number of trim curves in each loop*
INTEGER	VIS(nt)	*visibility flags for all trim curves*
INTEGER	ORDER(nt)	*curve orders for all trim curves*
INTEGER	TATYPE(nt)	*array of trim curve approximation types*
REAL	TAVAL(nt)	*array of trim curve approximation values*
INTEGER	TRATS(nt)	*curve types for all trim curves*
INTEGER	NTKNTS(nt)	*number of knots in each curve*
INTEGER	TKSTRT(nt)	*start indices of knot lists in TKNOTS*
REAL	TKNOTS(*)	*knots, for all trim curves*
INTEGER	NTCPTS(nt)	*number of ctrl points in each curve*
INTEGER	TCSTRT(nt)	*start indices of control point lists in TCPTS*
REAL	TCPTS(3, *)	*control points, for all trim curves*
REAL	TMIN(nt), TMAX(nt)	*parameter limits for all trim curves*

Note: In the array dimensions above, nt refers to the total number of trim curves.

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

NON-UNIFORM B-SPLINE SURFACE creates a structure element containing the definition of a non-uniform B-spline surface.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

uorder/vorder

Order of the surface in the *u* and *v* dimensions, respectively.

uknots/vknots

Pointers to Pfloat_list structures listing the knots for the *u* and *v* dimensions of the surface. Pfloat_list is defined in phigs.h as follows:

```
typedef struct {
        Pint            num_floats;     /* number of Pfloats in list */
        Pfloat          *floats;        /* list of floats */
} Pfloat_list;
```

rationality

The rationality selector. Prational is defined in phigs.h as follows:

```
typedef enum {
        PNON_RATIONAL = 0,
        PRATIONAL = 1
} Prational;
```

grid Pointer to a Ppoint_grid34 structure containing the surface control points. Ppoint_grid34 is defined in phigs.h as follows:

```
typedef struct {                        /* grid of 3D or 4D points,
                                           [u_dim] [v_dim] */
   Ppcs_dims    num_points;             /* number of points in each
                                           dimension */
   union {
        Ppoint3 *point3d;               /* array of 3D points */
        Ppoint4 *point4d;               /* array of 4D points */
   } points;
} Ppoint_grid34;
```

Ppcs_dims is defined in phigs.h as:

```
typedef struct {
```

```
        Pint    u_dim;    /* dimension (number of divisions) along u */
        Pint    v_dim;    /* dimension (number of divisions) along v */
} Ppcs_dims;
```

Ppoint3 is defined in phigs.h as:

```
typedef struct {
        Pfloat    x;    /* x coordinate */
        Pfloat    y;    /* y coordinate */
        Pfloat    z;    /* z coordinate */
} Ppoint3;
```

Ppoint4 is defined in phigs.h as:

```
typedef struct {
        Pfloat    x;    /* x coordinate */
        Pfloat    y;    /* y coordinate */
        Pfloat    z;    /* z coordinate */
        Pfloat    w;    /* w coordinate */
} Ppoint4;
```

nloops Number of trimming curve loops.

tloops A pointer to an array of Ptrimcurve_list structures that specifies the trimming curve loops. Each of these structures contains the specification of an individual trimming curve loop (set of trimming curves). Ptrimcurve_list is defined in phigs.h as:

```
typedef struct {
        Pint            num_curves;    /* number of trimming curves
                                          in list */
        Ptrimcurve      *curves;       /* list of curves */
} Ptrimcurve_list;
```

Ptrimcurve is defined in phigs.h as:

```
typedef struct {
        Pedge_flag      visible;       /* curve visibility flag */
        Prational       rationality;   /* rationality */
        Pint            order;         /* curve order */
        Pint            approx_type    /* approximation type */
        Pfloat          approx_val     /* approximation value */
        Pfloat_list     knots;         /* curve knot vector */
        Pfloat          tmin, tmax;    /* curve parameter range */
        Ppoint_list23   cpts;          /* control points */
} Ptrimcurve;
```

Pedge_flag is defined in phigs.h as:

```
typedef enum {
```

```
        PEDGE_OFF,
        PEDGE_ON
} Pedge_flag;
```

Prational is defined in phigs.h as:

```
typedef enum {
        PNON_RATIONAL = 0,
        PRATIONAL = 1
} Prational;
```

Pfloat_list is defined in phigs.h as:

```
typedef struct {
        Pint    num_floats;     /* number of Pfloats in list */
        Pfloat  *floats;        /* list of floats */
} Pfloat_list;
```

Ppoint_list23 is defined in phigs.h as:

```
typedef struct {                                /* list of 2D or 3D
                                                   points */
        Pint                num_points;     /* number of points */
        union {
                Ppoint      *point2d;       /* array of 2D points */
                Ppoint3     *point3d;       /* array of 3D points */
        } points;
} Ppoint_list23;
```

The *point3d* member of the points union is used if the rationality member of Ptrimcurve is PRATIONAL; otherwise, the *point2d* member is used. Ppoint is defined in phigs.h as:

```
typedef struct {
        Pfloat  x;      /* x coordinate */
        Pfloat  y;      /* y coordinate */
} Ppoint;
```

Ppoint3 is defined in phigs.h as:

```
typedef struct {
        Pfloat  x;      /* x coordinate */
        Pfloat  y;      /* y coordinate */
        Pfloat  z;      /* z coordinate */
} Ppoint3;
```

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

UORDER, VORDER
Order of the surface in the *u* and *v* dimensions, respectively.

NUKNTS, NVKNTS
Number of knots in the *u* and *v* dimensions, respectively.

UKNOTS(NUKNTS)
Array of knots for the *u* dimension.

VKNOTS(NVKNTS)
Array of knots for the *v* dimension.

RATSEL
Surface rationality is either *Rational* or *Nonrational*. This integer specifies the rationality and hence the dimensionality of the control points. Valid values defined are:

PRAT *Rational*
PNRAT *Nonrational*

NUCPTS, NVCPTS
Number of control points in the *u* and *v* dimensions, respectively.

DIMU The declared second dimension of CPTS, the three-dimensional control points array.

USTART, VSTART
The *u* and *v* indices of the first array element used in the control points array. For example, CPTS(1, USTART, VSTART) is the *x* coordinate of the first control point of the surface. This allows the use of the CPTS subarray.

*CPTS(4, DIMU, *)*
Array of control points. The point coordinates are stored as:

CPTS(1, u, v) = *x Coordinate*
CPTS(2, u, v) = *y Coordinate*
CPTS(3, u, v) = *z Coordinate*
CPTS(4, u, v) = *w Coordinate*

The *w* coordinate is not used if the surface is Nonrational, that is, RATSEL = PNRAT.

NLOOPS
Number of trimming curve loops.

LOOPS(NLOOPS)
Number of trimming curves in each loop.

Note: In the array dimensions below, *nt* refers to the total number of trim curves.

VIS(nt) Array of visibility flags, one entry for each trimming curve.

ORDER(nt)
Array of curve orders, one entry for each trimming curve.

TATYPE(nt)

Array of trim curve approximation types, one entry for each trimming curve. The curve approximation types defined in phigs77.h are:

PCAWSD	*Workstation-dependent*
PCAEPK	*Constant parametric between knots*
PCACSW	*Chordal size in WC space*
PCACSN	*Chordal size in NPC space*
PCACSD	*Chordal size in DC space*
PCACDW	*Chordal deviation in WC space*
PCACDN	*Chordal deviation in NPC space*
PCACDD	*Chordal deviation in DC space*

TAVAL(nt)

Array of trim curve approximation values, one entry for each trimming curve. The meaning of these values is dependent on the approximation type.

TRATS(nt)

Array of trimming curve rationalities, one entry for each trimming curve. Each entry is an integer specifying the rationality and hence the dimensionality of the control points. Valid values defined are:

PRAT	*Rational*
PNRAT	*Nonrational*

NTKNTS(nt)

Number of knots in each curve, one entry for each trimming curve.

TKSTRT(nt)

Start positions of the knot lists in the array TKNOTS, one entry for each trimming curve.

TKNOTS(*)

Array of knots for all trimming curves. Each curve uses a contiguous segment of the array. The index of the head of the array segment used by a curve is specified in the TKSTRT array.

NTCPTS(nt)

Number of control points in each curve, one entry for each trimming curve.

TCSTRT(nt)

Start positions of the control point lists in the TCPTS array, one entry for each trimming curve.

*TCPTS(3, *)*

Array of control points for all trimming curves. Each curve uses a contiguous segment of the array. The index of the head of the array segment used by a curve is specified in the TCSTRT array. The point coordinates are stored in each column of the array as:

TCPTS(1, t) = *u Coordinate*
TCPTS(2, t) = *v Coordinate*

TCPTS(3, t) = *w Coordinate*

The *w* coordinate is not used if the curve is Nonrational, that is, TRATS(t) = PNRAT.

TMIN(nt), TMAX(nt)

Arrays of parameter limits for the trimming curves, one entry in each array for each trimming curve.

Execution

Depending on the edit mode, a NON-UNIFORM B-SPLINE SURFACE element is either inserted into the open structure after the element pointer, or replaces the element pointed at by the element pointer. The element pointer is then updated to point to this NON-UNIFORM B-SPLINE SURFACE structure element.

The surface *u* and *v* orders must be positive integers. Surfaces of unsupported orders in either dimension are displayed by drawing the control grid, that is, the corresponding surface of order 2 in both dimensions.

The *u* and *v* knots must each form a non-decreasing sequence of numbers.

The *rationality selector* parameter may have the enumerated value Rational or Nonrational. When Rational is specified, the control points must be specified as 4D homogeneous modelling coordinates. When Nonrational is specified, the control points are 3D PHIGS modelling coordinates.

The number of control points in the *u* and *v* directions must be at least as large as the corresponding order. The number of control points in each direction plus the corresponding spline order must be equal to the corresponding number of knots.

At structure traversal time, a NON-UNIFORM B-SPLINE SURFACE primitive is drawn using the attributes that apply to FILL AREA SET 3 primitives, plus the extended interior attributes. See SET INTERIOR REPRESENTATION PLUS for a listing of these attributes.

Trimming Curves

The surface parameter area to be rendered may be defined by trimming curves. Trimming curves override the specified surface parameter range values, and the parameter range values are ignored during traversal if trimming curves are specified.

The trimming definition of a surface consists of a list of loops. Each of these loops is a list of one or more trimming
curves. Each trimming curve is a non-uniform rational B-spline curve defined in the 2D parameter space of the surface. The curves of a loop connect in a head-to-tail fashion. Each loop must be explicitly closed. Closure is assumed and not checked.

The restrictions on trimming curves are the same as those for non-uniform B-spline curves, and are specified in the function NON-UNIFORM B-SPLINE CURVE. Only the trimming curve's control polygon is used when it is of unsupported order or of order 1.

When the PCURV_WS_DEP approximation type is used for a trimming curve, the trimming curve tesselation is controlled by the tesselation of the surface. In this case, an approximation value between 0.0 and 1.0 indicates the relative quality of the trimming curve compared to surface quality. Higher values mean better relative quality.

Trimming loops may not go outside the parameter space of the surface. The trimming curves associated with a surface may touch each other only at their end points. No other pair of trimming curves may connect at that same point. A trimming curve may not be self-touching or self-intersecting.

If trimming curves are specified, then they define the edges of the surface. Each trimming curve has a flag that controls its visibility. These visibility flags have lower precedence than the edge flag attribute and are ignored if the edge flag attribute is set to OFF.

SunPHIGS Extensions restricts trimming loops of a single surface primitive to a certain canonical form. This form must redundantly identify the interior parameter area in two different ways: according to the odd winding and the curve handedness rules.

Attributes Applied

The attributes listed below are used to display the NON-UNIFORM B-SPLINE SURFACE primitive when the structure is traversed.

The Aspect Source Flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

interior colour	interior colour index ASF
back interior colour	back interior colour ASF
interior style	interior style ASF
back interior style	back interior style ASF
interior style index	interior style index ASF
back interior style index	back interior style index ASF
interior shading method	interior shading method ASF
back interior shading method	back interior shading method ASF
interior reflectance characteristics	interior reflectance characteristics ASF
back interior reflectance characteristics	back interior reflectance characteristics ASF
reflectance properties	reflectance properties ASF
back reflectance properties	back reflectance properties ASF
interior index	
edge colour	edge colour index ASF
edge flag	edge flag ASF
edgetype	edgetype ASF
edgewidth scale factor	edgewidth scale factor ASF
surface approximation criteria	surface approximation criteria ASF
edge index	
face distinguishing mode	
face culling mode	
depth cue index	

light source state
name set

ERRORS

005	Ignoring function, function requires state (PHOP, *, STOP, *)
600	Ignoring function, not enough control points for specified order
601	Ignoring function, knot sequence is not nondecreasing
602	Ignoring function, order is inconsistent with number of knots and control points specified
615	Ignoring function, parameter range is inconsistent with knots

SEE ALSO

SET SURFACE APPROXIMATION CRITERIA (3PP)
NON-UNIFORM B-SPLINE CURVE (3PP)
SET INTERIOR REPRESENTATION PLUS (3PP)
INQUIRE CURVE AND SURFACE FACILITIES (3PP)
INTRO (3PP)
GENERALIZED DRAWING PRIMITIVE 3 -19 (3P)

NAME

OPEN XPHIGS – open and initialize PHIGS in the X environment

SYNOPSIS

C Syntax

```
void
popen_xphigs ( error_file, memory, xinfo_mask, xinfo )
char            *error_file;    name of error file
Plong           memory;         not used
unsigned long   xinfo_mask;
Pxphigs_info    *xinfo;
```

FORTRAN Syntax

```
SUBROUTINE popenxphigs (ERRFIL, BUFA, DISPLAY, RMDB, NAME,
   APCLASS, ARGC, ARGV )
INTEGER             ERRFIL      error message file
INTEGER             BUFA        amount of memory units
INTEGER             DISPLAY     address of display
INTEGER             RMDB        address of resource database
CHARACTER*(*)       NAME        application name
CHARACTER*(*)       APCLASS     application class
INTEGER             ARGCP       address of argument count
INTEGER             ARGV        address of argument list
```

Required PHIGS Operating States

(PHCL, WSCL, STCL, ARCL)

DESCRIPTION

Purpose

OPEN XPHIGS is similar to OPEN PHIGS but allows specification of additional run-time options. It initializes the PHIGS environment and enables access to the PHIGS functions. OPEN PHIGS or OPEN XPHIGS must be called prior to calling any other PHIGS functions.

This function is not implemented in FORTRAN

C Input Parameters

error_file

A pointer to the *error file* in which to log PHIGS error messages. The *error file* can be either a pointer to a valid UNIX file name or a null pointer, for example (char*)0. A null pointer implies that standard error is to be used as the error file. When a file name is specified, PHIGS attempts to access the file for writing. If this attempt fails, then OPEN XPHIGS fails, and the appropriate error is reported to standard error.

The error file argument passed to OPEN XPHIGS is passed to ERROR HANDLING. ERROR HANDLING also passes this argument to ERROR LOGGING. If for some reason ERROR LOGGING cannot access the specified error file, then the error message is written to standard error. ERROR LOGGING appends messages to the error file; it does not truncate the file when OPEN XPHIGS is called. If the specified file does not exist, it is created only if ERROR LOGGING is called.

ERROR LOGGING writes the abstract PHIGS function name, the error number, and an error description to the error file. If for some reason the text for the function name and/or error description cannot be determined, then ERROR LOGGING simply writes the function number and the error number.

memory This parameter should be set to the constant PDEF_MEM_SIZE, which is defined in phigs.h.

xinfo_mask

A bitmask indicating which of the options are being set. This mask is a bitwise OR of one or more of the valid option mask bits and indicates which fields of the Pxphigs_info structure are set.

PXPHIGS_INFO_DISPLAY	*The display pointer: display*
PXPHIGS_INFO_RMDB	*The resource manager: rmdb*
PXPHIGS_INFO_APPL_ID	*The application name and class: appl_id*
PXPHIGS_INFO_ARGS	*The command line arguments: args*
PXPHIGS_INFO_FLAGS_NO_MON	*The No Monitor flag: flags.no_monitor*
PXPHIGS_INFO_FLAGS_CLIENT_SS	*The Force Client SS flag: flags.force_client_SS*

xinfo A pointer to a Pxphigs_info structure. This structure is used to specify X-related options to PHIGS. *xinfo* is defined in phigs.h as follows:

```
typedef struct {
    Display         *display;           /* valid display pointer */
    XrmDatabase     rmdb;               /* a valid database */
    struct {
        char        *name;
        char        *ap_class;
    } appl_id;                          /* for resolving database
                                           attributes */
    struct {
        int         *argc_p;
        char        **argv;
    } args;                             /* for merging args into specified
                                           database */
    struct {
        unsigned    no_monitor: 1;      /* 1 ==> monitor will not be
                                           executed */
        unsigned    force_client_SS;    /* 1 ==> always use client-side
                                           CSS */
    } flags;
} Pxphigs_info;
```

Only the fields indicated by *xinfo_mask* are examined.

display specifies both the PHIGS default server and the connection PHIGS is to use

when communicating with it. The PHIGS default server holds the master copy of the central structure store, if the following is true: it supports SunPHIGS structure storage and is the location where a *tool* workstation is opened if its location is not specified to the OPEN WORKSTATION function. PHIGS uses the specified connection for all communication to the default server, even if a different connection is specified for a *drawable* workstation in a subsequent call to OPEN WORKSTATION. PHIGS uses the specified connection for the duration of the PHIGS session; it must not be closed before calling CLOSE PHIGS. When *display* is not specified, the default server is returned by a call to XDisplayName() with an argument of NULL.

rmdb specifies an X resource database handle. PHIGS uses this database to build the default workstation description tables. Database searches are on the resource names and classes listed below. Each search is qualified by the name and class specified in *appl_id*. When *args* are also specified, PHIGS merges them into the database prior to searching the database for resources that PHIGS recognizes. Any merged arguments are removed from the argument list. The resources PHIGS recognizes are:

Argument String	***Resource Name***	***Resource Class***	***Type***	***Valid Values***
-display	display	Display	String	
-bufmode	bufMode	BufMode	String	single | double
=	geometry	Geometry	String	
-label	label	Label	String	
-iconlabel	iconLabel	IconLabel	String	

The *display* resource specifies the name of the default server. The *bufMode* resource specifies the default buffering mode, *single* or *double*. The *geometry* resource specifies the default window location and size for *phig_ws_type_x_tool* workstation types. The geometry is specified in the standard X geometry format:

<width>x<height>{+-}<xoffset>{+-}<yoffset>.

The *label* resource specifies the window label for *phig_ws_type_x_tool* workstation types. The *iconLabel* resource specifies the icon label for *phig_ws_type_x_tool* workstation types.

appl_id.name and *appl_id.ap_class* are the application name and class to use when resolving resource database attributes. If they are not specified, then the name *phigs* and class *Phigs* are used.

args.argc_p is a pointer to the argument count, *args.argv* is the array of command line arguments. The arguments are searched for attributes recognized by PHIGS and, if any are found, then the arguments are merged into the specified database.

flags – SunPHIGS does not use this element.

FORTRAN Input Parameters

ERRFIL

The FORTRAN Logical Unit Number of the error file to log PHIGS error messages to. Valid logical unit numbers are those that are:

- Associated with the standard preconnected files: standard error and standard output.
- Associated with a file that the application has opened prior to calling OPEN PHIGS.
- Valid with respect to the FORTRAN I/O system (but which are not yet associated with an open file).

PHIGS interaction with these various types of logical unit numbers is described in the *Execution* section below.

BUFA This parameter is ignored. –1 should be passed.

DISPLAY

The address of a valid Display structure, which specifies both the PHIGS default server and the connection PHIGS is to use when communicating with it. The PHIGS default server holds the master copy of the central structure store, if it supports SunPHIGS structure storage, and is where a tool workstation is opened when its location is not specified to the OPEN WORKSTATION function. PHIGS uses the specified connection for all communication to the default server, even if a different connection is specified for a drawable workstation in a subsequent call to OPEN WORKSTATION. PHIGS uses the specified connection for the duration of the PHIGS session; it must not be closed before calling CLOSE PHIGS. A value of 0 can be used, in which case PHIGS uses the default server returned by an XDisplayName() with a NULL argument.

RMDB An X resource database handle, which PHIGS uses to build the default workstation description tables. A value of 0 can be used if no resource database is to be used. When a database is specified, PHIGS uses this database to build the default workstation description tables. Database searches are on the resource names and classes listed below. Each search is qualified by the name and class specified in NAME and APCLASS. When args are also specified, PHIGS merges them into the database prior to searching the database for resources that PHIGS recognizes. The resources PHIGS recognizes are:

Argument String	***Resource Name***	***Resource Class***	***Type***	***Valid Values***
-display	display	Display	String	
-bufmode	bufMode	BufMode	String	single | double
=	geometry	Geometry	String	
-label	label	Label	String	
-iconlabel	iconLabel	IconLabel	String	

The *display* resource specifies the name of the default server. The *bufMode* resource specifies the default buffering mode, *single* or *double*. The *geometry* resource specifies the default window location and size for *phig_ws_type_x_tool* workstation types. The geometry is specified in the standard X geometry format:

<width>x<height>{+-}<xoffset>{+-}<yoffset>.

The *label* resource specifies the window label for *phig_ws_type_x_tool* workstation types. The *iconLabel* resource specifies the icon label for *phig_ws_type_x_tool*

workstation types.

NAME A character array containing the application name to be used when resolving resource database attributes. A null string can be specified, in which case the name *phigs* is used.

APCLASS
A character array containing the application class to be used when resolving resource database attributes. A null string can be specified, in which case the name *Phigs* is used.

ARGCP, ARGV
These parameters are provided for programs that use SunPHIGS in FORTRAN, but obtain a C-style argument count and argument list. ARGCP is an integer set to the address of the *argc* argument count. ARGV is the array of *argv* command line arguments. The arguments are searched for attributes recognized by PHIGS and, if any are found, then the arguments are merged into the specified database. A value of 0 should be used for ARGCP when no argument list is available, in which case ARGV is ignored.

Execution

OPEN_XPHIGS sets internal state information and then calls popen_phigs().

ERRORS

001	Ignoring function, function requires state (PHCL, WSCL, STCL, ARCL)
450	Ignoring function, the specified error file is invalid
–200	Ignoring function, cannot connect to the designated or default server
–201	Ignoring function, the specified or default X server does not support a compatible PHIGS extension

SEE ALSO

INQUIRE SYSTEM STATE VALUE (3P)
OPEN PHIGS (3P)
CLOSE PHIGS (3P)

NAME

POLYLINE SET 3 WITH DATA – creates a 3D polyline set structure element that includes colour and shading data

SYNOPSIS

C Syntax

void
ppolyline_set3_data (vflag, colr_type, npl, vdata)

Pint	vflag;	*data per vertex flag*
Pint	colr_type;	*colour type*
Pint	npl;	*number of polylines in the set*
Pline_vdata_list3	*vdata;	*per line vertex data list*

FORTRAN Syntax

SUBROUTINE pplsd3 (VFLAG, COLTYP, NPL, PLARR, NV, COORDS, IVCOLR, VCOLR, IMPL)

INTEGER	VFLAG	*data per vertex flag*
INTEGER	COLTYP	*colour type*
INTEGER	NPL	*number of point lists*
INTEGER	PLARR(NPL)	*array of end indices for point lists*
INTEGER	NV	*number of vertices*
REAL	COORDS(3,NV)	*array of vertex coordinates*
INTEGER	IVCOLR(NV)	*array of indirect vertex colours*
REAL	VCOLR(3,NV)	*array of vertex colours*
REAL	IMPL(*)	*implementation defined vertex data*

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

POLYLINE SET 3 WITH DATA creates a 3D polyline set primitive with colour and shading data.

A 3D polyline with data primitive is a set of connected lines in a 3D space defined by a series of Modeling Coordinate (MC) points. The element can optionally specify colour information for each vertex of the primitive.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

vflag The data per vertex flag specifies the available data for each vertex of the primitive.

0	PVERT_COORD	*Coordinates Specified*
1	PVERT_COORD_COLOUR	*Coordinates and Colours Specified*

colr_type

The colour type for specified vertex colours.

0	PINDIRECT	*Indirect*
1	PMODEL_RGB	*Red, Green, and Blue*

2	PMODEL_CIELUV	*CIE Colour Model*
3	PMODEL_HSV	*Hue, Saturation, and Value*
4	PMODEL_HLS	*Hue, Lightness, and Saturation*

npl The number of polylines in the set.

vdata A pointer to an array of *npl* Pline_vdata_list3 structures that specifies the list of vertices and optionally associated colour information. Pline_vdata_list3 is defined in phigs.h as follows:

```
typedef struct {
        Pint                num_vertices;   /* number of vertices */
        Pline_vdata_arr3    vertex_data;    /* line vertex data */
} Pline_vdata_list3;
```

Pline_vdata_arr3 is defined in phigs.h as follows:

```
typedef union {
        Ppoint3   *points;    /* array of points */
        Pptco3    *ptcolrs;   /* array of points with colours */
        /* implementation-dependent data */
}Pline_vdata_arr3;
```

Ppoint3 is defined in phigs.h as follows:

```
typedef struct {
        Pfloat    x;    /* x coordinate */
        Pfloat    y;    /* y coordinate */
        Pfloat    z;    /* z coordinate */
} Ppoint3;
```

Pptco3 is defined in phigs.h as follows:

```
typedef struct {
        Ppoint3   point;    /* point coordinates */
        Pcoval    colr;     /* colour */
} Pptco3;
```

Ppoint3 is defined above. Pcoval is defined in phigs.h as follows:

```
typedef union {
        Pint          ind;      /* index in workstation colour bundle
                                   table */
        Pcolr_rep     direct;   /* direct colour components */
} Pcoval;
```

When *colr_type* is PINDIRECT, *ind* is used; otherwise, *direct* is used.

Pcolr_rep is defined in phigs.h as follows:

```
typedef union {
```

```
        Prgb        rgb;        /* Red Green Blue colour
                                   specification */
        Pcieluv     cieluv;     /* CIE L*U*V*
                                   colour specification */
        Phls        hls;        /* Hue Lightness Saturation colour
                                   specification */
        Phsv        hsv;        /* Hue Saturation Value colour
                                   specification */
        Pdata       unsupp;     /* Colour in unsupported colour
                                   model */
} Pcolr_rep;
```

Prgb is defined in phigs.h as follows:

```
typedef struct {
        Pfloat      red;        /* red, hue, and so on */
        Pfloat      green;      /* green, saturation, lightness,
                                   and so on */
        Pfloat      blue;       /* blue, value, saturation, and
                                   so on */
} Prgb;
```

Pcieluv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat      cieluv_x;       /* x coefficient */
        Pfloat      cieluv_y;       /* y coefficient */
        Pfloat      cieluv_y_lum;   /* y luminance */
} Pcieluv;
```

Phls is defined in phigs.h as follows:

```
typedef struct {
        Pfloat      hue;        /* hue */
        Pfloat      lightness;  /* lightness */
        Pfloat      satur;      /* saturation */
} Phls;
```

Phsv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat      hue;        /* hue */
        Pfloat      satur;      /* saturation */
        Pfloat      value;      /* value */
} Phsv;
```

Pdata is defined in phigs.h as follows:

```
typedef struct {
```

```
        size_t      size;       /* size of data */
        char        *data       /* pointer to data */
    } Pdata;
```

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

VFLAG The data per vertex flag indicates the information specified with each vertex of the primitive. The possible values are:

0	PVCOOR	*Coordinates Only*
1	PVCLR	*Coordinates and Colours*

COLTYP

The colour type. If the data per vertex flag *(VFLAG)* indicates that a colour per vertex is specified, then *COLTYP* is set to one of the following:

0	PINDIR	*Colour Index Specified*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

NPL The number of point lists (polylines) in the polyline set.

PLARR(NPL)

An array of integers containing the end indices into the COORDS array for each point list in the polyline set.

NV The number of points used to define the polyline set.

COORDS(3,NV)

An array of reals containing the *x*, *y*, and *z* coordinates of the polyline set.

COORDS(1,v) = *x Coordinate*
COORDS(2,v) = *y Coordinate*
COORDS(3,v) = *z Coordinate*

IVCOLR(NV)

An array of integers containing an index into the workstation colour table for each vertex. This array is unused if VFLAG is set to PVCOOR or if COLTYP is greater than zero.

VCOLR(3,NV)

An array of reals containing the colours for each vertex. This array is unused VFLAG is set to PVCOOR or COLTYP is equal to zero (PINDIR).

VCOLR(1,v) = *Red, Hue, and so on*
VCOLR(2,v) = *Green, Saturation, Lightness, and so on*
VCOLR(3,v) = *Blue, Value, Saturation, and so on*

IMPL(*)

This implementation defined data type is not currently used.

Execution

Depending on the edit mode, a POLYLINE SET 3 WITH DATA element is inserted into the open structure after the element pointer or replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the newly created POLYLINE SET 3 WITH DATA element.

When the structure is traversed, POLYLINE SET 3 WITH DATA generates a set of unconnected polylines defined by a set of points. Each point sequence in the set generates connected line segments. Colour information for shading the line segments can be specified with each point as part of the primitive definition.

POLYLINE SET 3 WITH DATA elements containing less than two points can be created, but are ignored during structure traversal.

Attributes Applied

The attributes listed below are used to display the POLYLINE SET 3 WITH DATA primitive when the structure is traversed. The Aspect Source Flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, by using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

polyline colour	polyline colour index ASF
linewidth scale factor	linewidth scale factor ASF
linetype	linetype ASF
polyline shading method	polyline shading method ASF
polyline index	
depth cue index	
name set	

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

900 Storage overflow has occurred in PHIGS

2004 FORTRAN-specific binding error: ignoring function, input parameter size out of range

SEE ALSO

INTRO (3PP)
INQUIRE POLYLINE FACILITIES PLUS (3PP)
POLYLINE 3 (3P)

NAME

QUADRILATERAL MESH 3 WITH DATA – creates a 3D quadrilateral mesh structure element that includes colour and shading data

SYNOPSIS

C Syntax

void
pquad_mesh3_data (fflag, vflag, colr_type, dim, fdata, vdata)

Pint	fflag;	*what data per fill facet is available*
Pint	vflag;	*what data per vertex is available*
Pint	colr_type;	*colour type*
Pdims	*dim;	*dimension of cells (in vertices)*
Pfacet_data_arr3	*fdata;	*facet data*
Pfacet_vdata_arr3	*vdata;	*vertex data*

FORTRAN Syntax

SUBROUTINE pmed3 (FFLAG, VFLAG, COLTYP, VROWS, VCOLS, IFCOLR, FCOLR, FNORM, COORDS, IVCOLR, VCOLR, VNORM, IMPL)

INTEGER	FFLAG	*data per facet flag*
INTEGER	VFLAG	*data per vertex flag*
INTEGER	COLTYP	*colour type*
INTEGER	VROWS	*number of vertices along y axis*
INTEGER	VCOLS	*number of vertices along x axis*
INTEGER	IFCOLR((VROWS-1)*(VCOLS-1))	*array of indirect facet colours*
REAL	FCOLR(3,((VROWS-1)*(VCOLS-1)))	*array of facet colours*
REAL	FNORM(3,((VROWS-1)*(VCOLS-1)))	*array of facet normals*
REAL	COORDS(3,VROWS*VCOLS)	*array of vertex coordinates*
INTEGER	IVCOLR(VROWS*VCOLS)	*array of indirect vertex colours*
REAL	VCOLR(3,VROWS*VCOLS)	*array of vertex colours*
REAL	VNORM(3,VROWS*VCOLS)	*array of vertex normals*
REAL	IMPL(*)	*implementation defined vertex data*

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

QUADRILATERAL MESH 3 WITH DATA creates a 3D quadrilateral mesh primitive with colour and shading data.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

fflag The data per facet flag specifies the available data for each quadrilateral of the quadrilateral mesh.

0	PFACET_NONE	*No Facet Data Specified*
1	PFACET_COLOUR	*Facet Colours Specified*
2	PFACET_NORMAL	*Facet Normal Specified*

	3	PFACET_COLOUR_NORMAL	*Facet Normal and Colours Specified*

vflag The data per vertex flag specifies the available data for each vertex of the primitive.

0	PVERT_COORD	*Coordinates Specified*
1	PVERT_COORD_COLOUR	*Coordinates and Colours Specified*
2	PVERT_COORD_NORMAL	*Coordinates and Vertex Normal Specified*
3	PVERT_COORD_COLOUR_NORMAL	*Coordinates, Vertex Colour, and Vertex Normal Specified*

colr_type

The colour type for specified vertex colours.

0	PINDIRECT	*Colour Index Specified*
1	PMODEL_RGB	*Red, Green, and Blue*
2	PMODEL_CIELUV	*CIE Colour Model*
3	PMODEL_HSV	*Hue, Saturation, and Value*
4	PMODEL_HLS	*Hue, Lightness, and Saturation*

dim A pointer to a Pint_size structure that specifies the number of vertices along the *x* and *y* coordinates. Pint_size is defined in phigs.h as follows:

```
typedef struct {
        Pint    size_x;    /* number of vertices along the x axis */
        Pint    size_y;    /* number of vertices along the y axis */
} Pint_size;
```

fdata A pointer to a Pfacet_data_arr3 structure that specifies the colour and / or normal information for each facet. Note that number of facets = (size_x –1) * (size_y –1). Pfacet_data_arr3 is defined in phigs.h as follows:

```
typedef union {
    Pcoval      *colrs;     /* array (size_x –1) * (size_y –1) of colours */
    Pvec3       *norms;     /* array (size_x –1) * (size_y –1) of unit
                               normals */
    Pconorm3    *conorms;   /* array (size_x –1) * (size_y –1) of colours and
                               normals */
} Pfacet_data_arr3;
```

Pcoval is defined in phigs.h as follows:

```
typedef union {
        Pint        ind;      /* index in workstation colour bundle table */
        Pcolr_rep   direct;   /* direct colour components */
} Pcoval;
```

When *colr_type* is PINDIRECT, index is used; otherwise, direct is used.

Pcolr_rep is defined in phigs.h as follows:

```
typedef union {
        Prgb        rgb;        /* Red Green Blue colour specification */
        Pcieluv     cieluv;     /* CIE L*U*V* colour specification */
        Phls        hls;        /* Hue Lightness Saturation colour
                                   specification */
        Phsv        hsv;        /* Hue Saturation Value colour
                                   specification */
        Pdata       unsupp;     /* Colour in unsupported colour model */
} Pcolr_rep;
```

Prgb is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          red;            /* red, hue, and so on */
        Pfloat          green;          /* green, saturation, lightness,
                                           and so on */
        Pfloat          blue;           /* blue, value, saturation, and
                                           so on */
} Prgb;
```

Pcieluv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          cieluv_x;       /* x coefficient */
        Pfloat          cieluv_y;       /* y coefficient */
        Pfloat          cieluv_y_lum;   /* y luminance */
} Pcieluv;
```

Phls is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          lightness;      /* lightness */
        Pfloat          satur;          /* saturation */
} Phls;
```

Phsv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          satur;          /* saturation */
        Pfloat          value;          /* value */
} Phsv;
```

Pdata is defined in phigs.h as follows:

```
typedef struct {
        size_t          size;           /* size of data */
        char            *data           /* pointer to data */
```

```
} Pdata;
```

Pvec3 is defined as:

```
typedef struct {
        Pfloat    delta_x;    /* x magnitude */
        Pfloat    delta_y;    /* y magnitude */
        Pfloat    delta_z;    /* z magnitude */
} Pvec3;
```

Pconorm3 is defined as:

```
typedef struct {
        Pcoval    colr;       /* colour */
        Pvec3     norm;       /* unit normal */
} Pconorm3;
```

Pcoval and Pvec3 are defined above.

vdata A pointer to a Pfacet_vdata_arr3 structure that specifies the quadrilateral mesh vertices and optionally associated colour and normal information. Pfacet_vdata_arr3 is defined as:

```
typedef union {
   Ppoint3       *points;       /* array of (size_x × size_y) of points */
   Pptco3        *ptcolrs;      /* array of (size_x × size_y) of points and
                                   colours */
   Pptnorm3      *ptnorms;      /* array of (size_x × size_y) of points and
                                   normals */
   Pptconorm3    *ptconorms;    /* array of (size_x × size_y) of points, colours,
                                   and normals */
   /* implementation-dependent types can go here */
} Pfacet_vdata_arr3;
```

Ppoint3 is defined as:

```
typedef struct {
        Pfloat    x;    /* x coordinate */
        Pfloat    y;    /* y coordinate */
        Pfloat    z;    /* z coordinate */
} Ppoint3;
```

Pptco3 is defined as:

```
typedef struct {
        Ppoint3   point;    /* point coordinates */
        Pcoval    colr;     /* colour */
} Pptco3;
```

Ppoint3 is defined above. Pcoval is defined as:

```
typedef union {
        Pint        ind;      /* index in workstation colour bundle
                                 table */
        Pcolr_rep   direct;   /* direct colour components */
} Pcoval;
```

Pcolr_rep is defined above. Pptnorm3 is defined as:

```
typedef struct {
        Ppoint3 point;    /* point coordinates */
        Pvec3   norm;     /* unit normal */
} Pptnorm3;
```

Ppoint3 and Pvec3 are defined above. Pptconorm3 is defined as:

```
typedef struct {
        Ppoint3   point;    /* point coordinates */
        Pcoval    colr;     /* colour */
        Pvec3     norm;     /* unit normal */
} Pptconorm3;
```

Ppoint3, Pcoval, and Pvec3 are defined above.

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

Note: The following arrays are row major: IFCOLR, FCOLR, FNORM, COORDS, IVCOLR, VCOLR, and VNORM.

FFLAG The data per facet flag indicates the information specified for each facet of the quadrilateral mesh. The possible values are:

0	PFNONE	*No Facet Data Specified*
1	PFCLR	*Facet Colour Specified*
2	PFNORM	*Facet Normal Specified*
3	PFCLRN	*Facet Colour and Normal Specified*

VFLAG The data per vertex flag indicates the information specified with each vertex of the primitive. The possible values are:

0	PVCOOR	*Coordinates Only*
1	PVCLR	*Coordinates and Colours*
2	PVNORM	*Coordinates and Normals*
3	PVCLRN	*Coordinates, Colours, and Normals*

COLTYP

The colour type. If the data per facet flag *(FFLAG)* indicates that a facet colour is specified, or if the data per vertex flag *(VFLAG)* indicates that vertex colours are specified, then *COLTYP* is set to one of the following:

0	PINDIR	*Indirect*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*

3 PHSV *Hue, Saturation, and Value*
4 PHLS *Hue, Lightness, and Saturation*

VROWS The number of vertices along the *y* axis of the quadrilateral strip.

VCOLS The number of vertices along the *x* axis of the quadrilateral strip.

Note: The total number of vertices in the quadrilateral strip is the product of the number of vertices along the *y* axis and the number of vertices along the *x* axis (VROWS × VCOLS). The number of quadrilateral facets produced is (VROWS-1) × (VCOLS-1).

IFCOLR((VROWS-1)∗(VCOLS-1))
An array of integers containing indices into the workstation colour table for each facet or quadrilateral. The number of facets is equal to (VROWS-1) × (VCOLS-1). This array is unused if FFLAG is set to either PFNONE or PFNORM or if COLTYP is greater than zero.

FCOLR(3,((VROWS-1)∗(VCOLS-1)))
An array of reals containing the colours for each facet. The number of facets is equal to (VROWS-1) × (VCOLS-1). This array is unused if FFLAG is set to either PFNONE or PFNORM or COLTYP is equal to zero.

FCOLR(1,f) = *Red, Hue, and so on*
FCOLR(2,f) = *Green, Saturation, Lightness, and so on*
FCOLR(3,f) = *Blue, Value, Saturation, and so on*

FNORM(3,((VROWS-1)∗(VCOLS-1)))
An array of reals containing the unit normals for each facet. The number of facets is equal to (VROWS-1) × (VCOLS-1). This array is unused if FFLAG is set to either PFNONE or PFCLR.

FNORM(1,f) = *x Magnitude*
FNORM(2,f) = *y Magnitude*
FNORM(3,f) = *z Magnitude*

COORDS(3,VROWS∗VCOLS)
An array of reals containing the *x, y,* and *z* coordinates of the quadrilaterals, where the data in each set of four neighboring array elements defines a quadrilateral.

COORDS(1,v) = *x Coordinate*
COORDS(2,v) = *y Coordinate*
COORDS(3,v) = *z Coordinate*

IVCOLR(VROWS∗VCOLS)
An array of integers containing indices into the workstation colour table for each vertex. This array is unused if VFLAG is set to either PVCOOR or PVNORM or if COLTYP is greater than zero.

VCOLR(3,VROWS∗VCOLS)
An array of reals containing the colours for each vertex. This array is unused if VFLAG is set to either PVCOOR or PVERTCOORNORMAL or COLTYP is equal to zero.

VCOLR(1,v) = *Red, Hue, and so on*

VCOLR(2,v) = *Green, Saturation, Lightness, and so on*
VCOLR(3,v) = *Blue, Value, Saturation, and so on*

VNORM(3,VROWS∗VCOLS)
An array of reals containing the unit normals for each vertex. This array is unused if VFLAG is set to either PVCOOR or PVCLR.

VNORM(1,v) = *x Magnitude*
VNORM(2,v) = *y Magnitude*
VNORM(3,v) = *z Magnitude*

IMPL(∗)
Implementation defined vertex data.

Execution

Depending on the edit mode, a QUADRILATERAL MESH 3 WITH DATA element is inserted into the open structure after the element pointer, or replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the newly created QUADRILATERAL MESH 3 WITH DATA element.

When the structure is traversed, QUADRILATERAL MESH 3 WITH DATA generates a three-dimensional mesh of $(m-1) \times (n-1)$ quadrilaterals from a two dimensional array of $m \times n$ vertices. Line segments form the boundary of each quadrilateral in the mesh. Each quadrilateral can be rendered empty, hollow, shaded, or filled with a colour, pattern, or hatch style. A nonplanar quadrilateral is decomposed into two triangles when its interior is rendered. A mesh with $n < 2$ or $m < 2$ vertices can be created, but is ignored during traversal.

QUADRILATERAL MESH 3 WITH DATA uses the same attributes as the FILL AREA SET 3 WITH DATA primitive.

The QUADRILATERAL MESH 3 WITH DATA element can specify the facet colour and normal vector of each quadrilateral in the mesh, as well as colours and normal vectors for each vertex. These colours and normal vectors are used in conjunction with the current lighting and depth-cueing attributes to colour and shade the primitive.

Note: Unexpected lighting effects result when the supplied normals are not of unit length.

INTRO (3PP) describes how to use the output display attribute values.

Attributes Applied

The attributes listed below are used to display the QUADRILATERAL MESH 3 WITH DATA primitive when the structure is traversed. The Aspect Source Flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

interior colour	interior colour index ASF
back interior colour	back interior colour ASF
interior style	interior style ASF
back interior style	back interior style ASF
interior style index	interior style index ASF
back interior style index	back interior style index ASF

interior shading method	interior shading method ASF
back interior shading method	back interior shading method ASF
interior reflectance characteristics	interior reflectance characteristics ASF
back interior reflectance characteristics	back interior reflectance characteristics ASF
reflectance properties	reflectance properties ASF
back reflectance properties	back reflectance properties ASF
interior index	
edge colour	edge colour index ASF
edge flag	edge flag ASF
edgetype	edgetype ASF
edgewidth scale factor	edgewidth scale factor ASF
edge index	
face distinguishing mode	
face culling mode	
depth cue index	
light source state	
name set	

ERRORS

005	Ignoring function, function requires state (PHOP, *, STOP, *)
900	Storage overflow has occurred in PHIGS
2004	FORTRAN-specific binding error: ignoring function, input parameter size out of range

SEE ALSO

FILL AREA SET 3 WITH DATA (3PP)
INTRO (3PP)

NAME

SET BACK INTERIOR COLOUR – create a structure element to set the back interior colour

SYNOPSIS

C Syntax

```
void
pset_back_int_colr ( colour )
Pgcolr    *colour;    back interior colour
```

FORTRAN Syntax

```
SUBROUTINE psbico ( COLMOD, ICOLR, COLR )
INTEGER   COLMOD    colour model
INTEGER   ICOLR     colour table index
REAL      COLR(3)   colour triple
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET BACK INTERIOR COLOUR creates a structure element to set the back interior colour.

When the current back interior colour Aspect Source Flag (ASF) is set to INDIVIDUAL, the current value of this attribute defines the back interior colour to be applied to the following output primitives:

FILL AREA
FILL AREA 3
FILL AREA SET
FILL AREA SET 3
FILL AREA SET 3 WITH DATA
NON-UNIFORM B-SPLINE SURFACE
SET OF FILL AREA SET 3 WITH DATA
QUADRILATERAL MESH 3 WITH DATA
TRIANGLE STRIP 3 WITH DATA

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameter

The following data types are predefined in phigs.h.

colour The colour model. Pgcolr is defined as:

```
typedef struct {
   Pint          type;          /* indirect, RGB, CIE, HSV, HLS */
   union {
     Pint        index;         /* colour table index */
     struct {
          Pfloat       x;      /* red, hue, and so on */
          Pfloat       y;      /* green, saturation, lightness, and so on */
          Pfloat       z;      /* blue, value, saturation, and so on */
```

```
        } general;
      } val;
    } Pgcolr;
```

FORTRAN Input Parameters

The following data types are predefined in phigs77.h.

COLMOD

The colour model. The predefined values are:

0	PINDIR	*Colour Index Specified*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

ICOLR An integer index into the workstation colour table for determining the back interior colour. This value is not used if COLMOD is greater than zero.

COLR(3)

An array of reals containing the back interior colour. This array is unused if COLMOD is equal to zero or PINDIR.

COLR(1) = *Red, Hue, and so on*
COLR(2) = *Green, Saturation, Lightness, and so on*
COLR(3) = *Blue, Value, Saturation, and so on*

Execution

Depending on the edit mode, a SET BACK INTERIOR COLOUR element with the value back interior colour in the specified colour model is inserted into the open structure after the element pointer or replaces the element pointed at by the element pointer. The element pointer is then updated to point to this SET BACK INTERIOR COLOUR element.

Back face attributes (such as back area properties, back interior colour, back interior reflectance equation, back interior shading method, back interior style, and back interior style index) are used only when the face distinguishing mode has the value *Yes* and are applied to only the back facing portion of area defining primitives. See SET FACE DISTINGUISHING MODE (3PP) for details on front and back face determination.

When the current back interior colour ASF is set to INDIVIDUAL, area-defining output primitives, which follow in the structure network and are subject to back face attributes, are filled using the current back interior colour.

When the current back interior colour ASF is set to BUNDLED, the back interior colour is taken from the workstation extended representation indicated by the current interior index. In this case, the back interior colour set with SET BACK INTERIOR COLOUR has no effect.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

SET INTERIOR COLOUR (3PP)
SET FACE DISTINGUISHING MODE (3PP)
SET INDIVIDUAL ASF (3P)
SET INTERIOR REPRESENTATION PLUS (3PP)

NAME

SET BACK INTERIOR REFLECTANCE EQUATION – create structure element to set current back interior reflectance equation

SYNOPSIS

C Syntax

```
void
pset_back_refl_eqn ( equation )
Pint    equation;    back reflectance equation
```

FORTRAN Syntax

```
SUBROUTINE psbire ( REFLEQ )
INTEGER    REFLEQ    back reflectance equation
```

Required PHIGS Operating States

(PHOP, ∗, STOP, ∗)

DESCRIPTION

Purpose

SET BACK INTERIOR REFLECTANCE EQUATION creates a structure element to set the current back interior reflectance equation.

When the current back interior reflectance equation ASF is set to INDIVIDUAL, the current value of this attribute applies to the following output primitives:

FILL AREA
FILL AREA 3
FILL AREA SET
FILL AREA SET 3
FILL AREA SET 3 WITH DATA
NON-UNIFORM B-SPLINE SURFACE
SET OF FILL AREA SET 3 WITH DATA
QUADRILATERAL MESH 3 WITH DATA
TRIANGLE STRIP 3 WITH DATA

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameter

equation

The reflectance equation. The values defined in phigs.h are:

1	PREFL_NONE	*No Reflectance Calculation Performed*
2	PREFL_AMBIENT	*Use Ambient Term*
3	PREFL_AMB_DIFF	*Use Ambient and Diffuse Terms*
4	PREFL_AMB_DIFF_SPEC	*Use Ambient, Diffuse, and Specular Terms*

FORTRAN Input Parameter

The following data types are predefined in phigs77.h.

REFLEQ

The reflectance equation. The values are:

1	PRNONE	*No Reflectance Calculation Performed*
2	PRAMB	*Use Ambient Term*

3 PRAMBD *Use Ambient and Diffuse Terms*
4 PRAMDS *Use Ambient, Diffuse, and Specular Terms*

Execution

Depending on the edit mode, a SET BACK INTERIOR REFLECTANCE EQUATION element with the value reflectance equation is inserted into the open structure after the element pointer or replaces the element pointed at by the element pointer. The element pointer is then updated to point to this SET BACK INTERIOR REFLECTANCE EQUATION element.

At traversal time, if the specified reflectance equation is not defined, then equation one is substituted.

Back face attributes (such as back area properties, back interior colour, back interior reflectance equation, back interior shading method, back interior style, and back interior style index) are used only when the face distinguishing mode has the value *Yes* and are applied to only the back facing portion of area defining primitives. See SET FACE DISTINGUISHING MODE (3PP) for details on front and back face determination.

When the current back interior reflectance equation ASF is set to INDIVIDUAL, area-defining output primitives which follow in the structure network are filled using the current back interior reflectance equation.

When the current back interior reflectance equation ASF is set to BUNDLED, the back interior reflectance equation is taken from the workstation extended representation indicated by the current interior index. In this case, the back interior reflectance equation set with SET BACK INTERIOR REFLECTANCE EQUATION has no effect.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

SET FACE DISTINGUISHING MODE (3PP)
SET INDIVIDUAL ASF (3P)
SET INTERIOR REPRESENTATION PLUS (3PP)
SET REFLECTANCE EQUATION (3PP)

NAME

SET BACK INTERIOR SHADING METHOD – create structure element to set back interior shading method

SYNOPSIS

C Syntax

void
pset_back_int_shad_meth (method)
Pint method; *back shading method*

FORTRAN Syntax

SUBROUTINE psbism (METHOD)
INTEGER METHOD *back interior shading method*

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET BACK INTERIOR SHADING METHOD creates a structure element to set the back interior shading method.

When the current back interior shading method ASF is set to INDIVIDUAL, the current value of this attribute applies to the following output primitives:

FILL AREA
FILL AREA 3
FILL AREA SET
FILL AREA SET 3
FILL AREA SET 3 WITH DATA
NON-UNIFORM B-SPLINE SURFACE
SET OF FILL AREA SET 3 WITH DATA
QUADRILATERAL MESH 3 WITH DATA
TRIANGLE STRIP 3 WITH DATA

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameter

method The back interior shading method. The predefined values are:

1	PSD_NONE	*No Shading*
2	PSD_COLOUR	*Colour Interpolation Shading*
3	PSD_DOT_PRODUCT	*Dot Product Interpolation Shading*
4	PSD_NORMAL	*Normal Interpolation Shading*

FORTRAN Input Parameter

The following data types are predefined in phigs77.h.

METHOD

The value of the shading method. The predefined values are:

1	PSDNON	*No Shading*
2	PSDCOL	*Colour Interpolation Shading*
3	PSDDOT	*Dot Product Interpolation Shading*
4	PSDNOR	*Normal Interpolation Shading*

Execution

Depending on the edit mode, a SET BACK INTERIOR SHADING METHOD element with the value shading method is inserted into the open structure after the element pointer or replaces the element pointed at by the element pointer. The element pointer is then updated to point to this SET BACK INTERIOR SHADING METHOD element.

When the structure is traversed, the current interior shading method entry in the traversal state list is set to the specified value. This value is used with subsequent SunPHIGS Extension output primitives when the current interior shading method ASF entry in the traversal state list is INDIVIDUAL. During traversal, if the specified shading method is not available on a workstation, method one (*None*) is used on that workstation.

The interior shading method supported by a workstation type can be determined with the function INQUIRE INTERIOR FACILITIES PLUS.

Back face attributes (such as back area properties, back interior colour, back interior reflectance equation, back interior shading method, back interior style, and back interior style index) are used only when the face distinguishing mode has the value *Yes* and are applied to only the back facing portion of area defining primitives. See SET FACE DISTINGUISHING MODE (3PP) for details on front and back face determination.

When the current back interior shading method ASF is set to INDIVIDUAL, area-defining output primitives, which follow in the structure network and are subject to back face attributes, are filled using the current back interior shading method.

When the current back interior shading method ASF is set to BUNDLED, the back interior shading method is taken from the workstation extended representation indicated by the current interior index. In this case, the back interior shading method set with SET BACK INTERIOR SHADING METHOD has no effect.

SunPHIGS does not support the $ *Dot Product Interpolation Shading* and $ *Normal Interpolation Shading* methods.$ When one of these shading methods is specified, $ the backup NONE is used.$

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

INQUIRE INTERIOR FACILITIES PLUS (3PP)
SET FACE DISTINGUISHING MODE (3PP)
SET INDIVIDUAL ASF (3P)
SET INTERIOR REPRESENTATION PLUS (3PP)
SET INTERIOR SHADING METHOD (3PP)

NAME

SET BACK INTERIOR STYLE – create structure element to set back interior style

SYNOPSIS

C Syntax

```
void
pset_back_int_style ( style )
Pint_style    style;    back interior style
```

FORTRAN Syntax

```
SUBROUTINE psbis ( STYLE )
INTEGER    STYLE    back interior style
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET BACK INTERIOR STYLE creates a structure element to set the back interior style. SET BACK INTERIOR STYLE determines in what style the interior of the back facing area defining primitives should be filled: *Hollow, Solid, Pattern, Hatch, Empty,* or *General*.

When the current back interior style Aspect Source Flag (ASF) is set to INDIVIDUAL, the current value of this attribute defines the back interior style to be applied to the following output primitives:

FILL AREA
FILL AREA 3
FILL AREA SET
FILL AREA SET 3
FILL AREA SET 3 WITH DATA
NON-UNIFORM B-SPLINE SURFACE
SET OF FILL AREA SET 3 WITH DATA
QUADRILATERAL MESH 3 WITH DATA
TRIANGLE STRIP 3 WITH DATA

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameter

style A pointer to a Pint_style data structure. A Pint_style structure is defined in phigs.h as:

```
typedef enum {
        PHOLLOW,    Hollow
        PSOLID,     Solid
        PPATTERN,   Patterned
        PHATCH,     Hatched
        PEMPTY,     Empty
} Pint_style;
```

FORTRAN Input Parameter

The following data types are predefined in phigs77.h.

STYLE The back interior style. Predefined values are:

0	PHOLLO	*Hollow*
1	PSOLID	*Solid*
2	PPATTR	*Patterned*
3	PHATCH	*Hatched*
4	PISEMP	*Empty*

Execution

Depending on the edit mode, a SET BACK INTERIOR STYLE element with the value interior style is inserted into the open structure after the element pointer or replaces the element pointed at by the element pointer. The element pointer is then updated to point to this SET BACK INTERIOR STYLE element.

The back interior style is used to determine in what style the interior of the back facing area defining primitives should be filled:

Hollow No filling, but draw the boundary using the colour, or colours, resulting from the rendering pipeline. The linetype and linewidth of this bounding polyline are implementation-dependent. SunPHIGS presently uses a solid line of default width.

Solid Fill the interior using the colour, or colours, resulting from the rendering pipeline.

Pattern Fill the interior using the colours resulting from the rendering pipeline. The current *back interior style index* is used to select a pattern from the pattern table. In this context, the back interior style index is sometimes referred to as the *back pattern index* and must have a value greater than zero.

Hatch Fill the interior using the colours resulting from the rendering pipeline. The current back interior style index is used to select a hatch style from the hatch table. In this context, the back interior style index is sometimes referred to as the back hatch index.

Empty No filling is performed. Area defining primitives drawn using this value are visible only if the EDGE FLAG is set to ON.

If the requested interior style is not available on a workstation, then *Hollow* is used on that workstation.

Back face attributes (such as back area properties, back interior colour, back interior reflectance equation, back interior shading method, back interior style, and back interior style index) are used only when the face distinguishing mode has the value *Yes* and are applied only to the back facing portion of area defining primitives. See SET FACE DISTINGUISHING MODE (3PP) for details on front and back face determination.

When the current back interior style ASF is set to INDIVIDUAL, area-defining output primitives, which follow in the structure network and are subject to back face attributes, are filled using the current back interior style.

When the current back interior style ASF is set to BUNDLED, the back interior style is taken from the workstation extended representation indicated by the current interior index. In this case, the back interior style set with SET BACK INTERIOR STYLE has no effect.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

SET FACE DISTINGUISHING MODE (3PP)
SET BACK INTERIOR STYLE INDEX (3PP)
SET INDIVIDUAL ASF (3P)
SET INTERIOR REPRESENTATION PLUS (3PP)
SET INTERIOR STYLE (3P)

NAME

SET BACK INTERIOR STYLE INDEX – create structure element to set back interior style index

SYNOPSIS

C Syntax

void
pset_back_int_style_ind (index)
Pint index; *back interior style index*

FORTRAN Syntax

SUBROUTINE psbisi (ISTYLE)
INTEGER ISTYLE *back interior style index*

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET BACK INTERIOR STYLE INDEX creates a structure element to set the back interior style index.

When the current back interior style index Aspect Source Flag (ASF) is set to INDIVIDUAL, the current value of this attribute defines the back interior style index to be applied to the following output primitives:

FILL AREA
FILL AREA 3
FILL AREA SET
FILL AREA SET 3
FILL AREA SET 3 WITH DATA
NON-UNIFORM B-SPLINE SURFACE
SET OF FILL AREA SET 3 WITH DATA
QUADRILATERAL MESH 3 WITH DATA
TRIANGLE STRIP 3 WITH DATA

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameter

index The back interior style index.

FORTRAN Input Parameter

ISTYLE The back interior style index.

Execution

Depending on the edit mode, a SET BACK INTERIOR STYLE INDEX element with the value interior style index is inserted into the open structure after the element pointer or replaces the element pointed at by the element pointer. The element pointer is then updated to point to this SET BACK INTERIOR STYLE INDEX element.

During traversal, the back interior style index is only used for area defining facets that are back facing. For interior styles *Hollow* and *Solid*, the back interior style index value is not used. For interior style *Pattern*, this value is greater than zero and is a pointer into the

pattern table of the workstation. For interior style *Hatch*, this value determines which of a number of hatch styles is used. Hatch styles greater than zero are reserved for registration. Hatch styles less than one are workstation-dependent.

At traversal time, if the entry associated with the specified interior style index is not defined, then interior style index one is substituted.

Back face attributes (such as back area properties, back interior colour, back interior reflectance equation, back interior shading method, back interior style, and back interior style index) are used only when the face distinguishing mode has the value *Yes* and are applied only to the back facing portion of area defining primitives. See SET FACE DISTINGUISHING MODE (3PP) for details on front and back face determination.

When the current back interior style index ASF is set to INDIVIDUAL, area-defining output primitives, which follow in the structure network and are subject to back face attributes, are filled using the current back interior style index.

When the current back interior style index ASF is set to BUNDLED, the back interior style index is taken from the workstation extended representation indicated by the current interior index. In this case, the back interior style index set with SET BACK INTERIOR STYLE INDEX has no effect.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

SET FACE DISTINGUISHING MODE (3PP)
SET BACK INTERIOR STYLE (3PP)
SET INDIVIDUAL ASF (3P)
SET INTERIOR REPRESENTATION PLUS (3PP)
SET INTERIOR STYLE INDEX (3P)

NAME

SET BACK REFLECTANCE PROPERTIES – create structure element to set back area properties

SYNOPSIS

C Syntax

```
void
pset_back_refl_props ( properties )
Prefl_props    *properties;    back surface area properties
```

FORTRAN Syntax

```
SUBROUTINE psbap ( AMB, DIF, SPEC, COLTYPE, ICOLR, COLR, EXP, TRANS )
REAL       AMB       ambient reflectance coefficient
REAL       DIF       diffuse reflectance coefficient
REAL       SPEC      specular reflectance coefficient
INTEGER    COLTYPE   colour type
INTEGER    ICOLR     indirect colour
REAL       COLR(3)   specular colour
REAL       EXP       specular exponent
REAL       TRANS     transparency coefficient
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET BACK REFLECTANCE PROPERTIES creates a structure element to set the back area properties.

When the current back area properties ASF is set to INDIVIDUAL, the current value of this attribute applies to the following output primitives:

FILL AREA
FILL AREA 3
FILL AREA SET
FILL AREA SET 3
FILL AREA SET 3 WITH DATA
NON-UNIFORM B-SPLINE SURFACE
SET OF FILL AREA SET 3 WITH DATA
QUADRILATERAL MESH 3 WITH DATA
TRIANGLE STRIP 3 WITH DATA

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameter

properties

A pointer to a Prefl_props structure that specifies the back area properties. Prefl_props is defined in phigs.h as follows:

```
typedef struct {
        Pfloat    ambient_coef;    /* ambient reflectance coefficient */
```

```
        Pfloat    diffuse_coef;     /* diffuse reflectance coefficient */
        Pfloat    specular_coef;    /* specular reflectance coefficient */
        Pgcolr    specular_colr;    /* specular colour */
        Pfloat    specular_exp;     /* specular exponent */
        Pfloat    transpar_coef;    /* transparency coefficient */
} Prefl_props;
```

Pgcolr is defined as:

```
typedef struct {
   Pint           type;            /* indirect, RGB, CIE, HSV, HLS */
   union {
     Pint         ind;             /* colour table index */
     struct {
          Pfloat        x;        /* red, hue, and so on */
          Pfloat        y;        /* green, saturation, lightness, and so on */
          Pfloat        z;        /* blue, value, saturation, and so on */
     } general;
   } val;
} Pgcolr;
```

The color type is one of the following:

0	PINDIRECT	*Indirect*
1	PMODEL_RGB	*Red, Green, and Blue*
2	PMODEL_CIELUV	*CIE Colour Model*
3	PMODEL_HSV	*Hue, Saturation, and Value*
4	PMODEL_HLS	*Hue, Lightness, and Saturation*

FORTRAN Input Parameters

The following data types are predefined in phigs77.h.

AMB The ambient reflectance coefficient.

DIF The diffuse reflectance coefficient.

SPEC The specular reflectance coefficient.

COLTYPE

The colour type. The predefined values are:

0	PINDIR	*Indirect*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

ICOLR An integer value that is an index into the workstation colour table for determining the specular colour. ICOLR is used if COLTYPE is set to PINDIR.

COLR(3)

The specular colour. An array of reals containing the specular colour. COLR(3) is

used if COLTYPE is set to other than PINDIR.

COLR(1) = *Red, Hue, and so on*
COLR(2) = *Green, Saturation, Lightness, and so on*
COLR(3) = *Blue, Value, Saturation, and so on*

EXP The specular exponent.

TRANS The transparency coefficient.

Execution

Depending on the edit mode, a SET BACK REFLECTANCE PROPERTIES element is inserted into the open structure after the element pointer or replaces the element pointed at by the element pointer. The element pointer is then updated to point to this SET BACK REFLECTANCE PROPERTIES element.

Back face attributes (such as back area properties, back interior colour, back interior reflectance equation, back interior shading method, back interior style, and back interior style index) are used only when the face distinguishing mode has the value *Yes* and are applied only to the back facing portion of area defining primitives. See SET FACE DISTINGUISHING MODE (3PP) for details on front and back face determination.

When the current back area properties ASF is set to INDIVIDUAL, area-defining output primitives, which follow in the structure network and are subject to back face attributes, are filled using the current back area properties.

When the current back area properties ASF is set to BUNDLED, the back area properties are taken from the workstation extended representation indicated by the current interior index. In this case, the back area properties set with SET BACK REFLECTANCE PROPERTIES has no effect.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

SET REFLECTANCE PROPERTIES (3PP)
SET FACE DISTINGUISHING MODE (3PP)
SET INDIVIDUAL ASF (3P)
SET INTERIOR REPRESENTATION PLUS (3PP)
SET FACE CULLING MODE (3PP)

NAME

SET COLOUR MAPPING INDEX – create a structure element to set the colour mapping index

SYNOPSIS

C Syntax

```
void
pset_colr_map_ind ( index )
Pint    index;    colour mapping index
```

FORTRAN Syntax

```
SUBROUTINE pscmi ( CMI )
INTEGER    CMI    colour mapping index
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET COLOUR MAPPING INDEX creates an element in the currently open structure containing a colour mapping index. The index number corresponds to an entry in the table of defined colour mapping representations in the workstation state list.

C Input Parameter

index An index into the workstation colour mapping bundle table.

FORTRAN Input Parameter

CMI An index into the workstation colour mapping bundle table.

Execution

Depending on the edit mode, a SET COLOUR MAPPING INDEX element with a value colour mapping index is inserted into the open structure after the element pointer, or it replaces the element to which the element pointer points. The element pointer is then updated to point to this SET COLOUR MAPPING INDEX element.

When a SET COLOUR MAPPING INDEX structure element is traversed, the current colour mapping index value in the PHIGS Extension traversal state list is set to index. This value indexes a colour mapping representation in the table of defined colour mapping operations stored in the workstation state list. If the entry corresponding to the specified index is not defined, then an entry of zero is substituted in the index.

The current colour mapping index determines the colour mapping operation that is applied to all primitives subsequently displayed.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

121 Ignoring function, the colour mapping index is less than 0.

SEE ALSO

SET COLOUR MAPPING REPRESENTATION (3PP)
INQUIRE LIST OF COLOUR MAPPING INDICES (3PP)

NAME

SET COLOUR MAPPING REPRESENTATION – set colour mapping representation entry in the workstation table of defined colour mapping representations

SYNOPSIS

C Syntax

```
void
pset_colr_map_rep ( ws, index, map_method, map_data )
Pint            ws;          workstation identifier
Pint            index;       colour mapping bundle index
Pint            map_method;  mapping method
Pcolr_map_data  *map_data;   mapping method data record
```

FORTRAN Syntax

```
SUBROUTINE pscmr ( WKID, CMRI, MMETH, LDR, DATREC )
INTEGER          WKID          workstation identifier
INTEGER          CMRI          predefined index
INTEGER          MMETH         colour mapping method
INTEGER          LDR           dimension of data record array
CHARACTER*80     DATREC(LDR)   data record
```

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

SET COLOUR MAPPING REPRESENTATION defines or sets a colour mapping representation in the specified workstation's colour mapping table.

C Input Parameters

ws The workstation identifier specifies the workstation for which the colour mapping representation is to be defined.

index The index of the colour mapping representation to set.

map_method

The colour mapping method. Colour mapping methods are defined in phigs.h as follows:

1 PCOLR_MAP_TRUE
2 PCOLR_MAP_PSEUDO
3 PCOLR_MAP_PSEUDO_N

C Output Parameters

map_data

A pointer to a pointer to a Pcolr_map_data structure that contains the information specific to the data mapping method. Pcolr_map_data is defined in phigs.h as follows:

```
typedef union {
        struct {
                Pint            colr_model;
                Pfloat_list     weights;
```

```
            Pcolr_rep_list    colrs;
        } method_r2
        struct {
            Pint              colr_model;
            Pfloat_list_list  colr_lists;
        } method_r3
} Pcolr_map_data;
```

Pfloat_list is defined in phigs.h as follows:

```
typedef struct {
        Pint        num_floats;     /* number of Pfloats in list */
        Pfloat      *floats;        /* list of floats *
} Pfloat_list;
```

Pfloat_list_list is defined in phigs.h as follows:

```
typedef struct {
        Pint        num_lists;      /* number of lists in list */
        Pfloat_list *lists;         /* list of float lists */
} Pfloat_list_list;
```

Pcolr_rep_list is defined in phigs.h as follows:

```
typedef struct {
        Pint        num_colr_reps;  /* number of colours */
        Pcolr_rep   *colr_reps;     /* array of colours */
} Pcolr_rep_list;
```

Pcolr_rep is defined in phigs.h as follows:

```
typedef union {
        Prgb        rgb;        /* Red Green Blue colour specification */
        Pcieluv     cieluv;     /* CIE L*U*V* colour specification */
        Phls        hls;        /* Hue Lightness Saturation colour
                                   specification */
        Phsv        hsv;        /* Hue Saturation Value colour
                                   specification */
        Pdata       unsupp;     /* Colour in unsupported colour model */
} Pcolr_rep;
```

Prgb is defined in phigs.h as follows:

```
typedef struct {
        Pfloat      red;        /* red, hue, and so on */
        Pfloat      green;      /* green, saturation, lightness,
                                   and so on */
        Pfloat      blue;       /* blue, value, saturation,
                                   and so on */
```

```
} Prgb;
```

Pcieluv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          cieluv_x;       /* x coefficient */
        Pfloat          cieluv_y;       /* y coefficient */
        Pfloat          cieluv_y_lum;   /* y luminance */
} Pcieluv;
```

Phsv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          satur;          /* saturation */
        Pfloat          value;          /* value */
} Phsv;
```

Phls is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          lightness;      /* lightness */
        Pfloat          satur;          /* saturation */
} Phls;
```

Pdata is defined in phigs.h as follows:

```
typedef struct {
        size_t          size;           /* size of data */
        char            *data           /* pointer to data */
} Pdata;
```

There is no data associated with mapping method PCOLR_MAP_TRUE. The data for method PCOLR_MAP_PSEUDO are a list of weights and a list of direct colours. *colr_model* specifies the type of colours given in the list of colours, *colrs*. The colour models are defined in phigs.h as follows:

1 PMODEL_RGB
2 PMODEL_CIELUV
3 PMODEL_HSV
4 PMODEL_HLS

The data for mapping method PCOLR_MAP_PSEUDO_N is N lists of direct colours. These colours may be of the same types as for PCOLR_MAP_PSEUDO.

FORTRAN Input Parameters

WKID The workstation identifier of the workstation for which the colour mapping representation is to be defined.

CMRI The index of the colour mapping representation to set.

MMETH

The colour mapping method. Colour mapping methods are defined in phigs77.h as follows:

1 PCMAPTRUE
2 PCMAPPSEUDO
3 PCMAPPSEUDON

LDR The dimension of the data record array, DATREC.

DATREC(LDR)

The data record is packed by PACK DATA RECORD. The contents depend on the mapping method specified:

For MMETH = PCMAPTRUE:

IL = the number of integers, 0

RL = the number of real values, 0

SL = the number of strings, 0

For MMETH = PCMAPPSEUDO:

IL = the number of integers, 3

IA = an array of integers, with the following values:

IA(1) = the colour model. Valid values defined in phigs77.h are:

1 PRGB
2 PCIE
3 PHSV
4 PHLS

IA(2) = the number of weights.

IA(3) = the number of colour representations.

RL = the number of reals, IA(2) + 3 * IA(3).

RA = an array of reals with the following values:

RA(1) – RA(IA(2)) =
the weights RA(IA(2)+1) – RA(IA(2) + 3 + IA(3)) = the colour values for each colour representation (r1, g1, b1, r2, g2, b2, and so on).

SL = the number of strings, 0

For MMETH = PCMAPPSEUDON:

IL = the number of integers, 2 + (number of lists).

IA = an array of integers, with the following values:

IA(1) = the colour model. Valid values are defined in phigs77.h as follows:

1 PRGB
2 PCIE

3 PHSV
4 PHLS

IA(2) = the number of lists of float values.

IA(3) – IA(IL) = a list of end indices in RA for each list of colour coordinates.

RL = the number of reals, equal to the total of the list lengths in IA(3) – IA(IL).

RA = a list of lists of floats, whose end indices are indicated in IA(3) through IA(IL).

Execution

When SET COLOUR MAPPING REPRESENTATION is called, the specified entry in the table of defined colour mapping representations of the specified workstation is set to the values specified.

During structure traversal, the current colour mapping representation is selected from the workstation table by a structure element created by the SET COLOUR MAPPING INDEX subroutine.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, the specified workstation is not open

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT or OUTIN)

103 Ignoring function, setting this bundle table entry would exceed the maximum number of entries allowed in the workstation bundle table

110 Ignoring function, the specified colour model is not available on the workstation

121 Ignoring function, colour mapping index is less than 0.

125 Ignoring function, the total of the colour range fields in all the table entries is too large.

126 Ignoring function, the specified colour mapping method is not available on the specified workstation.

136 Ignoring function, one of the components of the colour specification is out of range

138 Ignoring function, one or more of the fields in the specified data record is inconsistent with the specified type.

SEE ALSO

PACK DATA RECORD (3P)
SET COLOUR MAPPING INDEX (3PP)
INQUIRE COLOUR MAPPING REPRESENTATION (3PP)
UNPACK DATA RECORD (3P)

NAME

SET CURVE APPROXIMATION CRITERIA – create a structure element to set the curve approximation criteria

SYNOPSIS

C Syntax

```
void
pset_curve_approx ( type, value )
Pint     type;     approximation type
Pfloat   value;    approximation value
```

FORTRAN Syntax

```
SUBROUTINE pscapx ( TYPE, VALUE )
INTEGER  TYPE      approximation type
REAL     VALUE     approximation value
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET CURVE APPROXIMATION CRITERIA places into the currently open structure a structure element that specifies the curve approximation type and value. The curve approximation type controls the display precision of non-uniform B-spline curves by specifying the method to use to render the curve.

When the current curve approximation criteria Aspect Source Flag (ASF) is set to INDIVIDUAL, the current value of this attribute defines the curve approximation criteria to be applied to the following output primitives:

GENERALIZED DRAWING PRIMITIVE
GENERALIZED DRAWING PRIMITIVE 3
NON-UNIFORM B-SPLINE CURVE

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

The following data types are predefined in phigs.h.

type The curve approximation type. Predefined values are:

0 PCURV_WS_DEP
1 PCURV_CONSTANT_PARAMETRIC
2 PCURV_CONSTANT_PARAMETRIC_BETWEEN_KNOTS
3 PCURV_METRIC_WC
4 PCURV_METRIC_NPC
5 PCURV_CHORDAL_DEVIATION_WC
6 PCURV_CHORDAL_DEVIATION_NPC

value The value to use when applying the specified approximation method. Valid values depend on the approximation method specified. See the *Execution* section below for a description of the use of *value* by each approximation method.

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

TYPE The curve approximation type. Predefined values are:

1	PCAWSD	*Workstation-dependent*
2	PCAEPK	*Constant Parametric Subdivision Between Knots*
3	PCACSW	*Chordal Size in WC*
4	PCACSN	*Chordal Size in NPC*
5	PCACSD	*Chordal Size in DC*
6	PCACDW	*Chordal Deviation in WC*
7	PCACDN	*Chordal Deviation in NPC*
8	PCACDD	*Chordal Deviation in DC*

VALUE The value to use when applying the specified approximation method. Valid values depend on the approximation type specified. See the *Execution* section below for a description of the use of .I "VALUE" by each approximation method.

Execution

Depending upon the edit mode, a SET CURVE APPROXIMATION CRITERIA element is inserted into the open structure after the element pointer, or replaces the element pointed at by the element pointer. The element pointer is then updated to point to this SET CURVE APPROXIMATION CRITERIA element.

The approximation methods corresponding to the defined approximation types and the use of the approximation value by each method are:

Workstation-dependent, Type 0

Use a workstation-dependent approximation method. Currently all SunPHIGS Extension workstation types use method 2 (Constant Parametric Subdivision Between Knots) as the workstation-dependent method. The *approximation value* has the same usage as for that method.

Constant Parametric Subdivision, Type 1

Tessellate using equal parametric increments along the entire curve. The integer portion of the approximation value is used and interpreted as follows:

≤ 0: Evaluate the curve only at the parameter limits (thus generating a single line per curve).

> 0: Evaluate the curve at the parameter limits and at the specified number of positions between the parameter limits.

Constant Parametric Subdivision Between Knots, Type 2

Tessellate using equal parametric increments between each pair of knots on the curve. The integer portion of the approximation value is used and interpreted as follows:

≤ 0: Evaluate the curve only at the parameter limits and at the knots that fall within the parameter limits.

> 0: Evaluate the curve at the parameter limits, at the knots that fall within the parameter limits, and at the specified number of positions between each pair of knots, but only if the position is within the

parameter limits.

Metric in WC, Type 3
Tessellate until the length in World Coordinates (WC) of every line segment (chord) representing the curve is less than the specified positive real-number approximation value.

Metric in NPC, Type 4
Tessellate until the length in Normalized Projection Coordinates (NPC) of every line segment (chord) representing the curve is less than the specified positive real-number approximation value.

SunPHIGS Extensions approximates this approximation type by a different adaptive method. The method used adjusts the tessellation of the curve according to the modelling and viewing transforms in effect when the curve is traversed. The approximation value corresponds to the maximum length of a tessellated segment, as a percentage of the workstation viewport. If an approximation value less than, or equal to, 0.0 is specified, then the value 0.01 (1 percent) is used.

Chordal Deviation in WC, Type 5
Tessellate until the absolute value of the maximum deviation measured in WC between the curve and approximating line segment (chord) is less than the specified positive real-number approximation value.

Chordal Deviation in NPC, Type 6
Tessellate until the absolute value of the maximum deviation measured in NPC between the curve and approximation line segment (chord) is less than the specified positive real-number approximating value.

SunPHIGS Extensions approximates this approximation type by a different adaptive method. The method used adjusts the tessellation of the curve according to the modelling and viewing transforms in effect when the curve is traversed. The approximation value corresponds to the maximum length of a tessellated segment, as a percentage of the workstation viewport. If an approximation value less than, or equal to, 0.0 is specified, then the value 0.01 (1 percent) is used.

At traversal time, approximation type zero (*Workstation-dependent*) and an approximation value of 1.0 are used if the specified approximation type is not supported by the workstation. If the approximation value is inconsistent with the specified approximation type, then a value of 1.0 is used. To determine the approximation types supported by a workstation type, see INQUIRE CURVE AND SURFACE FACILITIES, or the relevant entries in PHIGS WORKSTATION DESCRIPTION TABLE (7P).

When the current curve approximation criteria ASF is set to INDIVIDUAL, output primitives that follow in the structure network are filled using the current curve approximation criteria style.

When the current curve approximation criteria ASF is set to BUNDLED, the curve approximation criteria is taken from the workstation extended representation indicated by the current polyline index. In this case, the curve approximation criteria set with SET CURVE APPROXIMATION CRITERIA has no effect.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

NON-UNIFORM B-SPLINE CURVE (3PP)
INQUIRE CURVE AND SURFACE FACILITIES (3PP)
SET INDIVIDUAL ASF (3P)
INTRO (3PP)
GENERALIZED DRAWING PRIMITIVE (3P)
GENERALIZED DRAWING PRIMITIVE 3 (3P)
PHIGS WORKSTATION DESCRIPTION TABLE (7P)

NAME

SET DEPTH CUE INDEX – creates structure element containing value depth cue index

SYNOPSIS

C Syntax

```
void
pset_dcue_ind ( index )
Pint    index;    depth cue index
```

FORTRAN Syntax

```
SUBROUTINE psdci ( IDEPTH )
INTEGER    IDEPTH    depth cue index
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET DEPTH CUE INDEX creates an element in the currently open structure containing a depth cue index number. The index number corresponds to an entry in the table of defined depth cue operations in the workstation state list.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameter

index This is an index to the workstation depth cue bundle table.

The index number corresponds to an entry in the table of defined depth cue operations in the workstation state list. Up to 20 depth cue table entries may be defined. Entry zero is predefined to a depth cue mode of SUPPRESSED and cannot be changed. Entries other than zero can be changed with the SET DEPTH CUE REPRESENTATION subroutine.

FORTRAN Input Parameter

IDEPTH

This is an index to the workstation depth cue bundle table.

Execution

Depending on the edit mode, a SET DEPTH CUE INDEX element with a value depth cue index is inserted into the open structure after the element pointer or replaces the element pointed to by the element pointer. The element pointer is then updated to point to this SET DEPTH CUE INDEX element.

When a SET DEPTH CUE INDEX structure element is traversed, the current depth cue index value in the SunPHIGS Extension traversal state list is set to index. This value indexes a depth cue representation in the table of defined depth cue operations stored in the workstation state list. If the entry corresponding to the specified index is not defined, then an entry of zero is substituted in the index.

The current depth cue index determines the depth cue operation that will be applied to all primitives in the display. Each entry in the workstation table of defined depth cue operations contains the following information:

- depth cue index number
- depth cue mode (SUPPRESSED or ALLOWED)
- depth cue reference planes
- depth cue scale factors
- depth cue color

If the depth cue mode is SUPPRESSED, depth cueing is not applied to subsequent primitives in the display. If the depth cue mode is ALLOWED, depth cueing is applied to all subsequent primitives using the operation defined by the reference planes, scale factors, and depth cue color.

When a workstation is opened, Entries 0 and 1 are initialized with the following representation:

	Entry 0	*Entry 1*
depth cue mode	SUPPRESSED	ALLOWED
depth cue reference planes	0.0, 1.0	0.0, 1.0
depth cue scaling	1.0, 1.0	0.0, 1.0
depth cue colour	(INDIRECT, 0)	(INDIRECT, 0)

For depth cue reference planes and depth cue scaling, the first value is the back reference plane and scaling.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

119 Ignoring function, depth cue index value is less than zero

SEE ALSO

SET DEPTH CUE REPRESENTATION (3PP)
INQUIRE LIST OF DEPTH CUE INDICES (3PP)

NAME

SET DEPTH CUE REPRESENTATION – define depth cue representation entry in workstation table of defined depth cue representations

SYNOPSIS

C Syntax

```
void
pset_dcue_rep ( ws, index, rep )
Pint            ws;     workstation identifier
Pint            index;  depth cue bundle index
Pdcue_bundle    *rep;   depth cue representation pointer
```

FORTRAN Syntax

```
SUBROUTINE psdcr ( WKID, IDEPTH, MODE, BPLANE, FPLANE, BSCALE,
    FSCALE, COLTYPE, IDCOLR, DCOLR )
INTEGER   WKID            workstation identifier
INTEGER   IDEPTH          depth cue index
INTEGER   MODE            depth cue mode
REAL      BPLANE, FPLANE  back and front depth planes
REAL      BSCALE, FSCALE  back and front scale value
INTEGER   COLTYPE         colour type
INTEGER   IDCOLR          indirect depth cue colour
REAL      DCOLR(3)        array of depth cue colour
```

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

SET DEPTH CUE REPRESENTATION assigns parameter values defining a depth cue operation to a specified index number in the workstation table of defined depth cue representations. Each entry in this table contains a value for depth cue mode, reference planes, scaling, and colour. When enabled, depth cueing is applied to all output primitives subsequently displayed.

Depth cueing mixes a scaled portion of the depth cue colour with the primitive colour as a function of the *z* coordinate in NPC space. Using the background colour as the depth cue colour, this operation allows you to fade the colour of the parts of the primitives that are further from the viewer in the display.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

ws The workstation identifier specifies the workstation for which the depth cue representation is to be defined. The workstation must have been opened with the OPEN WORKSTATION subroutine before calling SET DEPTH CUE REPRESENTATION.

index The index to the workstation depth cue bundle table.

The index number corresponds to an entry in the table of defined depth cue operations in the workstation state list. Up to 20 depth cue table entries can be

defined. Entry zero is predefined to a depth cue mode of PSUPPRESSED and cannot be changed. Entries other than zero can be changed with the SET DEPTH CUE REPRESENTATION subroutine.

rep A pointer to a Pdcue_bundle data structure containing attribute values defining a depth cue representation. A Pdcue_bundle structure is defined in phigs.h as follows:

```
typedef struct {
    Pdcue_mode      mode;           /* depth cue mode */
    Pfloat          ref_planes[2];  /* depth cue reference planes */
    Pfloat          scaling[2];     /* depth cue scaling */
    Pgcolr          colr;           /* depth cue colour */
} Pdcue_bundle;
```

Pdcue_mode is defined as:

```
typedef enum {
        PSUPPRESSED,
        PALLOWED
} Pdcue_mode;
```

Refplanes and scaling entries of zero are back reference plane and corresponding scaling factors. Refplanes and scaling entries of one are front reference plane and corresponding scaling factors. These scaling factors define the portion of the primitive colour that should be combined with the depth cue colour, as a function of *z* in NPC.

Pgcolr is defined as:

```
typedef struct {
    Pint            type;       /* indirect, RGB, CIE, HSV, HLS */
    union {
        Pint        ind;        /* index in workstation colour bundle
                                   table */
        struct {
            Pfloat  x;          /* red, hue, and so on */
            Pfloat  y;          /* green, saturation, lightness,
                                   and so on */
            Pfloat  z;          /* blue, value, saturation, and so on */
        } general;
    } val;
} Pgcolr;
```

Constants defined for colour type are:

0	PINDIRECT	*Indirect*
1	PMODEL_RGB	*Red, Green, Blue*
2	PMODEL_CIELUV	*CIE*
3	PMODEL_HSV	*Hue, Saturation, Value*

4 PMODEL_HLS *Hue, Lightness, Saturation*

FORTRAN Input Parameters

The following data types are predefined in phigs77.h.

WKID The identifier of the workstation for which the depth cue representation is being defined.

IDEPTH

The depth cue index to the workstation depth cue table.

FORTRAN Output Parameters

MODE The depth cue *MODE* controls the disabling or enabling of depth cueing. The depth cue mode contains one of the following values:

0 PSUPPR *Disable Depth Cueing*
1 PALLOW *Enable Depth Cueing*

BPLANE,

The back and front depth planes. BPLANE is the back depth plane, a *z* value in NPC, which defines a back reference plane for depth cueing. FPLANE is the front depth plane, a *z* value in NPC, which defines a front reference plane for depth cueing.

BSCALE,

The back and front scale values. BSCALE is the back scale value, between 0.0 and 1.0, which corresponds to the *z* value of the back depth cue reference plane. This value defines the portion of the primitive colours that should be combined with the depth cue colour as a function of *z* in NPC. FSCALE is a front scale value, between 0.0 and 1.0, which corresponds to the *z* value of the front depth cue reference plane. This value defines the portion of the primitive colour that should be combined with the depth cue colour as a function of *z* in NPC.

COLTYPE

The colour type. If this value is PINDIR or zero, then IDCOLR contains an indexed colour value. If COLTYPE is greater than zero, then DCOLR contains the depth cue colour. The valid values for COLTYPE are:

0 PINDIR *Indirect*
1 PRGB *Red, Green, and Blue*
2 PCIE *CIE Colour Model*
3 PHSV *Hue, Saturation, and Value*
4 PHLS *Hue, Lightness, and Saturation*

IDCOLR

An integer index into the workstation colour table for determining the depth cue colour. This value is not used if COLTYPE is greater than zero.

DCOLR(3)

An array of reals containing the depth cue colour. This array is not used if COLTYPE is equal to zero. The reals are stored as follows:

DCOLR(1) = *Red, Hue, and so on*
DCOLR(2) = *Green, Saturation, Lightness, and so on*

DCOLR(3) = *Blue, Value, Saturation, and so on*

Execution

When a SET DEPTH CUE REPRESENTATION is called, the entry index in the table of defined depth cue representations on the workstation identifier is set to the values in *rep*.

During structure traversal, the current depth cue representation is selected from the workstation table by a structure element created by the SET DEPTH CUE INDEX subroutine. While the depth cue mode of the current representation is ALLOWED, the depth cue operation defined by the other representation parameters is applied to all the following primitives in the structure network.

The depth cueing operation changes the primitive colours in the display as a function of their *z* coordinate in NPC space. The first number in scaling specifies the portion of the primitive colour that is combined with the depth cue colour at, or in front of, the back reference plane. The second scaling value specifies the portion of primitive colour applied at, or behind, the front reference plane. The portion of depth cue colour applied to primitives between the two planes is scaled by a value linearly interpolated between the two scaling values.

When a workstation is opened, Entries 0 and 1 are initialized with the following representation:

	Entry 0	*Entry 1*
depth cue mode	SUPPRESSED	ALLOWED
depth cue reference planes	0.0, 1.0	0.0, 1.0
depth cue scaling	1.0, 1.0	0.0, 1.0
depth cue colour	(INDIRECT, 0)	(INDIRECT, 0)

For depth cue reference planes and depth cue scaling, the first value is the back reference plane and scaling.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, the specified workstation is not open

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

103 Ignoring function, setting this bundle table entry would exceed the maximum number of entries allowed in the workstation bundle table

110 Ignoring function, the specified colour model is not available on the workstation

120 Ignoring function, the depth cue index is less than one

130 Ignoring function, invalid reference planes; DQMIN > DQMAX

SEE ALSO

OPEN WORKSTATION (3P)

SET DEPTH CUE INDEX (3PP)
INQUIRE DEPTH CUE REPRESENTATION (3PP)

NAME

SET EDGE COLOUR – create structure element to set current edge direct colour attribute

SYNOPSIS

C Syntax

```
void
pset_edge_colr ( colour )
Pgcolr    *colour;    edge colour
```

FORTRAN Syntax

```
SUBROUTINE pseco ( COLMOD, ICOLR, COLR )
INTEGER   COLMOD    colour model
INTEGER   ICOLR     colour table index
REAL      COLR(3)   colour triple
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET EDGE COLOUR creates a structure element to set the current edge direct colour attribute.

When the current edge colour Aspect Source Flag (ASF) is set to INDIVIDUAL, the current value of this attribute applies to the following output primitives:

FILL AREA SET
FILL AREA SET 3
FILL AREA SET 3 WITH DATA
NON-UNIFORM B-SPLINE SURFACE
SET OF FILL AREA SET 3 WITH DATA
QUADRILATERAL MESH 3 WITH DATA
TRIANGLE STRIP 3 WITH DATA

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameter

colour Edge colour. Pgcolr is defined as:

```
typedef struct {
        Pint            type;           /* indirect, RGB, CIE, HSV, HLS */
        union {
          Pint          index;          /* colour table index */
          struct {
                Pfloat          x;      /* red, hue, and so on */
                Pfloat          y;      /* green, saturation, lightness,
                                           and so on */
                Pfloat          z;      /* blue, value, saturation, and
                                           so on */
          } general;
        } val;
```

```
} Pgcolr;
```

FORTRAN Input Parameters

The following data types are predefined in phigs77.h.

COLMOD

The colour model. The predefined values are:

0	PINDIR	*Indirect*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

ICOLR An integer index into the workstation colour table for determining the edge colour. This value is only used if COLMOD is equal to PINDIR.

COLR(3)

An array of reals containing the edge colour. This array is unused if COLMOD is equal to PINDIR.

COLR(1) = *Red, Hue, and so on*
COLR(2) = *Green, Saturation, Lightness, and so on*
COLR(3) = *Blue, Value, Saturation, and so on*

Execution

Depending on the edit mode, a SET EDGE COLOUR element with the value edge colour in the specified colour model is inserted into the open structure after the element pointer or replaces the element pointed at by the element pointer. The element pointer is then updated to point to this SET EDGE COLOUR element.

When the current edge colour ASF is set to INDIVIDUAL, filled area output primitives that follow in the structure network are drawn using the current edge colour.

When the current edge colour ASF is set to BUNDLED, the edge colour is taken from the workstation extended representation indicated by the current edge index. In this case, the edge colour set with SET EDGE COLOUR has no effect.

The standard PHIGS function SET EDGE COLOUR INDEX is equivalent to calling to SET EDGE COLOUR with type INDIRECT.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

COLOUR (7P)
SET EDGE REPRESENTATION PLUS (3PP)
SET EDGE COLOUR INDEX (3P)

NAME

SET EDGE REPRESENTATION PLUS – set extended edge attribute bundle on workstation

SYNOPSIS

C Syntax

```
void
pset_edge_rep_plus ( ws, index, rep )
Pint                ws;     workstation identifier
Pint                index;  edge bundle index number
Pedge_bundle_plus   *rep;   edge representation pointer
```

FORTRAN Syntax

```
SUBROUTINE pseer ( WKID, INDEX, EFLAG, ETYPE, EWIDTH, COLMOD,
    IECOLR, ECOLR )
INTEGER   WKID      workstation identifier
INTEGER   INDEX     edge bundle index
INTEGER   EFLAG     edge flag (PON/POFF)
INTEGER   ETYPE     edge type
REAL      EWIDTH    edge width scale factor
INTEGER   COLMOD    colour model
INTEGER   IECOLR    indirect colour index
REAL      ECOLR(3)  direct colour value
```

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

SET EDGE REPRESENTATION PLUS sets an extended edge attribute bundle on a workstation.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

ws The identifier of the workstation.

index This is an index to the workstation edge bundle table.

rep A pointer to a Pedge_bundle_plus structure. Pedge_bundle_plus is defined as:

```
typedef struct {
        Pedge_flag      flag;       /* edge flag */
        Pint            type;       /* edgetype */
        Pfloat          width;      /* edgewidth scale factor */
        Pgcolr          colr;       /* edge colour */
} Pedge_bundle_plus;
```

Pedge_flag is defined as:

```
typedef enum {
        PEDGE_OFF,
        PEDGE_ON
```

```
} Pedge_flag;
```

The edge type values are:

1	PLINE_SOLID	*Solid*
2	PLINE_DASH	*Dashed*
3	PLINE_DOT	*Dotted*
4	PLINE_DASH_DOT	*Dot-dashed*

Pgcolr is defined as:

```
typedef struct {
        Pint            type;       /* indirect, RGB, CIE, HSV, HLS */
        union {
           Pint         ind;        /* index in workstation colour
                                       bundle table */
           struct {
              Pfloat    x;          /* red, hue, and so on */
              Pfloat    y;          /* green, saturation, lightness,
                                       and so on */
              Pfloat    z;          /* blue, value, saturation, and
                                       so on */
           } general;
        } val;
} Pgcolr;
```

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

WKID The identifier of the workstation for which the edge representation is being defined.

INDEX The index into the specified workstation edge bundle table.

EFLAG The edge flag. Predefined values are:

0	POFF
1	PON

ETYPE The edge type. Predefined values are:

1	PLSOLI	*Solid*
2	PLDASH	*Dashed*
3	PLDOT	*Dotted*
4	PLDASD	*Dot-dashed*
0	PLNLONGDASH	*Long-dashed*
-1	PLNDOTDASHDOT	*Dot-dashed-dot-dotted*
-2	PLNCENTER	*Center (long-short dashed)*
-3	PLNPHANTOM	*Phantom (long-short-short dashed)*

EWIDTH

The edge width scale factor.

COLMOD

The colour model which determines the type of colour to be used. The valid values for *COLMOD* are:

0	PINDIR	*Indirect*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

IECOLR

An integer index into the workstation colour table for determining the edge colour. This value is only used if *COLMOD* is equal to PINDIR.

ECOLR(3)

An array of reals containing the edge colour. This array is not used if *COLMOD* is equal to PINDIR.

ECOLR(1) = *Red, Hue, and so on*
ECOLR(2) = *Green, Saturation, Lightness, and so on*
ECOLR(3) = *Blue, Value, Saturation, and so on*

Execution

In the edge bundle table of the workstation state list, the given index is associated with the specified parameters.

The edgewidth scale factor is a scale factor applied to nominal edgewidth. During structure traversal, the edgewidth scale factor is multiplied by the nominal edgewidth on a workstation. The result is mapped by the workstation to the nearest available edgewidth.

Edge colour is a general colour, either an indirect or a direct colour in a specified colour model. If an indirect colour is specified and the specified colour index is not available during structure traversal, then colour index one is used.

The extended edge bundle table in the workstation state list has predefined entries taken from the workstation description table. A number of these entries are predefined for every workstation of category OUTPUT or OUTIN. Any table entry, including the predefined entries, may be redefined with this function.

When area defining primitives are displayed during structure traversal, the current edge index in the PHIGS traversal state list refers to an entry in the extended edge bundle table. If the current edge index is not present in the extended edge bundle table, then edge index one is used. The setting of the corresponding Aspect Source Flags(ASFs) determines which aspects in the entry are used.

ERRORS

003	Ignoring function, function requires state (PHOP, WSOP, *, *)
054	Ignoring function, the specified workstation is not open
059	Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
100	Ignoring function, the bundle index value is less than one

103	Ignoring function, setting this bundle table entry would exceed the maximum number of entries allowed in the workstation bundle table
107	Ignoring function, the specified edgetype is not available on the specified workstation
110	Ignoring function, the specified colour model is not available on the workstation
113	Ignoring function, the colour index value is less than zero

SEE ALSO

SET INDIVIDUAL ASF (3P)
INQUIRE EDGE REPRESENTATION PLUS (3PP)
SET EDGE REPRESENTATION (3P)

NAME

SET ELEMENT POINTER AT PICK IDENTIFIER – set the element pointer to the next occurrence of the specified pick identifier

SYNOPSIS

C Syntax

```
void
pset_elem_ptr_pick_id ( pick_id )
Pint    pick_id;    pick identifier
```

FORTRAN Syntax

```
SUBROUTINE psepid ( PKID )
INTEGER    PKID    pick identifier
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET ELEMENT POINTER AT PICK IDENTIFIER sets the element pointer of the currently open structure to the next occurrence in the structure of an element containing the specified pick identifier. Pick identifier structure elements are created by the SET PICK IDENTIFIER subroutine.

This is a SunPHIGS Extension function based on customer requests and is not part of the PHIGS standard.

C Input Parameter

pick_id Specifies the pick identifier in the open structure at which to set the element pointer.

FORTRAN Input Parameter

PKID Specifies the pick identifier in the open structure at which to set the element pointer.

Execution

SET ELEMENT POINTER AT PICK IDENTIFIER searches for the pick identifier from the structure element following the current element pointer to the end of the open structure. If an occurrence of *pick identifier* is not found between the current element pointer and the end of the structure, the subroutine does not move the element pointer and generates an error.

ERRORS

–154 SunPHIGS-dependent: Ignoring function, the pick identifier does not exist in the open structure between the element pointer and the end of the structure

SEE ALSO

OFFSET ELEMENT POINTER (3P)
INQUIRE ELEMENT POINTER (3P)
INQUIRE CURRENT ELEMENT TYPE AND SIZE (3P)
SET ELEMENT POINTER (3P)
SET ELEMENT POINTER AT LABEL (3P)
ELEMENT SEARCH (3P)

NAME

SET FACE CULLING MODE – create structure element to set face culling mode

SYNOPSIS

C Syntax

```
void
pset_face_cull_mode ( mode )
Pcull_mode    mode;    culling mode
```

FORTRAN Syntax

```
SUBROUTINE psfcm ( MODE )
INTEGER    MODE    culling mode
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET FACE CULLING MODE creates a structure element to set the face culling mode. The attribute face culling mode defines control over the visibility of front and back facing portions of area defining primitives. See SET FACE DISTINGUISHING MODE (3PP) for details on front and back face determination.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

The following data types are predefined in phigs.h.

mode A pointer to a Pcull_mode structure. A Pcull_mode structure is defined as:

```
typedef enum {
        PCULL_NONE,
        PCULL_BACKFACE,
        PCULL_FRONTFACE
} Pcull_mode;
```

FORTRAN Input Parameters

The following data types are predefined in phigs77.h.

MODE The culling mode. Predefined values are:

0	PCULNO	*None*
1	PCULBF	*Backface Culling*
2	PCULFF	*Frontface Culling*

Execution

Depending on the edit mode, a SET FACE CULLING MODE element is inserted into the open structure after the element pointer or replaces the element pointed at by the element pointer. The element pointer is then updated to point to this SET FACE CULLING MODE element.

When face culling mode is *None*, both front and back facing portions are visible. When face culling mode is *Backface*, back facing portions are invisible. When it is *Frontface*, front facing portions are invisible.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

SET FACE DISTINGUISHING MODE (3PP)

NAME

SET FACE DISTINGUISHING MODE – create structure element to set face distinguishing mode

SYNOPSIS

C Syntax

void
pset_face_disting_mode (mode)
Pdisting_mode mode; *distinguishing mode*

FORTRAN Syntax

SUBROUTINE psfdm (MODE)
INTEGER MODE *distinguishing mode (YES/NO)*

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET FACE DISTINGUISHING MODE creates a structure element to set the face distinguishing mode. The attribute face distinguishing mode defines control over the attributes applied when rendering front-facing and back-facing portions of area-defining primitives.

A facet of an area-defining primitive is defined to be back-facing if its geometric normal has a negative *z* component when transformed to NPC; otherwise, the facet is front-facing.

If facet normals are supplied, they are used as geometric normals; otherwise, a geometric normal is computed for each facet using a method that depends on the primitive type:

Fill Area or *Fill Area Set*

A geometric normal is computed by first determining three points: A, B, and C. Point A is the first of the vertices. Point B is the next point in the list that is non-coincident with A. Point C is the next point in the list (after B) that is non-colinear with A and B. The geometric normal is the cross-product of the vector extending from A to B with the vector extending from A to C. The utility function COMPUTE FILL AREA SET GEOMETRIC NORMAL (3PP) can be used to determine the geometric normal for a fill area set primitive according to the above procedure.

Triangle Strip Primitive

Determination of the geometric normal for each triangle is computed as for a fill area, using the vertices of the triangle. Order of vertices in triangles is reversed for even numbered triangles, with triangle numbering starting at 1.

Quadrilateral Mesh Primitive

Determination of the geometric normal for each quadrilateral is calculated by taking the cross-product of its diagonals.

Non-uniform B-Spline Surface

The geometric normal is computed as the cross-product of the tangent vector along the *u* direction and the tangent vector along the *v* direction.

Notice that when the geometric normal is computed, the order of points determines the sense of direction — that is, whether it is front-facing or back-facing.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

The following data types are predefined in phigs.h.

mode A pointer to a Pdisting_mode structure. A Pdisting_mode structure is defined as:

```
typedef enum {
        PDISTING_NO,
        PDISTING_YES
} Pdisting_mode;
```

FORTRAN Input Parameters

The following data types are predefined in phigs77.h.

MODE The distinguishing mode. Predefined values are:

0	PNDIST	*No distinguishing mode*
1	PDIST	*Set distinguishing model*

Execution

Depending on the edit mode, a SET FACE DISTINGUISHING MODE element is inserted into the open structure after the element pointer, or replaces the element pointed at by the element pointer. The element pointer is then updated to point to the new SET FACE DISTINGUISHING MODE (3PP) element.

When the face distinguishing mode is *No*, the back-face attributes are not used, and the front-face attributes are used for both front-facing and back-facing portions of the primitives. When the face distinguishing mode is *Yes*, the back-face attributes are used for back-facing portions.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

SET FACE CULLING MODE (3PP)

,

NAME SET INTERIOR COLOUR – create structure element to set interior colour attribute

SYNOPSIS

C Syntax

void
pset_int_colr (colour)
Pgcolr *colour; *interior colour*

FORTRAN Syntax

SUBROUTINE psico (COLMOD, ICOLR, COLR)
INTEGER COLMOD *colour model*
INTEGER ICOLR *colour table index*
REAL COLR(3) *colour triple*

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET INTERIOR COLOUR creates a structure element to set the interior colour attribute.

When the current interior colour Aspect Source Flag (ASF) is set to INDIVIDUAL, the current value of this attribute defines the interior colour to be applied to the following output primitives:

FILL AREA
FILL AREA 3
FILL AREA SET
FILL AREA SET 3
FILL AREA SET 3 WITH DATA
NON-UNIFORM B-SPLINE SURFACE
POLYHEDRON 3 WITH DATA
QUADRILATERAL MESH 3 WITH DATA
TRIANGLE STRIP 3 WITH DATA

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

The following data types are predefined in phigs.h.

colour Interior colour. Pgcolr is defined as:

```
typedef struct {
   Pint          type;          /* indirect, RGB, CIE, HSV, HLS */
   union {
     Pint        index;         /* colour table index */
     struct {
          Pfloat       x;       /* red, hue, and so on */
          Pfloat       y;       /* green, saturation, lightness,
                                   and so on */
          Pfloat       z;       /* blue, value, saturation, and so on */
     } general;
   } val;
} Pgcolr;
```

FORTRAN Input Parameters

The following data types are predefined in phigs77.h.

COLMOD

The colour model. The predefined values are:

0	PINDIR	*Indirect*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness and Saturation*

ICOLR An integer index into the workstation colour table for determining the interior colour. This value is only used if COLMOD is equal to PINDIR.

COLR(3)

An array of reals containing the colours for the interior. This array is unused if COLMOD is equal to PINDIR.

COLR(1) = *Red, Hue, and so on*
COLR(2) = *Green, Saturation, Lightness, and so on*
COLR(3) = *Blue, Value, Saturation, and so on*

Execution

Depending on the edit mode, a SET INTERIOR COLOUR element is inserted into the open structure after the element pointer or replaces the element pointed at by the element pointer. The element pointer is then updated to point to this SET INTERIOR COLOUR element.

When the current interior colour ASF is set to INDIVIDUAL, filled-area output primitives, which follow in the structure network are filled using the current interior colour.

When the current interior colour ASF is set to BUNDLED, the interior colour is taken from the workstation extended representation indicated by the current interior index. In this case, the interior colour set with SET INTERIOR COLOUR has no effect.

The standard PHIGS function SET INTERIOR COLOUR INDEX is equivalent of calling to SET INTERIOR COLOUR with type INDIRECT.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

COLOUR (7P)
SET INTERIOR REPRESENTATION PLUS (3PP)
SET INTERIOR COLOUR INDEX (3P)

NAME

SET INTERIOR REPRESENTATION PLUS – set extended interior attribute bundle on workstation

SYNOPSIS

C Syntax

```
void
pset_int_rep_plus ( wsid, index, rep )
Pint              wsid;    workstation id
Pint              index;   interior index
Pint_bundle_plus  *rep;    representation
```

FORTRAN Syntax

SUBROUTINE pseir (WKID, INDEX, STYLE, ISTYLE, COLTYPE, ICOLR, COLR, REFEQ, SHADEM, AMCOEF, DICOEF, SPCOEF, SCOLM, ISCOLR, SCOLR, SPEXP, TRCOEF, APTYP, APVAL)

INTEGER	WKID	*workstation identifier*
INTEGER	INDEX	*bundle index*
INTEGER	STYLE(2)	*interior styles*
INTEGER	ISTYLE(2)	*interior style indexes*
INTEGER	COLTYPE(2)	*interior colour types*
INTEGER	ICOLR(2)	*interior colour index*
REAL	COLR(3,2)	*interior colour triples*
INTEGER	REFEQ(2)	*reflectance equations*
INTEGER	SHADEM(2)	*shading methods*
REAL	AMCOEF(2)	*ambient reflectance coefficients*
REAL	DICOEF(2)	*diffuse reflectance coefficients*
REAL	SPCOEF(2)	*specular reflectance coefficients*
INTEGER	SCOLM(2)	*specular colour models*
INTEGER	ISCOLR(2)	*specular colour indexes*
REAL	SCOLR(3,2)	*specular colour triples*
REAL	SPEXP(2)	*specular exponents*
REAL	TRCOEF(2)	*transparency coefficients*
INTEGER	APTYP	*surface approximation type*
REAL	APVAL(2)	*surface approximation values, u and v*

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

SET INTERIOR REPRESENTATION PLUS sets a bundle table entry in the workstation extended interior bundle table.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

wsid The identifier of the workstation for which the representation is being set.

index The index of the entry being set.

rep A pointer to a Pint_bundle_plus structure containing the extended interior attributes. Pint_bundle_plus is defined as:

```
typedef struct {
    Pint_style      style;          /* interior style */
    Pint            style_ind;      /* interior style index */
    Pgcolr          colr;           /* interior colour */
    Pint            refl_eqn;       /* reflectance equation */
    Pint            shad_meth;      /* shading method */
    Prefl_props     refl_props;     /* area properties */
    Pint_style      back_style;     /* interior style */
    Pint            back_style_ind; /* interior style index */
    Pgcolr          back_colr;      /* interior colour */
    Pint            back_refl_eqn;  /* back reflectance equation */
    Pint            back_shad_meth; /* back shading method */
    Prefl_props     back_refl_props;/* back area properties */
    Pint            approx_type;    /* approximation method */
    Pfloat          approx_val[2];  /* approximation values,
                                       u and v */
} Pint_bundle_plus;
```

Pint_style is defined as:

```
typedef enum {
        PSTYLE_HOLLOW,
        PSTYLE_SOLID,
        PSTYLE_PAT,
        PSTYLE_HATCH,
        PSTYLE_EMPTY
} Pint_style;
```

See SET INTERIOR STYLE for a description of each style.

Pgcolr is defined as:

```
typedef struct {
    Pint            type;       /* indirect, RGB, CIE, HSV, HLS */
    union {
        Pint        ind;        /* index in workstation colour
                                   bundle table */
        struct {
            Pfloat  x;          /* red, hue, and so on */
            Pfloat  y;          /* green, saturation, lightness,
                                   and so on */
```

```
        Pfloat          z;          /* blue, value, saturation, and so on */
      } general;
   } val;
} Pgcolr;
```

Prefl_props is defined as:

```
typedef struct {
    Pfloat          ambient_coef;     /* ambient reflectance coefficient */
    Pfloat          diffuse_coef;     /* diffuse reflectance coefficient */
    Pfloat          specular_coef;    /* specular reflectance coefficient */
    Pgcolr          specular_colr;    /* specular colour */
    Pfloat          specular_exp;     /* specular exponent */
    Pfloat          transpar_coef;    /* transparency coefficient */
} Prefl_props;
```

The values for ambient, diffuse, specular, and transparency coefficients must all be in the range [0,1]. The specular exponent must be greater than zero. Note that SunPHIGS has accepted transparency coefficients since release 1.1. However, only GT workstations use these values. A value of 0.0 means no transparency (entirely opaque), and a value of 1.0 means entirely transparent (invisible).

Constants defined for colour type are:

0	PINDIRECT	*Indirect*
1	PMODEL_RGB	*Red, Green, Blue*
2	PMODEL_CIELUV	*CIE*
3	PMODEL_HSV	*Hue, Saturation, Value*
4	PMODEL_HLS	*Hue, Lightness, Saturation*

The index member of the val union is used for type PINDIRECT. The general member is used for the other types.

Predefined reflectance equation values are:

1	PREFL_NONE	*No Reflectance Calculation Performed*
2	PREFL_AMBIENT	*Use Ambient Term*
3	PREFL_AMB_DIFF	*Use Ambient and Diffuse Terms*
4	PREFL_AMB_DIFF_SPEC	*Use Ambient, Diffuse, and Specular Terms*

Predefined constants for interior shading method are:

1	PSD_NONE	*No Shading*
2	PSD_COLOUR	*Colour Interpolation Shading*
3	PSD_DOT_PRODUCT	*Dot Product Interpolation Shading*
4	PSD_NORMAL	*Normal Interpolation Shading*

Constants defined for surface approximation type are:

1	PSURF_WS_DEP
2	PSURF_CONSTANT_PARAMETRIC_BETWEEN_KNOTS
3	PSURF_CHORDAL_SIZE_WC
4	PSURF_CHORDAL_SIZE_NPC

5 PSURF_CHORDAL_SIZE_DC
6 PSURF_PLANAR_DEVIATION_WC
7 PSURF_PLANAR_DEVIATION_NPC
8 PSURF_PLANAR_DEVIATION_DC

See SET SURFACE APPROXIMATION CRITERIA for a description of these constants.

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

WKID The identifier of the workstation for which the representation is being set.

INDEX The interior index of the entry being set.

STYLE(2)

The front and back interior styles. Valid values are defined as:

0 PHOLLO *Hollow*
1 PSOLID *Solid*
2 PPATTR *Patterned*
3 PHATCH *Hatched*
4 PISEMP *Empty*

See SET INTERIOR STYLE for a description of each style.

ISTYLE(2)

The front and back interior style indexes, which select entries in the workstation *Pattern*, *Hatch*, or *General* interior style tables.

COLTYPE(2)

The front and back interior colour types. Valid values are defined as:

0 PINDIR *Indirect*
1 PRGB *Red, Green, and Blue*
2 PCIE *CIE*
3 PHSV *Hue, Saturation, and Value*
4 PHLS *Hue, Lightness, and Saturation*

ICOLR(2)

The front and back colour indexes. *ICOLR(2)* is used only when the corresponding value in COLTYPE(2) is PINDIR.

COLR(3,2)

The front and back colour triples. *COLR(3,2)* is used only when the corresponding value in COLTYPE(2) is not PINDIR. The triples are stored as:

ICOLR(1,c) = *x Colour*
ICOLR(2,c) = *y Colour*
ICOLR(3,c) = *z Colour*

REFEQ(2)

The front and back reflectance equations. Predefined reflectance equation values are:

1 PRNONE *No Reflectance Calculation Performed*
2 PRAMB *Use Ambient Term*

3	PRAMBD	*Use Ambient and Diffuse Terms*
4	PRAMDS	*Use Ambient, Diffuse, and Specular Terms*

SHADEM(2)
The front and back interior shading methods. Predefined constants are:

1	PSDNON	*No Shading*
2	PSDCOL	*Colour Interpolation Shading*
3	PSDDOT	*Dot Product Interpolation Shading*
4	PSDNOR	*Normal Interpolation Shading*

AMCOEF(2)
The front and back ambient reflectance coefficients, in the range [0,1].

DICOEF(2)
The front and back diffuse reflectance coefficients, in the range [0,1].

SPCOEF(2)
The front and back specular reflectance coefficients, in the range [0,1].

SCOLM(2)
The front and back specular colour models.

ISCOLR(2)
The front and back specular colour indexes. ISCOLR(2) is used only when the corresponding value in SCOLM is PINDIR.

SCOLR(3,2)
The front and back specular colour triples. *SCOLR(3,2)* is used only if the corresponding value in SCOLM is not PINDIR.

SPEXP(2)
The front and back specular exponents, greater than zero.

TRCOEF(2)
The front and back transparency coefficients, in the range [0,1]. Note that SunPHIGS has accepted transparency coefficients since release 1.1. However, only GT workstations use these values. A value of 0.0 means no transparency (entirely opaque) and a value of 1.0 means entirely transparent (invisible).

APTYP The surface approximation type. Predefined values are:

1	PSAWSD	*Workstation-dependent*
2	PSAEPK	*Constant Parametric Subdivision Between Knots*
3	PSACSW	*Chordal Size in WC*
4	PSACSN	*Chordal Size in NPC*
5	PSACSD	*Chordal Size in DC*
6	PSAPDW	*Planar Deviation in WC*
7	PSAPDN	*Planar Deviation in NPC*
8	PSAPDD	*Planar Deviation in DC*

APVAL(2)
The surface approximation values.

Execution

The extended interior bundle table contains bundled entries of the PHIGS and SunPHIGS Extension interior attributes. SET INTERIOR REPRESENTATION PLUS sets the individual entries of this table. During traversal, the attribute values of these bundles are used when the corresponding aspect source flag in the workstation state list is set to BUNDLED.

The PHIGS interior bundle table is a portion of the extended table. The function SET INTERIOR REPRESENTATION sets the following attributes in the extended table: interior style, interior style index, and colour index (colour type of *Indirect* is assumed).

During structure traversal the current interior index refers to an entry in the workstation extended interior bundle table. Depending on the settings of the aspect source flags, the contents of that bundle table entry are used to determine the attributes associated with output primitives that use the extended interior bundle table.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, the specified workstation is not open

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

100 Ignoring function, the bundle index value is less than one

103 Ignoring function, setting this bundle table entry would exceed the maximum number of entries allowed in the workstation bundle table

108 Ignoring function, the specified interior style is not available on the workstation

110 Ignoring function, the specified colour model is not available on the workstation

112 Ignoring function, the pattern index value is less than one

SEE ALSO

SET INDIVIDUAL ASF (3P)
SET SURFACE APPROXIMATION CRITERIA (3PP)
INQUIRE INTERIOR FACILITIES PLUS (3PP)
INQUIRE INTERIOR REPRESENTATION PLUS (3PP)
SET INTERIOR STYLE INDEX (3P)
SET INTERIOR REPRESENTATION (3P)

NAME

SET INTERIOR SHADING METHOD – create structure element that sets current interior shading method

SYNOPSIS

C Syntax

void
pset_int_shad_meth (method)
Pint method; *shading method*

FORTRAN Syntax

SUBROUTINE psism (METHOD)
INTEGER METHOD *shading method*

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET INTERIOR SHADING METHOD places a structure element with the value shading method into the open structure. When the current interior shading method Aspect Source Flag (ASF) is set to INDIVIDUAL, the current value of this attribute applies to the following output primitives:

FILL AREA
FILL AREA SET
FILL AREA SET 3 WITH DATA
QUADRILATERAL MESH 3 WITH DATA
TRIANGLE STRIP 3 WITH DATA
POLYHEDRON 3 WITH DATA
NON-UNIFORM B-SPLINE SURFACE

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameter

The following data type is predefined in phigs.h.

method The interior shading method.

1	PSD_NONE	*No shading*
2	PSD_COLOUR	*Colour interpolation shading*
3	PSD_DOT_PRODUCT	*Dot product interpolation shading*
4	PSD_NORMAL	*Normal interpolation shading*

FORTRAN Input Parameters

The following data types are predefined in phigs77.h.

METHOD

The value of the shading method

1	PSDNON	*No shading*
2	PSDCOL	*Colour interpolation shading*
3	PSDDOT	*Dot product interpolation shading*
4	PSDNOR	*Normal interpolation shading*

Execution

Depending on the edit mode, a SET INTERIOR SHADING METHOD element is inserted into the open structure after the element pointer, or replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the newly created SET INTERIOR SHADING METHOD element.

When the structure is traversed, the current interior shading method entry in the traversal state list is set to the specified value. This value is used with subsequent filled area output primitives when the current interior shading method ASF entry in the traversal state list is INDIVIDUAL. During traversal, if the specified shading method is not available on a workstation, method one (*None*) will be used on that workstation.

When the current interior shading method ASF is set to BUNDLED, the interior shading method is taken from the workstation extended representation indicated by the current interior index. In this case, the interior shading method set with SET INTERIOR SHADING METHOD has no effect.

The interior shading methods supported by a workstation type can be determined with the function INQUIRE INTERIOR FACILITIES PLUS.

SunPHIGS does not support the *Dot Product Interpolation Shading* and *Normal Interpolation Shading* methods. When one of these shading methods is specified, the backup NONE is used.

When the shading method is NONE, each fill area or fill area set will be rendered in a single colour (this is sometimes called flat shading). If the shading method is NONE and the primitive data implies different colours for each vertex of the primitive, such as when vertex colour is specified or when lighting is active, the colour of the first vertex will be used for all the vertices of the primitive.

When the shading method is COLOUR and the workstation is in index colour mode, then the colours will be interpolated as colour indices instead of direct colours.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

INTRO (3PP)
FILL AREA (3P)
FILL AREA SET (3P)
FILL AREA SET 3 WITH DATA (3PP)
QUADRILATERAL MESH 3 WITH DATA (3PP)
TRIANGLE STRIP 3 WITH DATA (3PP)
SET OF FILL AREA SET 3 WITH DATA (3PP)
SET BACK INTERIOR SHADING METHOD (3PP)
SET INDIVIDUAL ASF (3P)
SET INTERIOR REPRESENTATION PLUS (3PP)
INQUIRE INTERIOR FACILITIES PLUS (3PP)
NON-UNIFORM B-SPLINE SURFACE (3PP)

NAME

SET LIGHT SOURCE REPRESENTATION – define light source representation entry in workstation table of defined light source representations

SYNOPSIS

C Syntax

void
pset_light_src_rep (ws, index, rep)

Pint	ws;	*workstation identifier*
Pint	index;	*light table index*
Plight_src_bundle	*rep;	*light source representation*

FORTRAN Syntax

SUBROUTINE pslsr (WKID, INDEX, LSTYPE, LDR, DATREC)

INTEGER	WKID	*workstation identifier*
INTEGER	INDEX	*light source table index*
INTEGER	LSTYPE	*light source type*
INTEGER	LDR	*dimension of data record array*
CHARACTER*80	DATREC(LDR)	*data record*

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

SET LIGHT SOURCE REPRESENTATION defines a light source representation for a specified index number on the workstation table of defined light source representations. Each integer entry on this table specifies a light source type (*Ambient*, *Directional*, *Positional*, or *Spot*) and a data record containing the parameters needed to define the light source type.

Defined light sources are activated or deactivated with the SET LIGHT SOURCE STATE subroutine. When lighting and shading are applied to a display, the currently active light sources are used, with the current surface properties and interior lighting and shading methods, to render the following area defining primitives: FILL AREA 3, FILL AREA SET 3, and FILL AREA SET 3 WITH DATA.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

ws The workstation identifier specifies the workstation for which the light source representation is to be defined. The workstation must have been opened with the OPEN WORKSTATION subroutine before calling SET LIGHT SOURCE REPRESENTATION.

index The number of the entry to be defined on the table of defined light source representations, in the specified workstation state list.

rep A pointer to light source representation. The Plight_src_bundle data structure contains values needed to define the specified type of light source representation. Plight_src_bundle is defined as:

```
typedef struct {
```

```
        Pint            type;       /* light source type */
        Plight_src_rec  rec;        /* light source data record */
} Plight_src_bundle;
```

The predefined light source types are:

1	PLIGHT_AMBIENT	*Ambient light source*
2	PLIGHT_DIRECTIONAL	*Directional light source*
3	PLIGHT_POSITIONAL	*Positional light source*
4	PLIGHT_SPOT	*Spot light source*

Plight_src_rec is defined as:

```
typedef union {
        Pamb_light_src_rec      ambient;
        Pdir_light_src_rec      directional;
        Ppos_light_src_rec      positional;
        Pspot_light_src_rec     spot;
} Plight_src_rec;
```

Pamb_light_src_rec is defined as:

```
typedef struct {
        Pgcolr          colr;           /* light source colour */
} Pamb_light_src_rec;
```

Pgcolr is defined as:

```
typedef struct {
        Pint            type;       /* indirect, RGB, CIE, HSV, HLS */
        union {
           Pint         ind;        /* index in workstation colour
                                       bundle table */
           struct {
              Pfloat    x;          /* red, hue, and so on */
              Pfloat    y;          /* green, saturation, lightness,
                                       and so on */
              Pfloat    z;          /* blue, value, saturation, and
                                       so on */
           } general;
        } val;
} Pgcolr;
```

Pdir_light_src_rec is defined as:

```
typedef struct {
        Pgcolr          colr;           /* light source colour */
        Pvec3           dir;            /* light source direction */
} Pdir_light_src_rec;
```

Pvec3 is defined as:

```
typedef struct {
        Pfloat          delta_x;        /* x magnitude */
        Pfloat          delta_y;        /* y magnitude */
        Pfloat          delta_z;        /* z magnitude */
} Pvec3;
```

Ppos_light_src_rec is defined as:

```
typedef struct {
        Pgcolr          colr;           /* light source colour */
        Ppoint3         pos;            /* light source position */
        Pfloat          coef[2];        /* attenuation coefficients */
} Ppos_light_src_rec;
```

Ppoint3 is defined as:

```
typedef struct {
        Pfloat          x;      /* x coordinate */
        Pfloat          y;      /* y coordinate */
        Pfloat          z;      /* z coordinate */
} Ppoint3;
```

Pspot_light_src_rec is defined as:

```
typedef struct {
        Pgcolr          colr;           /* light source colour */
        Ppoint3         pos;            /* light source position */
        Pvec3           dir;            /* light source direction */
        Pfloat          exp;            /* concentration exponent */
        Pfloat          coef[2];        /* attenuation coefficients */
        Pfloat          angle;          /* half-cone spread angle */
} Pspot_light_src_rec;
```

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

WKID The identifier of the workstation for which the light source representation is being set.

INDEX An index into to the light source table of the workstation state list for the entry being set.

LSTYPE

The light source type. The predefined values are:

1	PAMB	*Ambient light source*
2	PDIR	*Directional light source*
3	PPOS	*Positional light source*
4	PSPOT	*Spot light source*

LDR The dimension of the data record array, DATREC.

DATREC(LDR)

The data record is packed by PACK DATA RECORD. The contents depend on the light source type specified:

Ambient Light Source, Type 1

IL = the number of integers, 1 or 2

IA = an array of integers with the following values:

IA(1) = the colour model type. Valid values are:

0	PINDIR	*Indirect*
1	PRGB	*Red, green, and blue*
2	PCIE	*CIE colour model*
3	PHSV	*Hue, saturation, and value*
4	PHLS	*Hue, lightness, and saturation*

IA(2) = the light source colour table index, if IA(1) specifies PINDIR

RL = the number of reals, 3

RA = an array of reals with the following values:

RA(1) - RA(3) =
the light source colour triple (x, y, z), if IA(1) is other than PINDIR

Note: These values are not used if IA(1) is PINDIR.

SL = the number of strings, 0

Directional Light Source, Type 2

IL = the number of integers, 1 or 3

IA = an array of integers with the following values:

IA(1) = the colour model type

IA(2) = the colour table index, if IA(1) specifies PINDIR

RL = the number of reals, 6

RA = an array of reals with the following values:

RA(1) - RA(3) =
the light source colour triple (x, y, z), if IA(1) is other than PINDIR

RA(4) - RA(6) =
the light source direction (x, y, z)

SL = the number of strings, 0

Positional Light Source, Type 3

IL = the number of integers, 1 or 2

IA = an array of integers with the following values:

IA(1) = the colour model type

IA(2) = the colour table index, if IA(1) specifies PINDIR

RL = the number of reals, 8

RA = an array of reals with the following values:

RA(1) - RA(3) =
the light source colour triple (x, y, z), if IA(1) is other than PINDIR

RA(4) - RA(6) =
the light source position (x, y, z)

RA(7) - RA(8) =
the attenuation coefficients

SL = the number of strings, 0

Spot Light Source, Type 4

IL = the number of integers, 1 or 2

IA = an array of integers with the following values:

IA(1) = the colour model type

IA(2) = the colour table index, if IA(1) specifies PINDIR

RL = the number of reals, 13

RA = an array of reals with the following values:

RA(1) - RA(3) =
the light source colour triple (x, y, z), if IA(1) is other than PINDIR

RA(4) - RA(6) =
the light source position (x, y, z)

RA(7) - RA(9) =
the light source direction (x, y, z)

RA(10) = the concentration exponent

RA(11) - RA(12) =
the attenuation coefficients

RA(13) = the half-cone spread angle

SL = the number of strings, 0

Execution

When SET LIGHT SOURCE REPRESENTATION is called, the light source index entry in the table of defined light source representations on the workstation is set to the new attribute values.

Note: SunPHIGS defines the direction of a directional light to be the vector from the light to the object. This is the opposite of what some applications expect.

ERRORS

003	Ignoring function, function requires state (PHOP, WSOP, *, *)
054	Ignoring function, the specified workstation is not open
059	Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
103	Ignoring function, setting this bundle table entry would exceed the maximum number of entries allowed in the workstation bundle table
169	Ignoring function, the specified light direction is invalid
614	Ignoring function, the light source index is less than one
617	Ignoring function, the specified light source type is not available on the workstation
618	Ignoring function, the specified spot light spread angle is out of range

SEE ALSO

SET REFLECTANCE PROPERTIES (3PP)
SET REFLECTANCE EQUATION (3PP)
INQUIRE LIST OF LIGHT SOURCE INDICES (3PP)
INQUIRE LIGHT SOURCE REPRESENTATION (3PP)
INQUIRE LIGHT SOURCE FACILITIES (3PP)
INQUIRE PREDEFINED LIGHT SOURCE REPRESENTATION (3PP)

NAME

SET LIGHT SOURCE STATE – create structure element to enable and disable light sources

SYNOPSIS

C Syntax

```
void
pset_light_src_state ( activation, deactivation )
Pint_list   *activation;     activation list
Pint_list   *deactivation;   deactivation list
```

FORTRAN Syntax

```
SUBROUTINE pslss ( NACT, NDEACT, ACT, DEACT )
INTEGER   NACT              length of activation list
INTEGER   NDEACT            length of deactivation list
INTEGER   ACT(NACT)         light source activation list
INTEGER   DEACT(NDEACT)     light source deactivation list
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET LIGHT SOURCE STATE creates an element in the currently open structure enabling and disabling specified light source representations on the table of defined light sources stored on the workstation state list. The current light source state determines the number and type of light sources used to shade the primitives which follow in the structure network.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

activation

This is a list of light sources to be activated. This structure specifies both an integer value indicating the number of light source representations to be activated and a pointer to a list specifying the index numbers of the light source representations to be activated. Pint_list is defined as:

```
typedef struct {
        Pint    num_ints;       /* number of light sources to activate */
        Pint    *ints;          /* list of light sources to activate */
} Pint_list;
```

deactivation

This is a list of light sources to be deactivated. This structure specifies both an integer value indicating the number of light source representations to be deactivated and a pointer to a list specifying the index numbers of the light source representations to be deactivated. Pint_list is defined as:

```
typedef struct {
        Pint    num_ints;       /* number of light sources to activate */
        Pint    *ints;          /* list of light sources to activate */
} Pint_list;
```

FORTRAN Input Parameters

NACT
The length of the light source list to be activated.

NDEACT
The length of the light source list to be deactivated.

ACT(NACT)
The light source list to be activated.

DEACT(NDEACT)
The light source list to be deactivated.

Execution

Depending on the edit mode, a SET LIGHT SOURCE STATE element is inserted into the open structure after the element pointer or replaces the element pointed at by the element pointer. The element pointer is then updated to point to this SET LIGHT SOURCE STATE element.

When the SET LIGHT SOURCE STATE element is traversed, the light source representations listed in activate are added to the current light source state entry in the *PHIGS Extension* traversal state list, and the representations listed in deactivate are deleted from the list. The light source representations are defined with the SET LIGHT SOURCE REPRESENTATION subroutine.

The current light source state list defines the ambient, diffuse, and specular light sources that may be used when applying lighting and shading to area defining or polyline primitives. The types of light sources used are defined by the SET INTERIOR SHADING METHOD and SET POLYLINE SHADING METHOD subroutines.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

SET LIGHT SOURCE REPRESENTATION (3PP)
INQUIRE LIGHT SOURCE FACILITIES (3PP)

NAME

SET OF FILL AREA SET 3 WITH DATA – creates a 3D polyhedron structure element that includes colour and shading data

SYNOPSIS

C Syntax

void
pset_of_fill_area_set3_data (fflag, eflag, vflag, colr_type, num_sets, fdata, edata, vlist, vdata)

Pint	fflag;	*data per facet flag*
Pint	eflag;	*data per edge flag*
Pint	vflag;	*data per vertex flag*
Pint	colr_type;	*colour type*
Pint	num_sets;	*number of facets (fill area sets)*
Pfacet_data_arr3	*fdata;	*array of L({COLRV}{,NORM}) facet data*
Pedge_data_list_list	*edata;	*array of L(L(E)) edge data*
Pint_list_list	*vlist;	*array of L(L(I)) vertex indices*
Pfacet_vdata_list3	*vdata;	*array of L(P3,{COLRV}{,NORM}) facet vertex data*

FORTRAN Syntax

SUBROUTINE psfsd3 (FFLAG, EFLAG, VFLAG, COLTYP, NFA, IFCOLR, FCOLR, FNORM, NB, FARR, VFARR, EDARR, NV, COORDS, IVCOLR, VCOLR, VNORM, IMPL)

INTEGER	FFLAG	*data per facet flag*
INTEGER	EFLAG	*data per edge flag*
INTEGER	VFLAG	*data per vertex flag*
INTEGER	COLTYP	*colour type*
INTEGER	NFA	*number of facets (fill area sets)*
INTEGER	IFCOLR(NFA)	*array of indirect facet colours*
REAL	FCOLR(3,NFA)	*array of facet colours*
REAL	FNORM(3,NFA)	*array of facet normals*
INTEGER	NB(NFA)	*array of bounds (fill areas) per facet*
INTEGER	FARR(*)	*array of end indices per facet*
INTEGER	VFARR(*)	*array of vertex indices per facet*
INTEGER	EDARR(*)	*array of edge data (PON/POFF)*
INTEGER	NV	*number of vertices*
REAL	COORDS(3,NV)	*array of vertex coordinates*
INTEGER	IVCOLR(NV)	*array of indirect vertex colours*
REAL	VCOLR(3,NV)	*array of vertex colours*
REAL	VNORM(3,NV)	*array of vertex normals*
REAL	IMPL(*)	*implementation defined vertex data*

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET OF FILL AREA SET 3 WITH DATA creates a polyhedron primitive with colour and shading data.

A polyhedron generates a group of fill area sets (or facets) whose vertices are selected by indexing into a single list of vertices. If the facets share common vertices, the primitive can require less data than the corresponding collection of FILL AREA SET 3 WITH DATA primitives.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

The following data types are predefined in phigs.h.

fflag The data per facet flag specifies the available data for each facet in the set of fill area set 3.

0	PFACET_NONE	*No Facet Data Specified*
1	PFACET_COLOUR	*Facet Colours Specified*
2	PFACET_NORMAL	*Facet Normals Specified*
3	PFACET_COLOUR_NORMAL	*Facet Normals and Colours Specified*

eflag The data per edge flag specifies the available information for each edge of each facet in the set of fill area set 3.

0	PEDGE_NONE	*No Edge Data Specified*
1	PEDGE_VISIBILITY	*Edge Visibility Flags Specified*

vflag The data per vertex flag specifies the available data for each vertex of the primitive.

0	PVERT_COORD	*Coordinates Specified*
1	PVERT_COORD_COLOUR	*Coordinates and Vertex Colours Specified*
2	PVERT_COORD_NORMAL	*Coordinates and Vertex Normals Specified*
3	PVERT_COORD_COLOUR_NORMAL	*Coordinates, Vertex Colours, and Vertex Normals Specified*

colr_type

The colour type for specified vertex colour and/or norm information for each facet. Pfacet_data_arr3 is defined as:

```
typedef union {
        Pcoval      *colrs;     /* array (num_sets) of colours */
        Pvec3       *norms;     /* array (num_sets) of unit normals */
        Pconorm3    *conorms;   /* array (num_sets) of colours and
                                   normals */
        /* implementation-dependent types can go here */
} Pfacet_data_arr3;
```

Pcoval is defined as:

```
typedef union {
        Pint        ind;      /* index in workstation colour bundle
                                 table */
        Pcolr_rep   direct;   /* direct colour components */
} Pcoval;
```

When colour type is PINDIRECT, index is used; otherwise *direct* is used.

Pcolr_rep is defined in phigs.h as follows:

```
typedef union {
        Prgb        rgb;      /* Red Green Blue colour specification */
        Pcieluv     cieluv;   /* CIE L*U*V* colour specification */
        Phls        hls;      /* Hue Lightness Saturation colour
                                 specification */
        Phsv        hsv;      /* Hue Saturation Value colour
                                 specification */
        Pdata       unsupp;   /* Colour in unsupported colour type*/
} Pcolr_rep
```

Prgb is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          red;        /* red, hue, and so on */
        Pfloat          green;      /* green, saturation, lightness,
                                       and so on */
        Pfloat          blue;       /* blue, value, saturation, and
                                       so on */
} Prgb;
```

Pcieluv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          cieluv_x;       /* x coefficient */
        Pfloat          cieluv_y;       /* y coefficient */
        Pfloat          cieluv_y_lum;   /* y luminance */
} Pcieluv;
```

Phls is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;        /* hue */
        Pfloat          lightness;  /* lightness */
        Pfloat          satur;      /* saturation */
} Phls;
```

Phsv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;        /* hue */
```

```
        Pfloat          satur;          /* saturation */
        Pfloat          value;          /* value */
} Phsv;
```

Pdata is defined in phigs.h as follows:

```
typedef struct {
        size_t          size;           /* size of data */
        char            *data           /* pointer to data */
} Pdata;
```

Pconorm3 is defined as:

```
typedef struct {
        Pcoval  colr;       /* colour */
        Pvec3   norm;       /* unit normal */
} Pconorm3;
```

Pcoval is defined above. Pvec3 is defined as:

```
typedef struct {
        Pfloat  delta_x;    /* x magnitude */
        Pfloat  delta_y;    /* y magnitude */
        Pfloat  delta_z;    /* z magnitude */
} Pvec3;
```

edata A pointer to an array (num_sets) of Pedge_data_list_list structures. Each Pedge_data_list_list structure specifies the edge data list for the corresponding facet (fill area set). Each Pedge_data_list structure contains a list of edge flags for a single bound in a facet. The first element of the Pedge_data_arr edge data array contains the information pertaining to the edge that falls between the first and second vertex index. The number of edges will be the same as the number of vertices in the vlist data structure. Pedge_data_list_list (edge data for a single fill area set) is defined as:

```
typedef struct {
        Pint                num_lists;  /* number of Pedge_data_lists
                                           (bounds) in list */
        Pedge_data_list     *edgelist;  /* list of edge data list */
} Pedge_data_list_list;
```

Pedge_data_list (list of edges for one bound) is defined as:

```
typedef struct {
        Pint                num_edges;  /* number of edges */
        Pedge_data_arr      edgedata;   /* edge data */
} Pedge_data_list;
```

Pedge_data_arr is defined as:

```
typedef union {
        Pedge_flag    *edges;   /* array of edge flags */
        /* implementation-dependent data */
} Pedge_data_arr;
```

Pedge_flag is defined as:

```
typedef enum {
        PEDGE_OFF,
        PEDGE_ON
} Pedge_flag;
```

vlist A pointer to an array (num_sets) of Pint_list_list structures. Each Pint_list_list structure specifies the integer list for the corresponding facet (fill area set). Each Pint_list structure contains the indices into the vertex data that specify a single bound of each facet. Pint_list_list (bounds for a single fill area set) is defined as:

```
typedef struct {
        Pint        num_lists;    /* number of Pint_lists in list */
        Pint_list   *lists;       /* list of integer lists */
} Pint_list_list;
```

Pint_list (vertex indices for all vertices in a single bound) is defined as:

```
typedef struct {
        Pint    num_ints;    /* number of Pints in list */
        Pint    *ints;       /* list of integers */
} Pint_list;
```

vdata A pointer to a Pfacet_vdata_list3 structure that specifies the set of fill area set 3 vertices and optionally associated colour and norm information. Pfacet_vdata_list3 is defined as:

```
typedef struct {
        Pint                num_vertices;    /* number of vertices */
        Pfacet_vdata_arr3   vertex_data;     /* facet vertex data */
} Pfacet_vdata_list3;
```

Pfacet_vdata_arr3 is defined as:

```
typedef union {
        Ppoint3       *points;       /* array of points */
        Pptco3        *ptcolrs;      /* array of points and colours */
        Pptnorm3      *ptnorms;      /* array of points and normals */
        Pptconorm3    *ptconorms;    /* array of points, colours,
                                        and normals */
        /* implementation-dependent data pointer */
```

```
} Pfacet_vdata_arr3;
```

Ppoint3 is defined as:

```
typedef struct {
        Pfloat    x;    /* x coordinate */
        Pfloat    y;    /* y coordinate */
        Pfloat    z;    /* z coordinate */
} Ppoint3;
```

Pptco3 is defined as:

```
typedef struct {
        Ppoint3   point;    /* point coordinates */
        Pcoval    colr;     /* colour */
} Pptco3;
```

Ppoint3 and Pcoval are defined above.

Pptnorm3 is defined as:

```
typedef struct {
        Ppoint3   point;    /* point coordinates */
        Pvec3     norm;     /* unit normal */
} Pptnorm3;
```

Ppoint3 and Pvec3 are defined above.

Pptconorm3 is defined as:

```
typedef struct {
        Ppoint3   point;    /* point coordinates */
        Pcoval    colr;     /* colour */
        Pvec3     norm;     /* unit normal */
} Pptconorm3;
```

Ppoint3, Pcoval, and Pvec3 are defined above.

FORTRAN Input Parameters

The following data types are predefined in phigs77.h.

FFLAG The data per facet flag indicates the information specified for each facet (fill area set) in the polyhedron. The possible values are:

0	PFNONE	*No Facet Data Specified*
1	PFCLR	*Facet Colour Specified*
2	PFNORM	*Facet Normal Specified*
3	PFCLRN	*Facet Colour and Normal Specified*

EFLAG The data per edge flag indicates the information specified for each edge of each fill area set in the polyhedron. The possible values are:

0 PEDNO *No Edge Data Specified*
1 PEDVIS *Edge Visibility Flags Specified*

VFLAG The data per vertex flag indicates the information specified with each vertex of the primitive. The possible values are:

0 PVCOOR *Coordinates Only*
1 PVCLR *Coordinates and Colours*
2 PVNORM *Coordinates and Normals*
3 PVCLRN *Coordinates, Colours, and Normals*

COLTYP

The colour type. If the data per facet flag (FFLAG) indicates that a facet colour is specified, or if the data per vertex flag (VFLAG) indicates that a vertex colours are specified, then COLTYP is set to one of the following:

0 PINDIR *Indirect*
1 PRGB *Red, Green, and Blue*
2 PCIE *CIE Colour Model*
3 PHSV *Hue, Saturation, and Value*
4 PHLS *Hue, Lightness, and Saturation*

NFA The number of facets or fill area sets in the polyhedron.

IFCOLR(NFA)

An array of integers containing indices into the workstation colour table for determining the colour of each facet. This array is unused if FFLAG is set to either PFNONE or PFNORM or if COLTYP is greater than zero.

FCOLR(3,NFA)

An array of reals containing the colours for the facet. This array is unused if FFLAG is set to either PFNONE or PFNORM or COLTYP is equal to zero.

FCOLR(1,f) = *Red, Hue, and so on*
FCOLR(2,f) = *Green, Saturation, Lightness, and so on*
FCOLR(3,f) = *Blue, Value, Saturation, and so on*

FNORM(3,NFA)

An array of reals containing the unit normals for the facet. This array is unused if *FFLAG* is set to either PFNONE or PFCLR.

FNORM(1,f) = *x Magnitude*
FNORM(2,f) = *y Magnitude*
FNORM(3,f) = *z Magnitude*

NB(NFA)

An array of integers containing the bounds (fill areas) for each facet in the polyhedron.

FARR(*)

An array of integers containing the end indices into the VFARR array for each fill area set in the polyhedron. The dimension of this array is equal to the sum of the bounds (fill areas) in the entire polyhedron (total number of elements in the NB

array).

VFARR(*)
An array of integers containing the vertex indices into the COORDS array which bound each fill area set in the polyhedron. The number of elements in the VFARR array may differ from the total number of vertices in the COORDS array.

EDARR(*)
An array of integers containing the edge visibility data. The first element of the edge data array contains the information pertaining to the edge which falls between vertex index one and vertex index two. The number of edges will be the same as the number of vertices in the VFARR array. For example, a polyhedron containing two facets, one a triangle with edges on, and one a square with edges off would appear as:

```
      data edarr    / 1, 1, 1,
     +                0, 0, 0, 0 /
```

The possible values for each element of the array are:

0	POFF	*Off*
1	PON	*On*

NV The total number of points used to define the polyhedron.

COORDS(3,NV)
An array of reals containing the *x*, *y*, and *z* coordinates of the polyhedron.

COORDS(1,v) = *x Coordinate*
COORDS(2,v) = *y Coordinate*
COORDS(3,v) = *z Coordinate*

IVCOLR(NV)
An array of integers containing indices into the workstation colour table for each vertex. This array is unused if VFLAG is set to either PVCOOR or PVNORM or if COLTYP is greater than zero.

VCOLR(3,NV)
An array of reals containing the colours for each vertex. This array is unused if VFLAG is set to either PVCOOR or PVNORM or COLTYP is equal to zero.

VCOLR(1,v) = *Red, Hue, and so on*
VCOLR(2,v) = *Green, Saturation, Lightness, and so on*
VCOLR(3,v) = *Blue, Value, Saturation, and so on*

VNORM(3,NV)
An array of reals containing the unit normals for each vertex. This array is unused if VFLAG is set to either PVCOOR or PVCLR.

VNORM(1,v) = *x Magnitude*
VNORM(2,v) = *y Magnitude*
VNORM(3,v) = *z Magnitude*

IMPL(*)
This implementation-defined data is not currently defined.

Execution

Depending on the edit mode, a SET OF FILL AREA SET 3 WITH DATA element is inserted into the open structure after the element pointer, or replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the newly created SET OF FILL AREA SET 3 WITH DATA element.

When the structure is traversed, SET OF FILL AREA SET 3 WITH DATA generates a group of implicitly closed facets. Each facet is rendered as a fill area set. The collection of facets need not be coplanar and need not necessarily form a closed surface. Vertices that define the facet contours are selected through an array of indices. Each array references specific coordinates from the vertex list. The odd winding rule defines the facet interior. The application should insure that each facet's points are coplanar. Non-coplanar facet display is device dependent. Facets containing fewer than three vertices can be created, but are ignored during structure traversal.

SET OF FILL AREA SET 3 WITH DATA uses the same attributes as the FILL AREA SET 3 WITH DATA structure element.

Vertex and facet colours and normals are used to calculate the polyhedron colour and shading. The SET OF FILL AREA SET 3 WITH DATA element can specify the colour and normal vector of each facet, as well as colour and normal vectors for each vertex. Note that unexpected lighting effects will result if the supplied normals are not of unit length.

Each polyhedron edge has an optional visibility flag to determine if that edge is rendered. There must be as many lists of edge visibility flags as there are lists of vertex indices, and corresponding lists must be the same length. The first edge visibility array entry controls edge visibility between the first two corresponding contour vertices. The last edge visibility array entry controls edge visibility between the last and the first vertices of the corresponding contour. If edge visibility flags are specified, an edge is displayed only when the EDGE FLAG attribute and the edge visibility flag are both ON. If edge visibility flags are not specified, no edges are displayed.

Attributes Applied

The attributes listed below are used to display the SET OF FILL AREA SET 3 WITH DATA primitive when the structure is traversed. The Aspect Source Flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

interior colour	interior colour index ASF
back interior colour	back interior colour ASF
interior style	interior style ASF
back interior style	back interior style ASF
interior style index	interior style index ASF
back interior style index	back interior style index ASF
interior shading method	interior shading method ASF
back interior shading method	back interior shading method ASF
interior reflectance characteristics	interior reflectance characteristics ASF
back interior reflectance characteristics	back interior reflectance characteristics ASF
reflectance properties	reflectance properties ASF

back reflectance properties	back reflectance properties ASF
interior index	
edge colour	edge colour index ASF
edge flag	edge flag ASF
edgetype	edgetype ASF
edgewidth scale factor	edgewidth scale factor ASF
edge index	
face distinguishing mode	
face culling mode	
depth cue index	
light source state	
name set	

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

619 Ignoring function, one or more of the vertex indices is out of range

900 Storage overflow has occurred in PHIGS

2004 FORTRAN-specific binding error, ignoring function, input parameter size out of range

SEE ALSO

FILL AREA SET 3 WITH DATA (3PP)
INTRO (3PP)

NAME

SET PARAMETRIC SURFACE CHARACTERISTICS – create structure element to set the current parametric surface characteristics

SYNOPSIS

C Syntax

```
void
pset_para_surf_characs( psc_type, data )
Pint                psc_type;   type
Ppara_surf_characs  *data;      OUT data record
```

Required PHIGS Operating States

(PHOP, *, *, *)

FORTRAN Syntax

```
SUBROUTINE pspsc ( PSCTYP,LDR,DATREC )
INTEGER             PSCTYP          parametric surface characteristic type
INTEGER             LDR             dimension of data record array
CHARACTER*80        DATREC(LDR)     data record
```

Required PHIGS Operating States

(PHOP, *, *, *)

DESCRIPTION

Purpose

SET PARAMETRIC SURFACE CHARACTERISTICS sets current parametric surface characteristics to be used when displaying subsequent parametric surface output primitives.

C Input Parameter

psc_type

The parametric surface characteristic types.

1	PSC_NONE	*No surface characteristics specified*
2	PSC_WS_DEP	*Workstation dependent*
3	PSC_ISOPARAMETRIC_CURVES	*Isoparametric curves specified*
4	PSC_LEVEL_CURVES_MC	*Level curves specified in Modelling Coordinates*
5	PSC_LEVEL_CURVES_WC	*Level curves specified in World Coordinates*

C Output Parameters

data The surface characteristics data record. A pointer to a structure Ppara_surf_characs. Ppara_surf_characs is defined in phigs.h as follows:

```
typedef union {
struct {
        Pint            unused;
} psc_1;
struct {
        Pint            unused;
} psc_2;
```

```
struct {
        Pcurve_placement    placement;
        Pint                u_count;
        Pint                v_count;
} psc_3;
struct {
        Ppoint3             origin;
        Pvec3               direction;
        Pfloat_list         params;
} psc_4;
struct {
        Ppoint3             origin;
        Pvec3               direction;
        Pfloat_list         params;
} psc_5;
} Ppara_surf_characs;
```

Pcurve_placement is defined in phigs.h as follows:

```
typedef enum {
        PCP_UNIFORM,
        PCP_NON_UNIFORM
} Pcurve_placement;
```

FORTRAN Input Parameters

PSCTYP

The parametric surface characteristic type. The valid values defined in phigs77.h are:

1 PSCNON
2 PSCWSD
3 PSCISO
4 PSCLMC
5 PSCLWC

LDR The dimension of the data record array, DATREC.

DATREC(LDR)

The data record is packed by PACK DATA RECORD. The contents depend on the parametric surface characteristic type specified:
For PSCTYP = PSCNON or PSCWSD:

IL = the number of integers, 0

RL = the number of real values, 0

SL = the number of strings, 0

For PSCTYP = PSCICO:

IL = the number of integers, 3

IA = an array of integers, with the following values:

IA(1) = the curve placement. Valid values defined in phigs77.h are:

1 PUNIF

2 PNUNIF

IA(2) = the u_count

IA(3) = the v_count

RL = the number of reals, 0

SL = the number of strings, 0

For PSCTYP = PSCLMC or PSCLWC:

IL = 1

IA(1) = number of level curve parameters

RL = 6 + IA(1)

RA = an array of float values.

RA(1) – RA(3) are the *x*, *y*, and *z* coordinates of the origin.

RA(4) – RA(6) are the *x*, *y*, and *z* deltas of the direction vector.

RA(7) – RA(RL) are the level curve parameters.

SL = The number of strings, 0 LSTR = () STR = ()

Execution

When the structure is traversed, SET PARAMETRIC SURFACE CHARACTERISTICS updates the current parametric surface characteristics entry in the traversal state list. These values are used when displaying subsequent parametric surface output primitives. When an element of this type is interpreted, if the specified parametric surface characteristic type is not available on a workstation, or the data record are invalid, type 1 is used and data record contents are workstation-dependent.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

138 Ignoring function, one or more of the fields in the specified data record is inconsistent with the specified type

SEE ALSO

NON-UNIFORM B-SPLINE SURFACE (3PP)

NAME

SET PATTERN REPRESENTATION PLUS – set extended pattern attribute bundle on workstation

SYNOPSIS

C Syntax

```
void
pset_pat_rep_plus ( ws, index, rep )
Pint            ws;     workstation identifier
Pint            index;  pattern bundle index
Ppat_rep_plus   *rep;   pattern representation pointer
```

FORTRAN Syntax

```
SUBROUTINE psepr ( WKID, INDEX, PROWS, PCOLS, COLMOD, IPCOLR,
    PCOLR )
INTEGER   WKID      workstation identifier
INTEGER   INDEX     pattern bundle index
INTEGER   PROWS     pattern dimensions along y-axis
INTEGER   PCOLS     pattern dimensions along x-axis
INTEGER   COLMOD    colour model
INTEGER   IPCOLR    indirect colour index
REAL      PCOLR(3)  direct colour value
```

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

SET PATTERN REPRESENTATION PLUS sets an extended pattern attribute bundle on a workstation.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

ws The identifier of the workstation.

index This is an index to the workstation pattern bundle table.

rep A pointer to a Ppat_rep_plus structure. Ppat_rep_plus is defined as:

```
typedef struct {
        Pint_size   dims;       /* pattern's dimensions */
        Pint        type;       /* colour model */
        Pcoval      *colr_array; /* array of colours */
} Ppat_rep_plus;
```

Pint_size is defined as:

```
typedef struct {
        Pint        size_x;     /* dimension (number of divisions)
                                   along X */
        Pint        size_y;     /* dimension (number of divisions)
```

```
                                        along Y */
} Pint_size;
```

Pcoval is defined as:

```
typedef union {
        Pint            ind;        /* index in workstation colour
                                       bundle table */
        Pcolr_rep       direct;     /* direct colour components */
}Pcoval;
```

Pcolr_rep is defined as:

```
typedef union {
        Prgb            rgb;        /* Red Green Blue colour
                                       specification */
        Pcieluv         cieluv;     /* CIE L*U*V* colour specification */
        Phls            hls;        /* Hue Lightness Saturation colour
                                       specification */
        Phsv            hsv;        /* Hue Saturation Value colour
                                       specification */
        Pdata           unsupp;     /* Colour in unsupported colour
                                       model */
} Pcolr_rep;
```

Prgb is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          red;        /* red, hue, and so on */
        Pfloat          green;      /* green, saturation, lightness,
                                       and so on */
        Pfloat          blue;       /* blue, value, saturation, and
                                       so on */
} Prgb;
```

Pcieluv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          cieluv_x;       /* x coefficient */
        Pfloat          cieluv_y;       /* y coefficient */
        Pfloat          cieluv_y_lum;   /* y luminance */
} Pcieluv;
```

Phls is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;        /* hue */
        Pfloat          lightness;  /* lightness */
        Pfloat          satur;      /* saturation */
```

```
} Phls;

Phsv is defined in phigs.h as follows:
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          satur;          /* saturation */
        Pfloat          value;          /* value */
} Phsv;

Pdata is defined in phigs.h as follows:
typedef struct {
        size_t          size;           /* size of data */
        char            *data           /*pointer to data */
} Pdata;
```

Note: the pset_pat_rep_plus() call is not supported by the current SunPHIGS release, but is included for portability reasons.

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

WKID The identifier of the workstation for which the pattern representation is being defined.

INDEX The index into the specified workstation pattern bundle table.

PROWS The pattern dimensions along the *y*-axis.

PCOLS The pattern dimensions along the *x*-axis.

COLMOD

The colour model. The predefined values are:

0	PINDIR	*Colour Index Specified*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

IPCOLR

An integer index into the workstation colour table for determining pattern colour. This value is only used if COLMOD is equal to PINDIR.

PCOLR(3)

An array of reals containing the pattern colour. This array is unused if COLMOD is equal to PINDIR.

PCOLR(1) = *Red, Hue, and so on*
PCOLR(2) = *Green, Saturation, Lightness, and so on*
PCOLR(3) = *Blue, Value, Saturation, and so on*

Execution

In the pattern table of the workstation state list, the given index is associated with the specified parameters.

A grid of DX by DY cells is specified with DX as horizontal and DY as vertical. If the colour type is indirect colour, then colour is given individually for each cell by the colour index, which is a pointer into the colour table of the workstation. Otherwise, the colour is given individually for each cell by a direct colour value in the specified colour model.

If the workstation supports interior style PATTERN, the pattern table in the workstation state list has predefined entries taken from the workstation description table. A number of these entries are predefined for every workstation of category OUTPUT and OUTIN supporting interior style PATTERN. This function can redefine any table entry, including the predefined ones.

When area defining primitives are displayed during structure traversal, if the currently selected interior style is PATTERN, then the current, or back, interior style index in the PHIGS state list refers to an entry in the extended pattern table. The selection is made either via the interior bundle or individually, depending on the corresponding Aspect Source Flags(ASFs). If the current, or back, interior style index is not present in the extended pattern table, then interior style index one is used. If interior style index one is not present, the result is workstation dependent and the interior style PATTERN is not supported for this workstation. The setting of the corresponding ASFs determines which aspects in the entry are used.

During structure traversal, if the colour type of a specified pattern table entry is indirect colour and if a pattern colour index specified in the pattern colour array is not available on the workstation, then colour index one is used.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, the specified workstation is not open

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

103 Ignoring function, setting this bundle table entry would exceed the maximum number of entries allowed in the workstation bundle table

110 Ignoring function, the specified colour model is not available on the workstation

112 Ignoring function, the pattern index value is less than one

113 Ignoring function, the colour index value is less than zero

116 Ignoring function, one of the dimensions of the pattern colour array is less than one

SEE ALSO

SET INDIVIDUAL ASF (3P)
INQUIRE PATTERN REPRESENTATION PLUS (3PP)
INQUIRE PREDEFINED PATTERN REPRESENTATION PLUS (3PP)
SET PATTERN REPRESENTATION (3P)

NAME

SET POLYLINE COLOUR – create structure element to set polyline colour attribute

SYNOPSIS

C Syntax

```
void
pset_line_colr ( colour )
Pgcolr    *colour;    polyline colour
```

FORTRAN Syntax

```
SUBROUTINE psplco ( COLMOD, ICOLR, COLR )
INTEGER   COLMOD    colour model
INTEGER   ICOLR     colour table index
REAL      COLR(3)   colour triple
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET POLYLINE COLOUR creates a structure element to set the polyline colour attribute.

When the current polyline colour Aspect Source Flag (ASF) is set to INDIVIDUAL, the current value of this attribute applies to the following output primitives:

POLYLINE
POLYLINE 3
POLYLINE SET 3 WITH DATA
NON-UNIFORM B-SPLINE CURVE"

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

All of the following data types are predefined in phigs.h.

colour Polyline colour. Pgcolr is defined as:

```
typedef struct {
        Pint            type;           /* indirect, RGB, CIE, HSV, HLS */
        union {
          Pint          index;          /* colour table index */
          struct {
                Pfloat          x;      /* red, hue, and so on */
                Pfloat          y;      /* green, saturation, lightness,
                                           and so on */
                Pfloat          z;      /* blue, value, saturation, and
                                           so on */
          } general;
        } val;
} Pgcolr;
```

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

COLMOD

The colour model. The predefined values are:

0	PINDIR	*Colour Index Specified*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

ICOLR An integer index into the workstation colour table for determining the polyline colour. This value is only used if COLMOD is equal to PINDIR.

COLR(3)

An array of reals containing the polyline colour. This array is unused if COLMOD is equal to PINDIR.

COLR(1) = *Red, Hue, and so on*
COLR(2) = *Green, Saturation, Lightness, and so on*
COLR(3) = *Blue, Value, Saturation, and so on*

Execution

Depending on the edit mode, a SET POLYLINE COLOUR element with the value polyline colour in the specified colour model is inserted into the open structure after the element pointer or replaces the element pointed at by the element pointer. The element pointer is then updated to point to this SET POLYLINE COLOUR element.

When the current polyline colour ASF is set to INDIVIDUAL, linear curve output primitives, which follow in the structure network are drawn using the current polyline colour.

When the current polyline colour ASF is set to BUNDLED, the polyline colour is taken from the workstation extended representation indicated by the current polyline index. In this case, the polyline colour set with SET POLYLINE COLOUR has no effect.

The standard PHIGS function SET POLYLINE COLOUR INDEX is equivalent of calling to SET POLYLINE COLOUR with type INDIRECT.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

COLOUR (7P)
SET INDIVIDUAL ASF (3P)
INQUIRE POLYLINE REPRESENTATION PLUS (3PP)
SET POLYLINE COLOUR INDEX (3P)

NAME

SET POLYLINE REPRESENTATION PLUS – set extended polyline attribute bundle on a workstation

SYNOPSIS

C Syntax

```
void
pset_line_rep_plus ( ws, index, rep )
Pint                ws;     workstation identifier
Pint                index;  polyline bundle index
Pline_bundle_plus   *rep;   extended polyline representation pointer
```

FORTRAN Syntax

```
SUBROUTINE pseplr ( WKID, INDEX, LTYPE, LWIDTH, COLTYPE, ICOLR,
    COLR, METHOD, ATYPE, AVALUE )
INTEGER   WKID      workstation identifier
INTEGER   INDEX     polyline index
INTEGER   LTYPE     linetype
REAL      LWIDTH    linewidth scale factor
INTEGER   COLTYPE   colour type
INTEGER   ICOLR     polyline colour index
REAL      COLR(3)   colour values
INTEGER   METHOD    polyline shading method
INTEGER   ATYPE     curve approximation type
REAL      AVALUE    curve approximation value
```

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

SET POLYLINE REPRESENTATION PLUS sets a bundle of extended polyline attributes for a specified entry in a workstation extended polyline bundle table.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

ws The identifier of the workstation for which the extended polyline representation is being set.

index The index of the entry being set.

rep A pointer to a Pline_bundle_plus structure. Pline_bundle_plus is defined as:

```
typedef struct {
        Pint        type;       /* line type */
        Pfloat      width;      /* linewidth scale factor */
        Pgcolr      colr;       /* polyline colour */
```

```
        Pint        shad_meth;      /* polyline shading method */
        Pint        approx_type;    /* curve approximation
                                       criteria */
        Pfloat      approx_val;     /* curve approximation
                                       criteria */
    } Pline_bundle_plus;
```

Constants defined for linetype are:

1	PLINE_SOLID	*Solid*
2	PLINE_DASH	*Dashed*
3	PLINE_DOT	*Dotted*
4	PLINE_DASH_DOT	*Dot-dashed*

Pgcolr is defined as:

```
typedef struct {
    Pint            type;       /* indirect, RGB, CIE, HSV, HLS */
    union {
       Pint         ind;        /* index in workstation colour
                                   bundle table */
       struct {
          Pfloat    x;          /* red, hue, and so on */
          Pfloat    y;          /* green, saturation, lightness,
                                   and so on */
          Pfloat    z;          /* blue, value, saturation, and
                                   so on */
       } general;
    } val;
} Pgcolr;
```

Constants defined for colour type are:

0	PINDIRECT	*Indirect*
1	PMODEL_RGB	*Red, Green, and Blue*
2	PMODEL_CIELUV	*CIE*
3	PMODEL_HSV	*Hue, Saturation, and Value*
4	PMODEL_HLS	*Hue, Lightness, and Saturation*

The *ind* member of the val union is used for type PINDIRECT. The *general* member is used for the other types.

Constants defined for polyline shading method are:

1	PSD_NONE	*No Shading*
2	PSD_COLOUR	*Colour Interpolation Shading*
3	PSD_DOT_PRODUCT	*Dot Product Interpolation Shading*
4	PSD_NORMAL	*Normal Interpolation Shading*

Constants defined for curve approximation type are:

1	PCURV_WS_DEP

2	PCURV_CONSTANT_PARAMETRIC_BETWEEN_KNOTS	
3	PCURV_CHORDAL_SIZE_WC	
4	PCURV_CHORDAL_SIZE_NPC	
5	PCURV_CHORDAL_SIZE_DC	
6	PCURV_CHORDAL_DEVIATION_WC	
7	PCURV_CHORDAL_DEVIATION_NPC	
8	PCURV_CHORDAL_DEVIATION_DC	

See SET CURVE APPROXIMATION CRITERIA for a description of these constants and their use of the curve approximation value.

FORTRAN Input Parameters

The following data types are predefined in phigs77.h.

WKID The identifier of the workstation for which the extended polyline representation is being defined.

INDEX The index of the entry being defined.

LTYPE An integer value specifying a linetype. Constants defined for linetype are:

1	PLSOLI	*Solid*
2	PLDASH	*Dashed*
3	PLDOT	*Dotted*
4	PLDASD	*Dot-dashed*
0	PLNLONGDASH	*Long-dashed*
–1	PLNDOTDASHDOT	*Dot-dashed-dot-dotted*
–2	PLNCENTER	*Center (long-short dashed)*
–3	PLNPHANTOM	*Phantom (long-short-short dashed)*

See SET LINETYPE in PHIGS for caveats regarding the *Center* and *Phantom* linetypes.

LWIDTH

The linewidth scale factor.

COLTYPE

The colour type. Constants defined for colour type are:

0	PINDIR	*Indirect*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

ICOLR An integer index into the workstation colour table for determining the line colour. This value is only used if COLTYPE is equal to PINDIR.

COLR(3)

An array of reals containing the colour for the line. This array is unused if COLTYPE is equal to PINDIR.

COLR(1) = *Red, Hue, and so on*
COLR(2) = *Green, Saturation, Lightness, and so on*
COLR(3) = *Blue, Value, Saturation, and so on*

METHOD

The polyline shading method. Constants defined for polyline shading method are:

1	PSDNON	*No Shading*
2	PSDCOL	*Colour Interpolation Shading*

ATYPE The curve approximation type. Constants defined for curve approximation type are:

1	PCAWSD	*Workstation-dependent*
2	PCAEPK	*Constant Parametric Subdivision Between Knots*
3	PCACSW	*Chordal Size in WC*
4	PCACSN	*Chordal Size in NPC*
5	PCACSD	*Chordal Size in DC*
6	PCACDW	*Chordal Deviation in WC*
7	PCACDN	*Chordal Deviation in NPC*
8	PCACDD	*Chordal Deviation in DC*

See SET CURVE APPROXIMATION CRITERIA for a description of their meaning.

AVALUE

The curve approximation value. Its use is dependent upon the approximation type. See SET CURVE APPROXIMATION CRITERIA for a description of its use.

Execution

SET POLYLINE REPRESENTATION PLUS sets the specified extended polyline bundle table entry in the specified workstation state list. The extended polyline bundle table contains bundled entries of the PHIGS and SunPHIGS Extension polyline attributes. SET POLYLINE REPRESENTATION PLUS sets the individual entries of this table. During traversal, the attribute values of these bundles are used when the corresponding aspect source flag in the workstation state list is set to BUNDLED.

The PHIGS polyline bundle table is a portion of the extended table. The function SET POLYLINE REPRESENTATION sets the following attributes in the extended table: linetype, linewidth scale factor, and colour index (colour type of *Indirect* is assumed).

During structure traversal the current polyline index refers to an entry in the workstation extended polyline bundle table. Depending on the settings of the aspect source flags, the contents of that bundle table entry are used to determine the attributes associated with output primitives that use the extended polyline bundle table. These primitives are:

POLYLINE
POLYLINE 3
POLYLINE SET 3 WITH DATA
NON-UNIFORM B-SPLINE CURVE
GDP (*Linear Primitives*)
GDP3 (*Linear Primitives*)
CELL ARRAY
CELL ARRAY 3 (*When Performing the Minimal Simulation*)

ERRORS

003	Ignoring function, function requires state (PHOP, WSOP, *, *)
054	Ignoring function, the specified workstation is not open
059	Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)
100	Ignoring function, the bundle index value is less than one
103	Ignoring function, setting this bundle table entry would exceed the maximum number of entries allowed in the workstation bundle table
104	Ignoring function, the specified linetype is not available on the specified workstation
110	Ignoring function, the specified colour model is not available on the workstation
113	Ignoring function, the colour index value is less than zero
605	Ignoring function, the specified polyline shading method is not available on the workstation

SEE ALSO

INQUIRE POLYLINE FACILITIES PLUS (3PP)
SET INDIVIDUAL ASF (3P)
INQUIRE POLYLINE REPRESENTATION PLUS (3PP)
SET CURVE APPROXIMATION CRITERIA (3PP)
SET POLYLINE REPRESENTATION (3P)

NAME

SET POLYLINE SHADING METHOD – create structure element that sets current polyline shading method

SYNOPSIS

C Syntax

void
pset_line_shad_meth (method)
Pint method; *shading method*

FORTRAN Syntax

SUBROUTINE psplsm (METHOD)
INTEGER METHOD *shading method*

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET POLYLINE SHADING METHOD places a polyline structure element with the value polyline shading method into the open structure. When the current polyline shading method Aspect Source Flag (ASF) is set to INDIVIDUAL, the current value of this attribute applies to the following output primitives:

POLYLINE
POLYLINE 3
POLYLINE SET 3 WITH DATA
NON-UNIFORM B-SPLINE CURVE

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameter

The following data type is predefined in phigs.h.

method The polyline shading method.

1	PSD_NONE	*No Shading*
2	PSD_COLOUR	*Colour Interpolation Shading*

FORTRAN Input Parameter

The following data type is predefined in phigs77.h.

METHOD

The value of the shading method.

1	PSDNON	*No Shading*
2	PSDCOL	*Colour Interpolation Shading*

Execution

Depending on the edit mode, a SET POLYLINE SHADING METHOD element is inserted into the open structure after the element pointer, or replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the newly created SET POLYLINE SHADING METHOD element.

When the structure is traversed, the current polyline shading method entry in the traversal state list is set to the specified value. This value is used with subsequent line output primitives when the current polyline shading method ASF entry in the traversal state list is INDIVIDUAL. During traversal, if the specified shading method is not available on a workstation, method one will be used on that workstation.

When the current polyline shading method ASF is set to BUNDLED, the polyline shading method is taken from the workstation extended representation indicated by the current polyline index. In this case, the polyline shading method set with SET POLYLINE SHADING METHOD has no effect.

When the shading method is NONE, each segment of the polyline or curve will be drawn in a single colour. If the primitive specifies per-vertex colour, then each segment will be drawn using the colour associated with the first vertex of the segment.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

INTRO (3PP)
POLYLINE SET 3 WITH DATA (3PP)
PHIGS WORKSTATION DESCRIPTION TABLE (7P)
NON-UNIFORM B-SPLINE CURVE (3PP)

NAME

SET POLYMARKER COLOUR – create structure element to set current polymarker colour

SYNOPSIS

C Syntax

```
void
pset_marker_colr ( colour )
Pgcolr    *colour;    polymarker colour
```

FORTRAN Syntax

```
SUBROUTINE pspmco ( COLMOD, ICOLR, COLR )
INTEGER   COLMOD    colour model
INTEGER   ICOLR     colour table index
REAL      COLR(3)   colour triple
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET POLYMARKER COLOUR creates a structure element to set the current polymarker colour.

When the current polymarker colour Aspect Source Flag ASF is set to INDIVIDUAL, the current value of this attribute applies to the following output primitives:

POLYMARKER
POLYMARKER 3

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameter

The following data type is predefined in phigs.h.

colour Polymarker colour. Pgcolr is defined as:

```
typedef struct {
        Pint            type;           /* indirect, RGB, CIE, HSV, HLS */
        union {
          Pint          index;          /* colour table index */
          struct {
                Pfloat          x;      /* red, hue, and so on */
                Pfloat          y;      /* green, saturation, lightness,
                                           and so on */
                Pfloat          z;      /* blue, value, saturation, and
                                           so on */
          } general;
        } val;
} Pgcolr;
```

FORTRAN Input Parameters

The following data types are predefined in phigs77.h.

COLMOD

The colour model. The predefined values are:

0	PINDIR	*Indirect*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

ICOLR An integer index into the workstation colour table for determining the polymarker colour. This value is only used if COLMOD is equal to PINDIR.

COLR(3)

An array of reals containing the polymarker colour. This array is unused if COLMOD is equal to PINDIR.

COLR(1) = *Red, Hue, and so on*
COLR(2) = *Green, Saturation, Lightness, and so on*
COLR(3) = *Blue, Value, Saturation, and so on*

Execution

Depending on the edit mode, a SET POLYMARKER COLOUR element with the value polymarker colour in the specified colour model is inserted into the open structure after the element pointer or replaces the element pointed at by the element pointer. The element pointer is then updated to point to this SET POLYMARKER COLOUR element.

When the current polymarker colour ASF is set to INDIVIDUAL, marker output primitives, which follow in the structure network are drawn using the current polymarker colour.

When the current polymarker colour ASF is set to BUNDLED, the polymarker colour is taken from the workstation extended representation indicated by the current polymarker index. In this case, the polymarker colour set with SET POLYMARKER COLOUR has no effect.

The standard PHIGS function SET POLYMARKER COLOUR INDEX is equivalent to calling to SET POLYMARKER COLOUR with type INDIRECT.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

COLOUR (7P)
SET INDIVIDUAL ASF (3P)
SET POLYMARKER REPRESENTATION PLUS (3PP)
SET POLYMARKER COLOUR INDEX (3P)

NAME

SET REFLECTANCE EQUATION – create structure element to set current interior reflectance equation

SYNOPSIS

C Syntax

void
pset_refl_eqn (equation)
Pint equation; *reflectance equation*

FORTRAN Syntax

SUBROUTINE psireq (REFLEQ)
INTEGER REFLEQ *reflectance equation*

Required PHIGS Operating States

(PHOP, ∗, STOP, ∗)

DESCRIPTION

Purpose

SET REFLECTANCE EQUATION creates a structure element to set the current interior reflectance equation.

When the current interior reflectance equation Aspect Source Flag (ASF) is set to INDIVIDUAL, the current value of this attribute defines the interior reflectance equation to be applied to the following output primitives:

FILL AREA
FILL AREA 3
FILL AREA SET
FILL AREA SET 3
FILL AREA SET 3 WITH DATA
NON-UNIFORM B-SPLINE SURFACE
SET OF FILL AREA SET 3 WITH DATA
QUADRILATERAL MESH 3 WITH DATA
TRIANGLE STRIP 3 WITH DATA

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameter

equation

The reflectance equation. The possible values defined in phigs.h are:

1	PREFL_NONE	*No Reflectance Calculation Performed*
2	PREFL_AMBIENT	*Use Ambient Term*
3	PREFL_AMB_DIFF	*Use Ambient and Diffuse Terms*
4	PREFL_AMB_DIFF_SPEC	*Use Ambient, Diffuse, and Specular Terms*

FORTRAN Input Parameter

The following data type is predefined in phigs77.h.

REFLEQ

The reflectance equation. The possible values are:

1	PRNONE	*No Reflectance Calculation Performed*
2	PRAMB	*Use Ambient Term*
3	PRAMBD	*Use Ambient and Diffuse Terms*
4	PRAMDS	*Use Ambient, Diffuse, and Specular Terms*

Execution

Depending on the edit mode, a SET REFLECTANCE EQUATION element with the value reflectance equation is inserted into the open structure after the element pointer or replaces the element pointed at by the element pointer. The element pointer is then updated to point to this SET REFLECTANCE EQUATION element.

At traversal time, if the specified reflectance equation is not defined, then equation one is substituted.

When the current interior reflectance equation ASF is set to INDIVIDUAL, area-defining output primitives which follow in the structure network are filled using the current interior reflectance equation.

When the current interior reflectance equation ASF is set to BUNDLED, the interior reflectance equation is taken from the workstation extended representation indicated by the current interior index. In this case, the interior reflectance equation set with SET REFLECTANCE EQUATION has no effect.

ERRORS

002 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

SET INDIVIDUAL ASF (3P)
SET INTERIOR REPRESENTATION PLUS (3PP)

NAME

SET REFLECTANCE PROPERTIES – create structure element to set surface area properties

SYNOPSIS

C Syntax

```
void
pset_refl_props ( properties )
Prefl_props    *properties;    surface area properties
```

FORTRAN Syntax

```
SUBROUTINE psap ( AMB, DIF, SPEC, COLTYPE, ICOLR, COLR, EXP, TRANS )
REAL       AMB       ambient reflectance coefficient
REAL       DIF       diffuse reflectance coefficient
REAL       SPEC      specular reflectance coefficient
INTEGER    COLTYPE   colour type
INTEGER    ICOLR     indirect colour
REAL       COLR(3)   specular colour
REAL       EXP       specular exponent
REAL       TRANS     transparency coefficient
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET REFLECTANCE PROPERTIES creates a structure element to set surface area properties. When the current area properties Aspect Source Flag (ASF) is set to INDIVIDUAL, the current value of this attribute applies to the following output primitives:

FILL AREA
FILL AREA SET
FILL AREA SET 3 WITH DATA
QUADRILATERAL MESH 3 WITH DATA
TRIANGLE STRIP 3 WITH DATA
SET OF FILL AREA SET 3 WITH DATA
NON-UNIFORM B-SPLINE SURFACE

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameter

properties

A pointer to a Prefl_props structure that specifies the area properties. Prefl_props is defined in phigs.h as follows:

```
typedef struct {
        Pfloat  ambient_coef;   /* ambient reflectance coefficient */
        Pfloat  diffuse_coef;   /* diffuse reflectance coefficient */
        Pfloat  specular_coef;  /* specular reflectance coefficient */
        Pgcolr  specular_colr;  /* specular colour */
```

```
        Pfloat    specular_exp;    /* specular exponent */
        Pfloat    transpar_coef;   /* transparency coefficient */
} Prefl_props;
```

Pgcolr is defined as:

```
typedef struct {
        Pint            type;           /* indirect, RGB, CIE, HSV, HLS */
        union {
          Pint          ind;            /* colour table index */
          struct {
                Pfloat          x;      /* red, hue, and so on */
                Pfloat          y;      /* green, saturation, lightness,
                                           and so on */
                Pfloat          z;      /* blue, value, saturation, and so on */
          } general;
        } val;
} Pgcolr;
```

The color type is one of the following:

0	PINDIRECT	*Indirect*
1	PMODEL_RGB	*Red, Green, and Blue*
2	PMODEL_CIELUV	*CIE Colour Model*
3	PMODEL_HSV	*Hue, Saturation, and Value*
4	PMODEL_HLS	*Hue, Lightness, and Saturation*

FORTRAN Input Parameters

The following data types are predefined in phigs77.h.

AMB The ambient reflectance coefficient.

DIF The diffuse reflectance coefficient.

SPEC The specular reflectance coefficient.

COLTYPE

The colour type. The predefined values are:

0	PINDIR	*Indirect*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

ICOLR An integer value which is an index into the workstation colour table for determining the specular colour. ICOLR is used if COLTYPE is set to PINDIR.

COLR(3)

The specular colour. An array of reals containing the specular colour. COLR(3) is used if COLTYPE is set to other than PINDIR.

COLR(1) = *Red, Hue, and so on*
COLR(2) = *Green, Saturation, Lightness, and so on*
COLR(3) = *Blue, Value, Saturation, and so on*

EXP The specular exponent.

TRANS The transparency coefficient.

Execution

Depending upon the edit mode, a SET REFLECTANCE PROPERTIES element is inserted into the open structure after the element pointer or replaces the element pointed at by the element pointer. The element pointer is then updated to point to this SET REFLECTANCE PROPERTIES element.

When the structure is traversed, the current area properties entry in the traversal state list is set to the specified value. This value is used with subsequent filled area output primitives when the current area properties ASF entry in the traversal state list is INDIVIDUAL. During traversal, if the specified area properties are not available on a workstation, method one (*None*) will be used on that workstation.

When the current area properties ASF is set to BUNDLED, the area properties are taken from the workstation extended representation indicated by the current interior index. In this case, the area properties set with SET REFLECTANCE PROPERTIES have no effect.

SunPHIGS has accepted transparency coefficients since release 1.1; however, only GT workstations use these values. Valid transparency coefficients are between 0 and 1 where:

0.0 means no transparency (entirely opaque)
1.0 means entirely transparent (invisible)

The area properties supported by a workstation type can be determined with the function INQUIRE EXTENDED INTERIOR FACILITIES.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

INTRO (3PP)
FILL AREA (3P)
FILL AREA SET (3P)
FILL AREA 3 (3P)
FILL AREA SET 3 WITH DATA (3PP)
SET INDIVIDUAL ASF (3P)
SET INTERIOR REPRESENTATION PLUS (3PP)
SET INTERIOR REPRESENTATION PLUS (3PP)
INQUIRE INTERIOR FACILITIES PLUS (3PP)
NON-UNIFORM B-SPLINE SURFACE (3PP)

NAME

SET SURFACE APPROXIMATION CRITERIA – create structure element to set surface approximation criteria

SYNOPSIS

C Syntax

void
pset_surf_approx (type, uvalue, vvalue)
Pint type; *approximation type*
Pfloat uvalue; *approximation value for u dimension*
Pfloat vvalue; *approximation value for v dimension*

FORTRAN Syntax

SUBROUTINE pssapx (TYPE, UVAL, VVAL)
INTEGER TYPE *approximation type*
REAL UVAL *approximation value for u dimension*
REAL VVAL *approximation value for v dimension*

Required PHIGS Operating States

(PHOP, ∗, STOP, ∗)

DESCRIPTION

Purpose

SET SURFACE APPROXIMATION CRITERIA places into the currently-open structure a structure element that specifies the surface approximation type and values. The surface approximation type controls the display precision of non-uniform B-spline surfaces by specifying the method to use to render the surface. The approximation values are used in a type-specific way, as described in the *Execution* section below.

When the current surface approximation criteria Aspect Source Flag (ASF) is set to INDIVIDUAL, the current value of this attribute defines the surface approximation criteria to be applied to the NON-UNIFORM B-SPLINE SURFACE output primitive.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

type The surface approximation type. Possible values defined in phigs.h are:

1 PSURF_WS_DEP
2 PSURF_CONSTANT_PARAMETRIC_BETWEEN_KNOTS
3 PSURF_CHORDAL_SIZE_WC
4 PSURF_CHORDAL_SIZE_NPC
5 PSURF_CHORDAL_SIZE_DC
6 PSURF_PLANAR_DEVIATION_WC
7 PSURF_PLANAR_DEVIATION_NPC
8 PSURF_PLANAR_DEVIATION_DC
9 PSURF_RELATIVE_WC
10 PSURF_RELATIVE_NPC
11 PSURF_RELATIVE_DC

uvalue, vvalue

The values to use when applying the specified approximation method. Valid values depend on the approximation method. The value for the *u* dimension is *uvalue*. The value for the *v* dimension is *vvalue*. See the *Execution* section, below, for a description of the use of these values by each approximation method.

FORTRAN Input Parameters

The following data types are predefined in phigs77.h.

TYPE The surface approximation type. Predefined values are:

1	PSAWSD	*Workstation-dependent*
2	PSAEPK	*Constant Parametric Subdivision Between Knots*
3	PSACSW	*Chordal Size in WC*
4	PSACSN	*Chordal Size in NPC*
5	PSACSD	*Chordal Size in DC*
6	PSAPDW	*Planar Deviation in WC*
7	PSAPDN	*Planar Deviation in NPC*
8	PSAPDD	*Planar Deviation in DC*
9	PSARWC	*Relative quality in WC*
10	PSARNP	*Relative quality in NPC*
11	PSARDC	*Relative quality in DC*

UVAL, VVAL

The values to use when applying the specified approximation method. Valid values depend on the approximation type. *UVAL* is the value for the *u* dimension. *VVAL* is the value for the *v* dimension. See the *Execution* section below for a description of the use of these values by each approximation method.

Execution

Depending on the edit mode, a SET SURFACE APPROXIMATION CRITERIA element is either inserted into the open structure after the element pointer, or replaces the element pointed at by the element pointer. The element pointer is then updated to point to this SET SURFACE APPROXIMATION CRITERIA element.

The approximation methods corresponding to the defined approximation types and the use of the approximation values by each method are:

Workstation-dependent, Type 1

Use a workstation-dependent approximation method. Currently all SunPHIGS workstation types use method 2 (*Constant Parametric Subdivision Between Knots*) as the workstation-dependent method. The approximation values have the same usage as for that method.

Constant Parametric Subdivision Between Knots, Type 2

Tessellate using equal parametric increments between each pair of knots. The *uvalue* and *vvalue* indicate the minimum number of sample points between each pair of adjacent knots in the U and V direction. The integer portions of the approximation values are used and interpreted as follows:

≤ 0: Sample points are at the parameter range limits and at the knots

within the parameter range limits.

> 0: Sample points are at the parameter range limits and at the knots within the parameter range limits. In addition, sample points are at the specified number of positions between each pair of knots, but only if the position is within the parameter range limits.

Chordal Size in WC, Type 3

Chordal Size in NPC, Type 4

Chordal Size in DC, Type 5

Ensure that enough sample points will be generated so that the distance between two consecutive points on the surface in U and V parametric directions will be, less than or equal to the respective values of *uvalue* and *vvalue*. The distance is expressed in World Coordinates (WC), Normalized Projection Coordinates (NPC), or Device Coordinates (DC).

Planar Deviation in WC, Type 6

Planar Deviation in NPC, Type 7

Planar Deviation in DC, Type 8

Ensure that enough sample points will be generated so that the distance between the surface and the approximating facet will be less than or equal to the value of *uvalue*. The distance is expressed in World Coordinates (WC), Normalized Projection Coordinates (NPC), or Device Coordinates (DC). *vvalue* is ignored.

Relative Quality in WC, Type 9

Relative Quality in NPC, Type 10

Relative Quality in DC, Type 11

A single floating point value between 0 and 1 in the *uvalue* field indicates a relative quality of rendering to be maintained independent of scaling in World Coordinates (WC), Normalized Projection Coordinates (NPC), or Device Coordinates (DC). *vvalue* is ignored.

The approximation types supported by a workstation type can be determined with the function INQUIRE CURVE AND SURFACE FACILITIES, or by consulting the relevant entries in PHIGS WORKSTATION DESCRIPTION TABLE (7P).

When the current surface approximation criteria ASF is set to INDIVIDUAL, area-defining output primitives that follow in the structure network are filled by using the current curve approximation criteria style.

When the current surface approximation criteria ASF is set to BUNDLED, the surface approximation criteria is taken from the workstation extended representation indicated by the current interior index. In this case, the surface approximation criteria set with SET SURFACE APPROXIMATION CRITERIA has no effect.

ERRORS 005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

NON-UNIFORM B-SPLINE SURFACE (3PP)
INQUIRE CURVE AND SURFACE FACILITIES (3PP)
SET INDIVIDUAL ASF (3P)
SET INTERIOR REPRESENTATION PLUS (3PP)
PHIGS WORKSTATION DESCRIPTION TABLE (7P)
GENERALIZED DRAWING PRIMITIVE 3 -19 (3P)

NAME

SET TEXT COLOUR – create structure element to set text colour attribute

SYNOPSIS

C Syntax

```
void
pset_text_colr ( colour )
Pgcolr    *colour;    text colour
```

FORTRAN Syntax

```
SUBROUTINE pstxco ( COLMOD, ICOLR, COLR )
                INTEGER    COLMOD    colour model
                INTEGER    ICOLR     colour table index
                REAL       COLR(3)   colour triple
```

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

SET TEXT COLOUR creates a structure element to set the text colour attribute.

When the current interior shading method Aspect Source Flag ASF is set to INDIVIDUAL, the current value of this attribute applies to the following output primitives:

TEXT
TEXT 3
ANNOTATION TEXT RELATIVE
ANNOTATION TEXT RELATIVE 3
GENERALIZED DRAWING PRIMITIVE -15
GENERALIZED DRAWING PRIMITIVE 3 -15
GENERALIZED DRAWING PRIMITIVE -16
GENERALIZED DRAWING PRIMITIVE 3 -16
GENERALIZED DRAWING PRIMITIVE -17
GENERALIZED DRAWING PRIMITIVE 3 -17
GENERALIZED DRAWING PRIMITIVE -18
GENERALIZED DRAWING PRIMITIVE 3 -18

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameter

The following data types are predefined in phigs.h.

colour Text colour. Pgcolr is defined as:

```
typedef struct {
        Pint            type;           /* indirect, RGB, CIE, HSV, HLS */
        union {
          Pint          index;          /* colour table index */
          struct {
                Pfloat          x;      /* red, hue, and so on */
```

```
                Pfloat      y;      /* green, saturation, lightness,
                                       and so on */
                Pfloat      z;      /* blue, value, saturation, and
                                       so on */
            } general;
        } val;
} Pgcolr;
```

FORTRAN Input Parameters

The following data types are predefined in phigs77.h.

COLMOD

The colour model. The predefined values are:

0	PINDIR	*Colour Index Specified*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

ICOLR An integer index into the workstation colour table for determining the text colour. This value is used only if COLMOD is equal to PINDIR.

COLR(3)

An array of reals containing the text colour. This array is unused if COLMOD is equal to PINDIR.

COLR(1) = *Red, Hue, and so on*
COLR(2) = *Green, Saturation, Lightness, and so on*
COLR(3) = *Blue, Value, Saturation, and so on*

Execution

Depending on the edit mode, a SET TEXT COLOUR element with the value text colour in the specified colour model is inserted into the open structure after the element pointer or replaces the element pointed at by the element pointer. The element pointer is then updated to point to this SET TEXT COLOUR element.

When the current text colour ASF is set to INDIVIDUAL, text output primitives that follow in the structure network are drawn using the current text colour.

When the current text colour ASF is set to BUNDLED, the text colour is taken from the workstation extended representation indicated by the current text index. In this case, the text colour set with SET TEXT COLOUR has no effect.

The standard PHIGS function SET TEXT COLOUR INDEX is equivalent to calling to SET TEXT COLOUR with type INDIRECT.

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

SEE ALSO

COLOUR (7P)
SET TEXT REPRESENTATION PLUS (3PP)
SET INDIVIDUAL ASF (3P)
SET TEXT COLOUR INDEX (3P)

NAME

SET TEXT REPRESENTATION PLUS – define extended text attribute representation bundle on workstation

SYNOPSIS

C Syntax

```
void
pset_text_rep_plus ( ws, index, rep )
Pint                ws;     workstation identifier
Pint                index;  text bundle index
Ptext_bundle_plus   *rep;   text representation pointer
```

FORTRAN Syntax

```
SUBROUTINE psetr ( WKID, INDEX, TFONT, TPREC, TCEXPF, TSPACE,
    COLMOD, ITCOLR, TCOLR )
INTEGER   WKID      workstation identifier
INTEGER   INDEX     text bundle index
INTEGER   TFONT     text font
INTEGER   TPREC     text precision
REAL      TCEXPF    text character expansion factor
REAL      TSPACE    text character spacing
INTEGER   COLMOD    colour model
INTEGER   ITCOLR    indirect colour index
REAL      TCOLR(3)  direct colour value
```

Required PHIGS Operating States

(PHOP, WSOP, *, *)

DESCRIPTION

Purpose

SET TEXT REPRESENTATION PLUS defines an extended text attribute representation bundle on a workstation.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

ws The identifier of the workstation.

index This is an index to the workstation text bundle table.

rep A pointer to a Ptext_bundle_plus structure. Ptext_bundle_plus is defined as:

```
typedef struct {
        Pint        font;       /* text font */
        Ptext_prec  prec;       /* text precision */
        Pfloat      char_expan; /* character expansion
                                   factor */
        Pfloat      char_space; /* character spacing */
        Pgcolr      colr_ind;   /* text colour */
} Ptext_bundle_plus;
```

Ptext_prec is defined as:

```
typedef enum {
        PPREC_STRING,
        PPREC_CHAR,
        PPREC_STROKE
} Ptext_prec;
```

Pgcolr is defined as:

```
typedef struct {
        Pint            type;   /* indirect, RGB, CIE, HSV, HLS */
        union {
           Pint         ind;    /* index in workstation colour
                                   bundle table */
           struct {
              Pfloat    x;      /* red, hue, and so on */
              Pfloat    y;      /* green, saturation, lightness,
                                   and so on */
              Pfloat    z;      /* blue, value, saturation, and
                                   so on */
           } general;
        } val;
} Pgcolr;
```

FORTRAN Input Parameters

The following data types are predefined in phigs77.h.

WKID The identifier of the workstation for which the text representation is being defined.

INDEX The index into the specified workstation text bundle table.

TFONT The text font.

TPREC The text precision.

TCEXPF
The text character expansion factor.

TSPACE
The text character spacing.

COLMOD
The colour model. The predefined values are:

0	PINDIR	*Colour Index Specified*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

ITCOLR

An integer index into the workstation colour table for determining the text colour. This value is only used if COLMOD is equal to PINDIR.

TCOLR(3)

An array of reals containing the text colour. This array is unused if COLMOD is equal to PINDIR.

TCOLR(1) = *Red, Hue, and so on*
TCOLR(2) = *Green, Saturation, Lightness, and so on*
TCOLR(3) = *Blue, Value, Saturation, and so on*

Execution

In the text bundle table of the workstation state list, the given index is associated with the specified parameters.

The text precision value determines the fidelity with which the other aspects are used. The values of text precision, in order of increasing fidelity, are STRING, CHAR, and STROKE. During structure traversal, if the specified text precision is not available on a workstation, the value STRING is used on that workstation.

The character expansion factor specifies the deviation of the width to height ratio of the characters from the ratio indicated by the font designer. During structure traversal, only the magnitude of the character expansion factor is considered. If the result of taking the absolute value of the character expansion factor value is less than the smallest supported character expansion factor on a workstation, the smallest supported character expansion factor on that workstation is used.

Character spacing specifies how much additional space is inserted between two adjacent character bodies. Character spacing is specified as a fraction of the font-nominal character height.

Text colour is a general colour, either an indirect or a direct colour in a specified colour model. If an indirect colour is specified and the specified colour index is not available during structure traversal, then colour index one is used.

The extended text bundle table in the workstation state list has predefined entries taken from the workstation description table. A number of these entries are predefined for every workstation of category OUTPUT or OUTIN. Any table entry, including the predefined entries, may be redefined with this function.

When either text, or annotation text, is displayed during structure traversal, the current text index in the PHIGS traversal state list refers to an entry in the extended text bundle table. If the current text index is not present in the extended text bundle table, then text index one is used. The setting of the corresponding Aspect Source Flags(ASFS) determines which aspects in the entry are used.

ERRORS

003 Ignoring function, function requires state (PHOP, WSOP, *, *)

054 Ignoring function, the specified workstation is not open

059 Ignoring function, the specified workstation does not have output capability (that is, the workstation category is not OUTPUT, OUTIN, or MO)

103	Ignoring function, setting this bundle table entry would exceed the maximum number of entries allowed in the workstation bundle table
106	Ignoring function, the specified font is not available for the requested text precision on the specified workstation
110	Ignoring function, the specified colour model is not available on the workstation
113	Ignoring function, the colour index value is less than zero

SEE ALSO

SET INDIVIDUAL ASF (3P)
INQUIRE TEXT REPRESENTATION PLUS (3PP)
SET TEXT REPRESENTATION (3P)

NAME

TRIANGLE STRIP 3 WITH DATA – creates a 3D triangle strip structure element that includes colour and shading data

SYNOPSIS

C Syntax

void
ptri_strip3_data (fflag, vflag, colr_type, nv, fdata, vdata)

Pint	fflag;	*what data is specified per facet*
Pint	vflag;	*what data is specified per vertex*
Pint	colr_type;	*colour type*
Pint	nv;	*number of vertices*
Pfacet_data_arr3	*fdata;	*facet data*
Pfacet_vdata_arr3	*vdata;	*facet vertex data*

FORTRAN Syntax

SUBROUTINE ptsd3 (FFLAG, VFLAG, COLTYP, NV, IFCOLR, FCOLR, FNORM, COORDS, IVCOLR, VCOLR, VNORM, IMPL)

INTEGER	FFLAG	*data per facet flag*
INTEGER	VFLAG	*data per vertex flag*
INTEGER	COLTYP	*colour type*
INTEGER	NV	*number of vertices*
INTEGER	IFCOLR(NV-2)	*array of indirect facet colours*
REAL	FCOLR(3,NV-2)	*array of facet colours*
REAL	FNORM(3,NV-2)	*array of facet normals*
REAL	COORDS(3,NV)	*array of vertex coordinates*
INTEGER	IVCOLR(NV)	*array of indirect vertex colours*
REAL	VCOLR(3,NV)	*array of vertex colours*
REAL	VNORM(3,NV)	*array of vertex normals*
REAL	IMPL(*)	*implementation defined vertex data*

Required PHIGS Operating States

(PHOP, *, STOP, *)

DESCRIPTION

Purpose

TRIANGLE STRIP 3 WITH DATA places a 3D triangle strip primitive with colour and shading data into the open structure.

This is a SunPHIGS Extension function based on PHIGS PLUS and is not part of the PHIGS standard.

C Input Parameters

fflag The data per facet flag specifies the available data for each facet in the strip.

0	PFACET_NONE	*No Facet Coordinates Specified*
1	PFACET_COLOUR	*Facet Colours Specified*
2	PFACET_NORMAL	*Facet Normal Specified*
3	PFACET_COLOUR_NORMAL	*Facet Normal and Colours Specified*

vflag The data per vertex flag specifies the available data for each vertex of the primitive.

0	PVERT_COORD	*Coordinates Specified*
1	PVERT_COORD_COLOUR	*Coordinates and Colours Specified*
2	PVERT_COORD_NORMAL	*Coordinates and Normals Specified*
3	PVERT_COORD_COLOUR_NORMAL	*Coordinates, Colours, and Normals Specified*

colr_type

The colour type for specified vertex colours.

0	PINDIRECT	*Indirect*
1	PMODEL_RGB	*Red, Green, and Blue*
2	PMODEL_CIELUV	*CIE Colour Model*
3	PMODEL_HSV	*Hue, Saturation, and Value*
4	PMODEL_HLS	*Hue, Lightness, and Saturation*

nv The number of vertices.

fdata A pointer, to a Pfacet_data_arr3 structure, that specifies the colours and / or norms information for each facet. Pfacet_data_arr3 is defined as:

```
typedef union {
        Pcoval      *colrs;     /* array (nv–2) of colours */
        Pvec3       *norms;     /* array (nv–2) of unit normals */
        Pconorm3    *conorms;   /* array (n–2) of colours and
                                   normals */
        /* implementation-dependent types can go here */
} Pfacet_data_arr3;
```

Pcoval is defined as:

```
typedef union {
        Pint        ind;        /* index in workstation colour
                                   bundle table */
        Pcolr_rep   direct;     /* direct colour components */
} Pcoval;
```

Pcolr_rep is defined as:

```
typedef union {
        Prgb        rgb;        /* Red Green Blue colour specification */
        Pcieluv     cieluv;     /* CIE L*U*V* colour specification */
        Phls        hls;        /* Hue Lightness Saturation colour
                                   specification */
        Phsv        hsv;        /* Hue Saturation Value colour
                                   specification */
```

```
        Pdata       unsupp;     /* Colour in unsupported colour model */
} Pcolr_rep
```

Prgb is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          red;            /* red, hue, and so on */
        Pfloat          green;          /* green, saturation, lightness,
                                           and so on */
        Pfloat          blue;           /* blue, value, saturation,
                                           and so on */
} Prgb;
```

Pcieluv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          cieluv_x;       /* x coefficient */
        Pfloat          cieluv_y;       /* y coefficient */
        Pfloat          cieluv_y_lum;   /* y luminance */
} Pcieluv;
```

Phls is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          lightness;      /* lightness */
        Pfloat          satur;          /* saturation */
} Phls;
```

Phsv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          satur;          /* saturation */
        Pfloat          value;          /* value */
```

Pdata is defined in phigs.h as follows:

```
typedef struct {
        size_t          size;           /* size of data */
        char            *data           /* pointer to data */
} Pdata;
```

Pvec3 is defined as:

```
typedef struct {
        Pfloat          delta_x;        /* x magnitude */
        Pfloat          delta_y;        /* y magnitude */
        Pfloat          delta_z;        /* z magnitude */
```

```
} Pvec3;
```

Pconorm3 is defined as:

```
typedef struct {
        Pcoval          colr;           /* colour */
        Pvec3           norm;           /* unit normal */
} Pconorm3;
```

Pcoval and Pvec3 are defined above.

vdata A pointer, to a Pfacet_vdata_arr3 structure, that specifies the coordinates of each triangle and optionally associated colour information. Pfacet_vdata_arr3 is defined as:

```
typedef union {
        Ppoint3         *points;        /* array (nv) of points */
        Pptco3          *ptcolrs;       /* array (nv) of points and
                                           colours */
        Pptnorm3        *ptnorms;       /* array (nv) of points and
                                           normals */
        Pptconorm3      *ptconorms;     /* array (nv) of points, colours,
                                           and normals */
        /* implementation dependent types can go here */
} Pfacet_vdata_arr3;
```

Ppoint3 is defined as:

```
typedef struct {
        Pfloat          x;              /* x coordinate */
        Pfloat          y;              /* y coordinate */
        Pfloat          z;              /* z coordinate */
} Ppoint3;
```

Pptco3 is defined as:

```
typedef struct {
        Ppoint3         point;          /* point coordinates */
        Pcoval          colr;           /* colour */
} Pptco3;
```

Ppoint3 is defined above. Pcoval is defined as:

```
typedef union {
        Pint            ind;            /* index in workstation colour
                                           bundle table */
        Pcolr_rep       direct;         /* direct colour components */
} Pcoval;
```

Pcolr_rep is defined as:

```
typedef union {
        Prgb        rgb;        /* Red Green Blue colour specification */
        Pcieluv     cieluv;     /* CIE L*U*V* colour specification */
        Phls        hls;        /* Hue Lightness Saturation colour
                                   specification */
        Phsv        hsv;        /* Hue Saturation Value colour
                                   specification */
        Pdata       unsupp;     /* Colour in unsupported colour model */
} Pcolr_rep
```

Prgb is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          red;            /* red, hue, and so on */
        Pfloat          green;          /* green, saturation, lightness,
                                           and so on */
        Pfloat          blue;           /* blue, value, saturation, and
                                           so on */
} Prgb;
```

Pcieluv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          cieluv_x;       /* x coefficient */
        Pfloat          cieluv_y;       /* y coefficient */
        Pfloat          cieluv_y_lum;   /* y luminance */
} Pcieluv;
```

Phls is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          lightness;      /* lightness */
        Pfloat          satur;          /* saturation */
} Phls;
```

Phsv is defined in phigs.h as follows:

```
typedef struct {
        Pfloat          hue;            /* hue */
        Pfloat          satur;          /* saturation */
        Pfloat          value;          /* value */
} Phsv;
```

Pdata is defined in phigs.h as follows:

```
typedef struct {
        size_t          size;           /* size of data */
```

```
        char        *data       /* pointer to data */
} Pdata;
```

Pptnorm3 is defined as:

```
typedef struct {
        Ppoint3     point;      /* point coordinates */
        Pvec3       norm;       /* unit normal */
} Pptnorm3;
```

Ppoint3 is defined above. Pvec3 is defined as:

```
typedef struct {
        Pfloat      delta_x;    /* x magnitude */
        Pfloat      delta_y;    /* y magnitude */
        Pfloat      delta_z;    /* z magnitude */
} Pvec3;
```

Pptconorm3 is defined as:

```
typedef struct {
        Ppoint3     point;      /* point coordinates */
        Pcoval      colr;       /* colour */
        Pvec3       norm;       /* unit normal */
        /* point coordinates with colour and unit normal */
} Pptconorm3;
```

Ppoint3, Pcoval, and Pvec3 are defined above.

FORTRAN Input Parameters

All of the following data types are predefined in phigs77.h.

FFLAG The data per facet flag indicates the information specified for each facet of the triangle strip. The possible values are:

0	PFNONE	*No Facet Data Specified*
1	PFCLR	*Facet Colour Specified*
2	PFNORM	*Facet Normal Specified*
3	PFCLRN	*Facet Colour and Normal Specified*

VFLAG The data per vertex flag indicates the information specified with each vertex of the primitive. The possible values are:

0	PVCOOR	*Coordinates Only*
1	PVCLR	*Coordinates and Colours*
2	PVNORM	*Coordinates and Normals*
3	PVCLRN	*Coordinates, Colours, and Normals*

COLTYP

The colour type. If the data per facet flag (FFLAG) indicates that a facet colour is specified, or if the data per vertex flag (VFLAG) indicates that vertex colours are specified, then COLTYP is set to one of the following:

0	PINDIR	*Indirect*
1	PRGB	*Red, Green, and Blue*
2	PCIE	*CIE Colour Model*
3	PHSV	*Hue, Saturation, and Value*
4	PHLS	*Hue, Lightness, and Saturation*

NV The number of vertices used to define the triangle strip. The number of triangles will always be 2 less than the number of vertices (NV-2).

IFCOLR(NV-2)
An array of integers containing indices into the workstation colour table for each facet or triangle. This array is unused if FFLAG is set to either PFNONE or PFNORM or if COLTYP is greater than PINDIR.

FCOLR(3,NV-2)
An array of reals containing the colours for each facet. This array is unused if FFLAG is set to either PFNONE or PFNORM or COLTYP is equal to PINDIR.

FCOLR(1,f) = *Red, Hue, and so on*
FCOLR(2,f) = *Green, Saturation, Lightness, and so on*
FCOLR(3,f) = *Blue, Value, Saturation, and so on*

FNORM(3,NV-2)
An array of reals containing the unit normals for each facet. This array is unused if FFLAG is set to either PFNONE or PFCLR.

FNORM(1,f) = *x Magnitude*
FNORM(2,f) = *y Magnitude*
FNORM(3,f) = *z Magnitude*

COORDS(3,NV)
An array of reals containing the *x*, *y*, and *z* coordinates of each of the triangles in the triangle strip.

COORDS(1,v) = *x Coordinate*
COORDS(2,v) = *y Coordinate*
COORDS(3,v) = *z Coordinate*

IVCOLR(NV)
An array of integers containing indices into the workstation colour table for each vertex. This array is unused if VFLAG is set to either PVCOOR or PVNORM or if COLTYP is greater than PINDIR.

VCOLR(3,NV)
An array of reals containing the colours for each vertex. This array is unused if VFLAG is set to either PVCOOR or PVNORM or if COLTYP is equal to PINDIR.

VCOLR(1,v) = *Red, Hue, and so on*
VCOLR(2,v) = *Green, Saturation, Lightness, and so on*
VCOLR(3,v) = *Blue, Value, Saturation, and so on*

VNORM(3,NV)
An array of reals containing the unit normals for each vertex. This array is

unused if VFLAG is set to either PVCOOR or PVCLR.

VNORM(1,v) = *x Magnitude*
VNORM(2,v) = *y Magnitude*
VNORM(3,v) = *z Magnitude*

IMPL(*)
The implementation defined vertex data.

Execution

Depending on the edit mode, a TRIANGLE STRIP 3 WITH DATA element is inserted into the open structure after the element pointer, or replaces the element pointed to by the element pointer. In either case, the element pointer is updated to point to the newly created TRIANGLE STRIP 3 WITH DATA element.

When the structure is traversed, TRIANGLE STRIP 3 WITH DATA generates a strip of *n*–2 connected triangles from *n* vertices. Line segments form the boundaries of all triangles in the strip. The triangular areas within the strip can be empty, hollow, shaded, or filled with a color, pattern, or hatch style. A strip with less than three vertices can be created, but is ignored during traversal.

TRIANGLE STRIP 3 WITH DATA uses the same attributes as the FILL AREA SET 3 WITH DATA structure element.

The TRIANGLE STRIP 3 WITH DATA element can specify the facet colour and normal vector of each triangle, as well as colours and normal vectors for each vertex. These colours and normal vectors are used in conjunction with the current lighting and depth cueing attributes to colour and shade the primitive. Note that unexpected lighting effects will result if the supplied normals are not of unit length. INTRO (3PP) describes how to use the output display attribute values.

Attributes Applied

The attributes listed below are used to display the TRIANGLE STRIP 3 WITH DATA primitive when the structure is traversed. The Aspect Source Flags (ASFs) tell where to access the output display attributes. These attributes can come directly from the traversal state list, or they can be accessed indirectly, using the appropriate index in the traversal state list and the corresponding bundled representation in the workstation state list.

interior colour	interior colour index ASF
back interior colour	back interior colour ASF
interior style	interior style ASF
back interior style	back interior style ASF
interior style index	interior style index ASF
back interior style index	back interior style index ASF
interior shading method	interior shading method ASF
back interior shading method	back interior shading method ASF
interior reflectance characteristics	interior reflectance characteristics ASF
back interior reflectance characteristics	back interior reflectance characteristics ASF
reflectance properties	reflectance properties ASF
back reflectance properties	back reflectance properties ASF

interior index	
edge colour	edge colour index ASF
edge flag	edge flag ASF
edgetype	edgetype ASF
edgewidth scale factor	edgewidth scale factor ASF
edge index	
face distinguishing mode	
face culling mode	
depth cue index	
light source state	
name set	

ERRORS

005 Ignoring function, function requires state (PHOP, *, STOP, *)

900 Storage overflow has occurred in PHIGS

2004 FORTRAN-specific binding error, ignoring function, input parameter size out of range

SEE ALSO

INTRO (3PP)
FILL AREA SET 3 WITH DATA (3PP)

Index

B

C

D

E

F

Q

S

T

W