INTERNATIONAL STANDARD

ISO/IEC 9593-3

First edition
1990-04-15

AMENDMENT 1
1994-06-15

Information technology — Computer graphics — Programmer's Hierarchical Interactive Graphics System (PHIGS) language bindings —

Part 3:
Ada

AMENDMENT 1: Incorporation of PHIGS PLUS

Technologies de l'information — Infographie — Interfaces langage avec système graphique hiérarchisé interactif de programmation —

Partie 3: Ada

AMENDEMENT 1: Incorporation du PHIGS PLUS

Reference number
ISO/IEC 9593-3:1990/Amd.1:1994(E)

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC take part in this work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Amendment 1 to International Standard ISO/IEC 9593-3:1990 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology.*

Introduction

page vi: The following text should replace the text in the Introduction.

Part 1 of PHIGS, ISO/IEC 9592-1 : 1989, provides a set of functions for the display and modification of 2D or 3D graphical data. Part 1 is extended by Part 4 (PHIGS PLUS) to incorporate the effects of lighting, shading, and other properties that are important for the display of surfaces and multidimensional data.

ISO/IEC 9592-1 and ISO/IEC 9592-4 are specificd in a language independent manner and must be embedded in language dependent layers (language bindings) for use with particular programming languages.

The purpose of this document is to define a standard binding of ISO/IEC 9592-4 to the Ada computer programming language.

Information technology — Computer graphics — Programmer's Hierarchical Interactive Graphics System (PHIGS) language bindings —

Part 3:
Ada

AMENDMENT 1: Incorporation of PHIGS PLUS

page 1: The following phrase should be inserted on a line following the word "Ada" in the title.

to include PHIGS Part 4 (PHIGS PLUS)

1 Scope

page 1: The following text should replace the text in clause 1 of ISO/IEC 9593-3:

ISO/IEC 9592-1 and ISO/IEC 9592-4 specify a language independent nucleus of a graphics system. For integration into a programming language, PHIGS and PHIGS PLUS are embedded in a language dependent layer obeying the particular conventions of that language. This part of ISO/IEC 9593 specifies such a language dependent layer for the Ada computer programming language.

2 Normative references

page 2: The following reference should be added:

ISO/IEC 9592-4 : 1992, *Information processing systems - Computer graphics - Programmer's Hierarchical Interactive Graphics System (PHIGS) — Part 4 - Plus Lumière und Surfaces (PHIGS PLUS).*

3 Principles

page 3: No changes.

3.1 Conformance

page 3: The following should be added to the list:

- To conform with PHIGS, the implementation shall correctly implement the binding defined in clauses 4 and 5; to conform with PHIGS PLUS, the implementation shall correctly implement the binding defined in clauses 4, 5, 6, and 7.

- A PHIGS Ada application should run without modification under a PHIGS PLUS Ada binding implementation.

3.2 Implications of the Language

page 3: No changes.

3.2.1 Functional Mapping

pages 3 and 4: No changes.

3.2.2 Implementations and Host Dependencies

page 4: No changes.

3.2.3 Error Handling

page 4: No changes.

3.2.4 Data Mapping

pages 4 to 6: No changes.

3.2.5 Multi-tasking

page 6: No changes.

3.2.6 Packaging

page 6 and 7: No changes.

3.2.7 Application Program Environment

page 7: No changes.

3.2.8 Registration

page 7: No changes.

4 Tables

page 8: No changes.

4.1 Abbreviations used in procedure names

page 8: No changes.

4.1.1 List of procedures using the abbreviations

pages 8 to 10: The following should be added to the list of procedures using the abbreviation INQ:

INQ	INQ_COLOUR_MAPPING_FACILITIES
	INQ_COLOUR_MAPPING_METHOD_FACILITIES
	INQ_COLOUR_MAPPING_REPRESENTATION
	INQ_COLOUR_MAPPING_STATE
	INQ_CURVE_AND_SURFACE_FACILITIES
	INQ_DATA_MAPPING_FACILITIES
	INQ_DATA_MAPPING_REPRESENTATION
	INQ_DEPTH_CUE_FACILITIES
	INQ_DEPTH_CUE_REPRESENTATION
	INQ_DIRECT_COLOUR_MODEL_FACILITIES
	INQ_DYNAMICS_OF_WS_ATTRIBUTES (PLUS)
	INQ_EDGE_REPRESENTATION (PLUS)
	INQ_INTERIOR_FACILITIES (PLUS)
	INQ_INTERIOR_REPRESENTATION (PLUS)
	INQ_LIGHT_SOURCE_FACILITIES
	INQ_LIGHT_SOURCE_REPRESENTATION
	INQ_LIST_OF_COLOUR_MAPPING_INDICES
	INQ_LIST_OF_DATA_MAPPING_INDICES
	INQ_LIST_OF_DEPTH_CUE_INDICES
	INQ_LIST_OF_LIGHT_SOURCE_INDICES
	INQ_LIST_OF_PARAMETRIC_SURFACE_INDICES
	INQ_LIST_OF_REFLECTANCE_INDICES
	INQ_PARAMETRIC_SURFACE_REPRESENTATION
	INQ_PATTERN_REPRESENTATION (PLUS)
	INQ_POLYLINE_FACILITIES (PLUS)
	INQ_POLYLINE_REPRESENTATION (PLUS)
	INQ_POLYMARKER_REPRESENTATION (PLUS)
	INQ_PREDEFINED_COLOUR_MAPPING_REPRESENTATION
	INQ_PREDEFINED_DATA_MAPPING_REPRESENTATION
	INQ_PREDEFINED_DEPTH_CUE_REPRESENTATION
	INQ_PREDEFINED_EDGE_REPRESENTATION (PLUS)
	INQ_PREDEFINED_INTERIOR_REPRESENTATION (PLUS)
	INQ_PREDEFINED_LIGHT_SOURCE_REPRESENTATION
	INQ_PREDEFINED_PARAMETRIC_SURFACE_REPRESENTATION

INQ_PREDEFINED_PATTERN_REPRESENTATION (PLUS)
INQ_PREDEFINED_POLYLINE_REPRESENTATION (PLUS)
INQ_PREDEFINED_POLYMARKER_REPRESENTATION (PLUS)
INQ_PREDEFINED_REFLECTANCE_REPRESENTATION
INQ_PREDEFINED_TEXT_REPRESENTATION (PLUS)
INQ_REFLECTANCE_FACILITIES
INQ_REFLECTANCE_REPRESENTATION
INQ_RENDERING_COLOUR_MODEL_FACILITIES
INQ_TEXT_REPRESENTATION (PLUS)
INQ_WS_STATE_TABLE_LENGTHS (PLUS)

page 10: The following should be added to the list of procedures using the abbreviation WS:

WS	INQ_DYNAMICS_OF_WS_ATTRIBUTES (PLUS)
	INQ_WS_STATE_TABLE_LENGTHS (PLUS)

4.1.2 Alphabetical by bound name

pages 11 to 15: The following list of functions should be added alphabetically to the alphabetical list of bound names:

CELL_ARRAY	cell array 3 plus
FILL_AREA_SET	fill area set 3 with data
FILL_AREA_SET	fill area set with data
INQ_COLOUR_MAPPING_FACILITIES	inquire colour mapping facilities
INQ_COLOUR_MAPPING_METHOD_FACILITIES	inquire colour mapping method facilities
INQ_COLOUR_MAPPING_REPRESENTATION	inquire colour mapping representation
INQ_COLOUR_MAPPING_STATE	inquire colour mapping state
INQ_CURVE_AND_SURFACE_FACILITIES	inquire curve and surface facilities
INQ_DATA_MAPPING_FACILITIES	inquire data mapping facilities
INQ_DATA_MAPPING_REPRESENTATION	inquire data mapping representation
INQ_DEPTH_CUE_FACILITIES	inquire depth cue facilities
INQ_DEPTH_CUE_REPRESENTATION	inquire depth cue representation
INQ_DIRECT_COLOUR_MODEL_FACILITIES	inquire direct colour model facilities
INQ_DYNAMICS_OF_WS_ATTRIBUTES	inquire dynamics of workstation attributes plus
INQ_EDGE_REPRESENTATION	inquire edge representation plus
INQ_INTERIOR_FACILITIES	inquire interior facilities plus
INQ_INTERIOR_REPRESENTATION	inquire interior representation plus
INQ_LIGHT_SOURCE_FACILITIES	inquire light source facilities
INQ_LIGHT_SOURCE_REPRESENTATION	inquire light source representation
INQ_LIST_OF_COLOUR_MAPPING_INDICES	inquire list of colour mapping indices
INQ_LIST_OF_DATA_MAPPING_INDICES	inquire list of data mapping indices
INQ_LIST_OF_DEPTH_CUE_INDICES	inquire list of depth cue indices
INQ_LIST_OF_LIGHT_SOURCE_INDICES	inquire list of light source indices
INQ_LIST_OF_PARAMETRIC_SURFACE_INDICES	inquire list of parametric surface indices
INQ_LIST_OF_REFLECTANCE_INDICES	inquire list of reflectance indices
INQ_PARAMETRIC_SURFACE_REPRESENTATION	inquire parametric surface representation
INQ_PATTERN_REPRESENTATION	inquire pattern representation plus
INQ_POLYLINE_FACILITIES	inquire polyline facilities plus
INQ_POLYLINE_REPRESENTATION	inquire polyline representation plus
INQ_POLYMARKER_REPRESENTATION	inquire polymarker representation plus
INQ_PREDEFINED_COLOUR_MAPPING_REPRESENTATION	inquire predefined colour mapping representation
INQ_PREDEFINED_DATA_MAPPING_REPRESENTATION	inquire predefined data mapping representation
INQ_PREDEFINED_DEPTH_CUE_REPRESENTATION	inquire predefined depth cue representation
INQ_PREDEFINED_EDGE_REPRESENTATION	inquire predefined edge representation plus
INQ_PREDEFINED_INTERIOR_REPRESENTATION	inquire predefined interior representation plus
INQ_PREDEFINED_LIGHT_SOURCE_REPRESENTATION	inquire predefined light source representation
INQ_PREDEFINED_PARAMETRIC_SURFACE_REPRESENTATION	inquire predefined parametric surface representation
INQ_PREDEFINED_PATTERN_REPRESENTATION	inquire predefined pattern representation plus
INQ_PREDEFINED_POLYLINE_REPRESENTATION	inquire predefined polyline representation plus
INQ_PREDEFINED_POLYMARKER_REPRESENTATION	inquire predefined polymarker representation plus
INQ_PREDEFINED_REFLECTANCE_REPRESENTATION	inquire predefined reflectance representation

INQ_PREDEFINED_TEXT_REPRESENTATION	inquire predefined text representation plus
INQ_REFLECTANCE_FACILITIES	inquire reflectance facilities
INQ_REFLECTANCE_REPRESENTATION	inquire reflectance representation
INQ_RENDERING_COLOUR_MODEL_FACILITIES	inquire rendering colour model facilities
INQ_TEXT_REPRESENTATION	inquire text representation plus
INQ_WS_STATE_TABLE_LENGTHS	inquire workstation state table lengths plus
NON_UNIFORM_B_SPLINE_CURVE	non-uniform B-spline curve
NON_UNIFORM_B_SPLINE_CURVE	non-uniform B-spline curve with colour
NON_UNIFORM_B_SPLINE_SURFACE	non-uniform B-spline surface
NON_UNIFORM_B_SPLINE_SURFACE	non_uniform B-spline surface with data
POLYLINE_SET	polyline set 3 with colour
QUADRILATERAL_MESH	quadrilateral mesh 3 with data
QUADRILATERAL_MESH	quadrilateral mesh with data
SET_BACK_DATA_MAPPING_INDEX	set back data mapping index
SET_BACK_DATA_MAPPING_METHOD	set back data mapping method
SET_BACK_INTERIOR_COLOUR	set back interior colour
SET_BACK_INTERIOR_INDEX	set back interior index
SET_BACK_INTERIOR_SHADING_METHOD	set back interior shading method
SET_BACK_INTERIOR_STYLE	set back interior style
SET_BACK_INTERIOR_STYLE_INDEX	set back interior style index
SET_BACK_REFLECTANCE_INDEX	set back reflectance index
SET_BACK_REFLECTANCE_MODEL	set back reflectance model
SET_BACK_REFLECTANCE_PROPERTIES	set back reflectance properties
SET_COLOUR_MAPPING_INDEX	set colour mapping index
SET_COLOUR_MAPPING_REPRESENTATION	set colour mapping representation
SET_CURVE_APPROXIMATION_CRITERIA	set curve approximation criteria
SET_DATA_MAPPING_INDEX	set data mapping index
SET_DATA_MAPPING_METHOD	set data mapping method
SET_DATA_MAPPING_REPRESENTATION	set data mapping representation
SET_DEPTH_CUE_INDEX	set depth cue index
SET_DEPTH_CUE_REPRESENTATION	set depth cue representation
SET_EDGE_COLOUR	set edge colour
SET_EDGE_REPRESENTATION	set edge representation plus
SET_FACET_CULLING_MODE	set facet culling mode
SET_FACET_DISTINGUISHING_MODE	set facet distinguishing mode
SET_INTERIOR_COLOUR	set interior colour
SET_INTERIOR_REPRESENTATION	set interior representation plus
SET_INTERIOR_SHADING_METHOD	set interior shading method
SET_LIGHT_SOURCE_REPRESENTATION	set light source representation
SET_LIGHT_SOURCE_STATE	set light source state
SET_OF_FILL_AREA_SETS	set of fill area sets 3 with data
SET_OF_FILL_AREA_SETS	set of fill area sets with data
SET_PARAMETRIC_SURFACE_CHARACTERISTICS	set parametric surface characteristics
SET_PARAMETRIC_SURFACE_INDEX	set parametric surface index
SET_PARAMETRIC_SURFACE_REPRESENTATION	set parametric surface representation
SET_PATTERN_REPRESENTATION	set pattern representation plus
SET_POLYLINE_COLOUR	set polyline colour
SET_POLYLINE_REPRESENTATION	set polyline representation plus
SET_POLYLINE_SHADING_METHOD	set polyline shading method
SET_POLYMARKER_COLOUR	set polymarker colour
SET_POLYMARKER_REPRESENTATION	set polymarker representation plus
SET_REFLECTANCE_INDEX	set reflectance index
SET_REFLECTANCE_MODEL	set reflectance model
SET_REFLECTANCE_PROPERTIES	set reflectance properties
SET_REFLECTANCE_REPRESENTATION	set reflectance representation
SET_RENDERING_COLOUR_MODEL	set rendering colour model
SET_SURFACE_APPROXIMATION_CRITERIA	set surface approximation criteria
SET_TEXT_COLOUR	set text colour
SET_TEXT_REPRESENTATION	set text representation plus
TRIANGLE_SET	triangle set 3 with data
TRIANGLE_SET	triangle set with data
TRIANGLE_STRIP	triangle strip 3 with data
TRIANGLE_STRIP	triangle strip with data

4.1.3 Alphabetical PHIGS functions

page 15: No changes.

4.2 Data type definitions

page 15: No changes.

4.2.1 Abbreviations used in the data type definitions

page 16: No changes.

4.2.2 Alphabetical list of type definitions

pages 16 to 66: No changes.

4.2.3 Alphabetical list of private type definitions

pages 66 to 68: No changes.

4.2.4 List of constant declarations

pages 68 to 69: No changes.

4.2.5 PHIGS configuration values

pages 69 to 71: No changes.

4.3 Error Codes

page 72: No changes.

4.3.1 Precluded Error Codes

page 72: No changes.

page 72: The following text should be added after clause 4.3.1 as clause 4.3.2 of ISO/IEC 9593-3.

4.3.2 Binding Specific Error Codes

The following binding specific error has been defined for use with this binding:

2502 Ignoring function, the parameters have inconsistent dimensions.

5 Functions in the Ada Binding of PHIGS

page 73: No changes.

5.1 Control functions

pages 73 and 74: No changes.

5.2 Output primitive functions

pages 74 to 77: No changes.

5.3 Attribute specification functions

pages 78 to 84: No changes.

5.4 Transformation and clipping functions

pages 85 to 91: No changes.

5.5 Structure content functions

pages 91 to 94: No changes.

5.6 Structure manipulation functions

pages 94 and 95: No changes.

5.7 Structure display functions

page 95: No changes.

5.8 Structure archive functions

pages 95 to 97: No changes.

5.9 Input functions

pages 98 to 106: No changes.

5.10 Metafile functions

pages 106 and 107: No changes.

5.11 Inquiry functions

pages 107 to 132: No changes.

5.12 Error control functions

page 132: No changes.

5.13 Special interface functions

pages 133 and 134: No changes.

5.14 Additional functions

page 134: No changes.

5.14.1 Subprograms for manipulating input data records

pages 134 to 138: No changes.

5.14.2 PHIGS generic coordinate system package

pages 138 and 141: No changes.

5.14.3 PHIGS generic list utility package

pages 141 to 143: No changes.

5.14.4 PHIGS name set facility package

pages 144 to 147: No changes.

5.14.5 Deallocation of structure element records

pages 147 to 149: No changes.

5.14.6 Metafile function utilities

page 149: No changes.

5.15 Conformal variants

page 150: No changes.

page 150: The following text should be added after clause 5 as clauses 6 and 7 of ISO/IEC 9593-3:

6 Tables for PHIGS PLUS

6.1 Data type definitions

This clause contains modifications to type definitions used for PHIGS as well as new type definitions used to support PHIGS PLUS.

6.1.1 Abbreviations used in the data type definitions

page 16: The following should be added to the list of abbreviations used in the data type definitions in clause 4.2.1:

APPROX approximation

6.1.2 Replacement definition for type ASPECT

The following definition should replace the type ASPECT in 4.2.2:

ASPECT

```
type ASPECT is ( TYPE_OF_LINE,
                 LINEWIDTH_SF,
                 LINE_COLOUR,
```

```
    TYPE_OF_MARKER,
    SIZE,
    MARKER_COLOUR,

    FONT,
    PRECISION,
    EXPANSION,
    SPACING,
    TEXT_COLOUR,

    STYLE_OF_INTERIOR,
    STYLE_IND,
    INTERIOR_COLOUR,

    FLAG,
    TYPE_OF_EDGE,
    EDGEWIDTH_SF,
    EDGE_COLOUR,

    GENERAL_POLYLINE_COLOUR,
    GENERAL_POLYMARKER_COLOUR,
    GENERAL_TEXT_COLOUR,
    GENERAL_INTERIOR_COLOUR,
    GENERAL_EDGE_COLOUR,

    METHOD_OF_POLYLINE_SHADING,
    METHOD_OF_INTERIOR_SHADING,
    METHOD_OF_DATA_MAPPING,
    REFLECTANCE_PROPERTIES,
    MODEL_OF_REFLECTANCE,

    BACK_STYLE_OF_INTERIOR,
    BACK_STYLE_IND,
    BACK_INTERIOR_COLOUR_DIR,
    BACK_METHOD_OF_INTERIOR_SHADING,
    BACK_METHOD_OF_DATA_MAPPING,
    BACK_REFLECTANCE_PROPERTIES,
    BACK_REFLECTANCE_MODEL,

    CRITERIA_FOR_CURVE_APPROX,
    CRITERIA_FOR_SURFACE_APPROX,
    CHARACTERISTICS_OF_PARAMETRIC_SURFACE);
```

-- This type lists the aspects for which an aspect source flag exists in PHIGS PLUS.

6.1.3 Replacement definition for type ATTRIBUTES_USED_TYPE

The following definition should replace the type ATTRIBUTES_USED_TYPE in 4.2.2:

ATTRIBUTES_USED_TYPE

```
type ATTRIBUTES_USED_TYPE is ( POLYLINE_ATTRIBUTES,
                               POLYMARKER_ATTRIBUTES,
                               TEXT_ATTRIBUTES,
                               INTERIOR_ATTRIBUTES,
                               EDGE_ATTRIBUTES,
                               REFLECTANCE_ATTRIBUTES,
                               PARAMETRIC_SURFACE_ATTRIBUTES);

-- The types of attributes which may be used in generating output for a GDP and in generating
-- prompt and echo information for certain prompt and echo types of certain classes of input
-- devices.
```

6.1.4 Replacement definition for type ELEMENT_TYPE

The following definition should replace the type ELEMENT_TYPE in 4.2.2:

ELEMENT_TYPE

```
type ELEMENT_TYPE is
        (ALL_ELEMENT_TYPES,
         NIL,

         POLYLINE_3,
         POLYLINE,

         POLYMARKER_3,
         POLYMARKER,

         TEXT_3,
         TEXT,

         ANNOTATION_TEXT_RELATIVE_3,
         ANNOTATION_TEXT_RELATIVE,
```

FILL_AREA_3,
FILL_AREA,
FILL_AREA_SET_3,
FILL_AREA_SET,

CELL_ARRAY_3,
CELL_ARRAY,

GDP_3,
GDP,

SET_POLYLINE_INDEX,
SET_POLYMARKER_INDEX,
SET_TEXT_INDEX,
SET_INTERIOR_INDEX,
SET_EDGE_INDEX,

SET_LINETYPE,
SET_LINEWIDTH_SCALE_FACTOR,
SET_POLYLINE_COLOUR_INDEX,

SET_MARKER_TYPE,
SET_MARKER_SIZE_SCALE_FACTOR,
SET_POLYMARKER_COLOUR_INDEX,

SET_TEXT_FONT,
SET_TEXT_PRECISION,
SET_CHAR_EXPANSION_FACTOR,
SET_CHAR_SPACING,
SET_TEXT_COLOUR_INDEX,
SET_CHAR_HEIGHT,
SET_CHAR_UP_VECTOR,
SET_TEXT_PATH,
SET_TEXT_ALIGNMENT,

SET_ANNOTATION_TEXT_CHAR_HEIGHT,
SET_ANNOTATION_TEXT_CHAR_UP_VECTOR,
SET_ANNOTATION_TEXT_PATH,
SET_ANNOTATION_TEXT_ALIGNMENT,
SET_ANNOTATION_STYLE,

SET_INTERIOR_STYLE,
SET_INTERIOR_STYLE_INDEX,
SET_INTERIOR_COLOUR_INDEX,

SET_EDGE_FLAG,
SET_EDGETYPE,
SET_EDGEWIDTH_SCALE_FACTOR,
SET_EDGE_COLOUR_INDEX,

SET_PATTERN_SIZE,
SET_PATTERN_REFERENCE_POINT_AND_VECTORS,
SET_PATTERN_REFERENCE_POINT,

ADD_NAMES_TO_SET,
REMOVE_NAMES_FROM_SET,

SET_INDIVIDUAL_ASF,
SET_HLHSR_IDENTIFIER,

SET_LOCAL_TRANSFORMATION_3,
SET_LOCAL_TRANSFORMATION,
SET_GLOBAL_TRANSFORMATION_3,
SET_GLOBAL_TRANSFORMATION,
SET_MODELLING_CLIPPING_VOLUME_3,
SET_MODELLING_CLIPPING_VOLUME,
SET_MODELLING_CLIPPING_INDICATOR,
RESTORE_MODELLING_CLIPPING_VOLUME,
SET_VIEW_INDEX,

EXECUTE_STRUCTURE,

LABEL,
APPLICATION_DATA,
GSE,
SET_PICK_IDENTIFIER,

POLYLINE_SET_3_WITH_COLOUR,
FILL_AREA_SET_3_WITH_DATA,
FILL_AREA_SET_WITH_DATA,
CELL_ARRAY_3_PLUS,
SET_OF_FILL_AREA_SETS_3_WITH_DATA,
SET_OF_FILL_AREA_SETS_WITH_DATA,
TRIANGLE_SET_3_WITH_DATA,
TRIANGLE_SET_WITH_DATA,
TRIANGLE_STRIP_3_WITH_DATA,
TRIANGLE_STRIP_WITH_DATA,
QUADRILATERAL_MESH_3_WITH_DATA,
QUADRILATERAL_MESH_WITH_DATA,
NON_UNIFORM_B_SPLINE_CURVE,
NON_UNIFORM_B_SPLINE_CURVE_WITH_COLOUR,
NON_UNIFORM_B_SPLINE_SURFACE,
NON_UNIFORM_B_SPLINE_SURFACE_WITH_DATA,

SET_DATA_MAPPING_INDEX,
SET_REFLECTANCE_INDEX,

SET_BACK_INTERIOR_INDEX,
SET_BACK_DATA_MAPPING_INDEX,
SET_BACK_REFLECTANCE_INDEX,

SET_PARAMETRIC_SURFACE_INDEX,

SET_POLYLINE_COLOUR,
SET_POLYLINE_SHADING_METHOD,

SET_POLYMARKER_COLOUR,

SET_TEXT_COLOUR,

SET_FACET_DISTINGUISHING_MODE,
SET_FACET_CULLING_MODE,

SET_INTERIOR_COLOUR,
SET_INTERIOR_SHADING_METHOD,

SET_DATA_MAPPING_METHOD,
SET_REFLECTANCE_PROPERTIES,
SET_REFLECTANCE_MODEL,

SET_BACK_INTERIOR_STYLE,
SET_BACK_INTERIOR_STYLE_INDEX,
SET_BACK_INTERIOR_COLOUR,
SET_BACK_INTERIOR_SHADING_METHOD,
SET_BACK_DATA_MAPPING_METHOD,
SET_BACK_REFLECTANCE_PROPERTIES,
SET_BACK_REFLECTANCE_MODEL,

SET_LIGHT_SOURCE_STATE,

SET_EDGE_COLOUR,

SET_CURVE_APPROX_CRITERIA,
SET_SURFACE_APPROX_CRITERIA,
SET_PARAMETRIC_SURFACE_CHARACTERISTICS,

```
        SET_RENDERING_COLOUR_MODEL,
        SET_DEPTH_CUE_INDEX,
        SET_COLOUR_MAPPING_INDEX);

    -- This type lists the element types which exist in PHIGS PLUS.
```

6.1.5 Replacement definition for type STRUCTURE_ELEMENT_RECORD

The following definition should replace the type STRUCTURE_ELEMENT_RECORD in 4.2.2:

STRUCTURE_ELEMENT_RECORD

```
    type STRUCTURE_ELEMENT_RECORD
      (ELEMENT_TYPE : STRUCTURE_ELEMENT_TYPE := NIL) is
        record
          case ELEMENT_TYPE is

            -- The empty element

            when NIL =>
              null;

            -- PHIGS Primitive Elements

            when POLYLINE_3 =>
              POLYLINE_3_POINTS : MC.ACCESS_POINT_LIST_3;

            when POLYLINE =>
              POLYLINE_POINTS : MC.ACCESS_POINT_LIST_2;

            when POLYMARKER_3 =>
              POLYMARKER_3_POINTS : MC.ACCESS_POINT_LIST_3;

            when POLYMARKER =>
              POLYMARKER_POINTS : MC.ACCESS_POINT_LIST_2;

            when TEXT_3 =>
              TEXT_3_POINT : MC.POINT_3;
              TEXT_DIRECTION_VECTORS : MC.VECTOR_PAIR_3;
              TEXT_3_CHAR_STRING : ACCESS_STRING;

            when TEXT =>
              TEXT_POINT : MC.POINT_2;
              TEXT_CHAR_STRING : ACCESS_STRING;
```

```
when ANNOTATION_TEXT_RELATIVE_3 =>
  ANNOTATION_TEXT_RELATIVE_3_REF_POINT : MC.POINT_3;
  ANNOTATION_TEXT_RELATIVE_3_OFFSET : NPC.POINT_3;
  ANNOTATION_TEXT_3_CHAR_STRING : ACCESS_STRING;

when ANNOTATION_TEXT_RELATIVE =>
  ANNOTATION_TEXT_RELATIVE_REF_POINT : MC.POINT_2;
  ANNOTATION_TEXT_RELATIVE_OFFSET : NPC.POINT_2;
  ANNOTATION_TEXT_CHAR_STRING : ACCESS_STRING;

when FILL_AREA_3 =>
  FILL_AREA_3_POINTS : MC.ACCESS_POINT_LIST_3;

when FILL_AREA =>
  FILL_AREA_POINTS : MC.ACCESS_POINT_LIST_2;

when FILL_AREA_SET_3 =>
  FILL_AREA_SET_3_POINTS : MC.ACCESS_LIST_OF_POINT_LIST_3;

when FILL_AREA_SET =>
  FILL_AREA_SET_POINTS : MC.ACCESS_LIST_OF_POINT_LIST_2;

when CELL_ARRAY_3 =>
  CORNER_P_3 : MC.POINT_3;
  CORNER_Q_3 : MC.POINT_3;
  CORNER_R_3 : MC.POINT_3;
  CELL_ARRAY_3_CELLS : ACCESS_COLOUR_MATRIX;

when CELL_ARRAY =>
  CORNER_P : MC.POINT_2;
  CORNER_Q : MC.POINT_2;
  CELL_ARRAY_CELLS : ACCESS_COLOUR_MATRIX;

when GDP_3 =>
  GDP_3_POINTS : MC.ACCESS_POINT_LIST_3;
  GDP_3_DATA : GDP_3_RECORD;

when GDP =>
  GDP_POINTS : MC.ACCESS_POINT_LIST_2;
  GDP_DATA : GDP_RECORD;

-- PHIGS Bundle Index Elements

when SET_POLYLINE_INDEX =>
  POLYLINE_IND : POLYLINE_INDEX;

when SET_POLYMARKER_INDEX =>
  POLYMARKER_IND : POLYMARKER_INDEX;
```

```
when SET_TEXT_INDEX =>
  TEXT_IND : TEXT_INDEX;

when SET_INTERIOR_INDEX =>
  INTERIOR_IND : INTERIOR_INDEX;

when SET_EDGE_INDEX =>
  EDGE_IND : EDGE_INDEX;

-- PHIGS Individual Aspect Elements

when SET_LINETYPE =>
  TYPE_OF_LINE : LINETYPE;

when SET_LINEWIDTH_SCALE_FACTOR =>
  LINEWIDTH_SF : LINEWIDTH;

when SET_POLYLINE_COLOUR_INDEX =>
  LINE_COLOUR : COLOUR_INDEX;

when SET_MARKER_TYPE =>
  TYPE_OF_MARKER : MARKER_TYPE;

when SET_MARKER_SIZE_SCALE_FACTOR =>
  SIZE : MARKER_SIZE;

when SET_POLYMARKER_COLOUR_INDEX =>
  MARKER_COLOUR : COLOUR_INDEX;

when SET_TEXT_FONT =>
  FONT : TEXT_FONT;

when SET_TEXT_PRECISION =>
  PRECISION : TEXT_PRECISION;

when SET_CHAR_EXPANSION_FACTOR =>
  EXPANSION : CHAR_EXPANSION;

when SET_CHAR_SPACING =>
  SPACING : CHAR_SPACING;

when SET_TEXT_COLOUR_INDEX =>
  TEXT_COLOUR : COLOUR_INDEX;

when SET_CHAR_HEIGHT =>
  HEIGHT : MC.MAGNITUDE;

when SET_CHAR_UP_VECTOR =>
  CHAR_UP_VECTOR : MC.VECTOR_2;
```

```
when SET_TEXT_PATH =>
  PATH : TEXT_PATH;

when SET_TEXT_ALIGNMENT =>
  ALIGNMENT : TEXT_ALIGNMENT;

when SET_ANNOTATION_TEXT_CHAR_HEIGHT =>
  ANNOTATION_HEIGHT : NPC.MAGNITUDE;

when SET_ANNOTATION_TEXT_CHAR_UP_VECTOR =>
  ANNOTATION_CHAR_UP_VECTOR : NPC.VECTOR_2;

when SET_ANNOTATION_TEXT_PATH =>
  ANNOTATION_PATH : TEXT_PATH;

when SET_ANNOTATION_TEXT_ALIGNMENT =>
  ANNOTATION_ALIGNMENT : TEXT_ALIGNMENT;

when SET_ANNOTATION_STYLE =>
  STYLE_OF_ANNOTATION : ANNOTATION_STYLE;

when SET_INTERIOR_STYLE =>
  STYLE_OF_INTERIOR : INTERIOR_STYLE;

when SET_INTERIOR_STYLE_INDEX =>
  STYLE_IND : STYLE_INDEX;

when SET_INTERIOR_COLOUR_INDEX =>
  INTERIOR_COLOUR : COLOUR_INDEX;

when SET_EDGE_FLAG =>
  FLAG : EDGE_FLAG;

when SET_EDGETYPE =>
  TYPE_OF_EDGE : EDGETYPE;

when SET_EDGEWIDTH_SCALE_FACTOR =>
  EDGEWIDTH_SF : EDGEWIDTH;

when SET_EDGE_COLOUR_INDEX =>
  EDGE_COLOUR : COLOUR_INDEX;

-- PHIGS Pattern Attribute Elements

when SET_PATTERN_SIZE =>
  PATTERN_SIZE : MC.SIZE_2;

when SET_PATTERN_REFERENCE_POINT_AND_VECTORS =>
  PATTERN_REFERENCE_POINT_3 : MC.POINT_3;
  PATTERN_REFERENCE_VECTORS : MC.VECTOR_PAIR_3;
```

```
when SET_PATTERN_REFERENCE_POINT =>
  PATTERN_REFERENCE_POINT : MC.POINT_2;

-- PHIGS Name Set Elements

when ADD_NAMES_TO_SET =>
  NAMES_TO_ADD : NAME_SET;

when REMOVE_NAMES_FROM_SET =>
  NAMES_TO_REMOVE : NAME_SET;

-- PHIGS ASF Elements

when SET_INDIVIDUAL_ASF =>
  ATTRIBUTE_ID : ASPECT;
  SOURCE_FLAG : ASF;

-- PHIGS HLHSR Elements

when SET_HLHSR_IDENTIFIER =>
  HLHSR_IDENTIFIER : HLHSR_ID;

-- PHIGS Transformation Elements

when SET_LOCAL_TRANSFORMATION_3 =>
  LOCAL_MATRIX_3 : TRANSFORMATION_MATRIX_3;
  HOW_APPLIED_3 : COMPOSITION_TYPE;

when SET_LOCAL_TRANSFORMATION =>
  LOCAL_MATRIX : TRANSFORMATION_MATRIX_2;
  HOW_APPLIED : COMPOSITION_TYPE;

when SET_GLOBAL_TRANSFORMATION_3 =>
  GLOBAL_MATRIX_3 : TRANSFORMATION_MATRIX_3;

when SET_GLOBAL_TRANSFORMATION =>
  GLOBAL_MATRIX : TRANSFORMATION_MATRIX_2;

when SET_MODELLING_CLIPPING_VOLUME_3 =>
  MODELLING_CLIPPING_OPERATOR_3
                    : MODELLING_CLIP_OPERATION_TYPE;
  MODELLING_CLIPPING_LIMITS_3
                    : MC.ACCESS_HALF_SPACE_LIST_3;

when SET_MODELLING_CLIPPING_VOLUME =>
  MODELLING_CLIPPING_OPERATOR
                    : MODELLING_CLIP_OPERATION_TYPE;
  MODELLING_CLIPPING_LIMITS : MC.ACCESS_HALF_SPACE_LIST_2;
```

```
when SET_MODELLING_CLIPPING_INDICATOR =>
  MODELLING_CLIPPING_INDICATOR : CLIPPING_INDICATOR;

when RESTORE_MODELLING_CLIPPING_VOLUME =>
  null;

when SET_VIEW_INDEX =>
  VIEW_IND : VIEW_INDEX;

-- PHIGS Invocation Elements

when EXECUTE_STRUCTURE =>
  STRUCTURE_IDENTIFIER : STRUCTURE_ID;

-- PHIGS Structure Content Identification Elements

when LABEL =>
  LABEL_IDENTIFIER : LABEL_ID;

when APPLICATION_DATA =>
  DATA : APPLICATION_DATA_RECORD;

when GSE =>
  GSE_DATA : GSE_RECORD;

when SET_PICK_IDENTIFIER =>
  PICK_IDENTIFIER : PICK_ID;

-- PHIGS PLUS Primitive Elements

when POLYLINE_SET_3_WITH_COLOUR =>
  POLYLINE_SET_3_VERTICES
                    : MC.ACCESS_LIST_OF_VERTEX_COLOUR_LIST_SET_3;

when FILL_AREA_SET_3_WITH_DATA =>
  FILL_AREA_SET_3_FACET : MC.ACCESS_FACET_DATA_SET;
  FILL_AREA_SET_3_EDGES : ACCESS_LIST_OF_EDGE_FLAG_LIST;
  FILL_AREA_SET_3_VERTICES
                    : MC.ACCESS_LIST_OF_VERTEX_DATA_LIST_SET_3;

when FILL_AREA_SET_WITH_DATA =>
  FILL_AREA_SET_FACET : MC.ACCESS_FACET_DATA_SET;
  FILL_AREA_SET_EDGES : ACCESS_LIST_OF_EDGE_FLAG_LIST;
  FILL_AREA_SET_VERTICES
                    : MC.ACCESS_LIST_OF_VERTEX_DATA_LIST_SET_2;
```

```
when CELL_ARRAY_3_PLUS =>
  CORNER_P_3_PLUS : MC.POINT_3;
  CORNER_Q_3_PLUS : MC.POINT_3;
  CORNER_R_3_PLUS : MC.POINT_3;
  CELL_ARRAY_3_PLUS_CELLS : ACCESS_COLOUR_VALUE_ARRAY;

when SET_OF_FILL_AREA_SETS_3_WITH_DATA =>
  SET_OF_FILL_AREA_SETS_3_FACETS
                        : MC.ACCESS_FACET_DATA_LIST_SET;
  SET_OF_FILL_AREA_SETS_3_EDGES
                        : ACCESS_LIST_OF_LIST_OF_EDGE_FLAG_LIST;
  SET_OF_FILL_AREA_SETS_3_VERTICES
                        : MC.ACCESS_VERTEX_DATA_LIST_SET_3;
  SET_OF_FILL_AREA_SETS_3_INDICES
                        : ACCESS_LIST_OF_LIST_OF_VERTEX_INDEX_LIST;

when SET_OF_FILL_AREA_SETS_WITH_DATA =>
  SET_OF_FILL_AREA_SETS_FACETS : MC.ACCESS_FACET_DATA_LIST_SET;
  SET_OF_FILL_AREA_SETS_EDGES
                        : ACCESS_LIST_OF_LIST_OF_EDGE_FLAG_LIST;
  SET_OF_FILL_AREA_SETS_VERTICES
                        : MC.ACCESS_VERTEX_DATA_LIST_SET_2;
  SET_OF_FILL_AREA_SETS_INDICES
                        : ACCESS_LIST_OF_LIST_OF_VERTEX_INDEX_LIST;

when TRIANGLE_SET_3_WITH_DATA =>
  TRIANGLE_SET_3_FACETS : MC.ACCESS_FACET_DATA_LIST_SET;
  TRIANGLE_SET_3_EDGES : ACCESS_LIST_OF_EDGE_FLAG_TRIPLET;
  TRIANGLE_SET_3_VERTICES : MC.ACCESS_VERTEX_DATA_LIST_SET_3;
  TRIANGLE_SET_3_INDICES
                        : ACCESS_LIST_OF_VERTEX_INDEX_TRIPLET;

when TRIANGLE_SET_WITH_DATA =>
  TRIANGLE_SET_FACETS : MC.ACCESS_FACET_DATA_LIST_SET;
  TRIANGLE_SET_EDGES : ACCESS_LIST_OF_EDGE_FLAG_TRIPLET;
  TRIANGLE_SET_VERTICES : MC.ACCESS_VERTEX_DATA_LIST_SET_2;
  TRIANGLE_SET_INDICES
                        : ACCESS_LIST_OF_VERTEX_INDEX_TRIPLET;

when TRIANGLE_STRIP_3_WITH_DATA =>
  TRIANGLE_STRIP_3_FACETS : MC.ACCESS_FACET_DATA_LIST_SET;
  TRIANGLE_STRIP_3_EDGES : ACCESS_EDGE_FLAG_LIST;
  TRIANGLE_STRIP_3_VERTICES
                        : MC.ACCESS_VERTEX_DATA_LIST_SET_3;

when TRIANGLE_STRIP_WITH_DATA =>
  TRIANGLE_STRIP_FACETS : MC.ACCESS_FACET_DATA_LIST_SET;
  TRIANGLE_STRIP_EDGES : ACCESS_EDGE_FLAG_LIST;
  TRIANGLE_STRIP_VERTICES : MC.ACCESS_VERTEX_DATA_LIST_SET_2;
```

```
when QUADRILATERAL_MESH_3_WITH_DATA =>
  QUADRILATERAL_MESH_3_FACETS
                        : MC.ACCESS_FACET_DATA_ARRAY_SET;
  QUADRILATERAL_MESH_3_EDGES
                        : ACCESS_ARRAY_OF_EDGE_FLAG_PAIR;
  QUADRILATERAL_MESH_3_VERTICES
                        : MC.ACCESS_VERTEX_DATA_ARRAY_SET_3;

when QUADRILATERAL_MESH_WITH_DATA =>
  QUADRILATERAL_MESH_FACETS
                        : MC.ACCESS_FACET_DATA_ARRAY_SET;
  QUADRILATERAL_MESH_EDGES
                        : ACCESS_ARRAY_OF_EDGE_FLAG_PAIR;
  QUADRILATERAL_MESH_VERTICES
                        : MC.ACCESS_VERTEX_DATA_ARRAY_SET_2;

when NON_UNIFORM_B_SPLINE_CURVE =>
  NURB_CURVE_SPLINE : MC.ACCESS_CURVE_GEOMETRYSPLINE_3;
  NURB_CURVE_LIMITS : SPC.RANGE_OF_MAGNITUDES;

when NON_UNIFORM_B_SPLINE_CURVE_WITH_COLOUR =>
  NURB_CURVE_COLOUR_GEOMETRYSPLINE
                        : MC.ACCESS_CURVE_GEOMETRYSPLINE_3;
  NURB_CURVE_COLOUR_LIMITS : SPC.RANGE_OF_MAGNITUDES;
  NURB_CURVE_COLOUR_COLOURSPLINE
                        : ACCESS_CURVE_COLOURSPLINE;

when NON_UNIFORM_B_SPLINE_SURFACE =>
  NURB_SURFACE_GEOMETRYSPLINE
                        : MC.ACCESS_SURFACE_GEOMETRYSPLINE;
  NURB_SURFACE_TRIM_CURVES
                        : ACCESS_LIST_OF_TRIMCURVE_LIST;

when NON_UNIFORM_B_SPLINE_SURFACE_WITH_DATA =>
  NURB_SURFACE_DATA_GEOMETRYSPLINE
                        : MC.ACCESS_SURFACE_GEOMETRYSPLINE;
  NURB_SURFACE_DATA_TRIM_CURVES
                        : ACCESS_LIST_OF_TRIMCURVE_LIST;
  NURB_SURFACE_DATA_COLOURSPLINE
                        : ACCESS_SURFACE_COLOURSPLINE;
  NURB_SURFACE_DATA_DATASPLINES
                                : ACCESS_DATASPLINE_LIST;

-- PHIGS PLUS Bundle Index Elements

when SET_DATA_MAPPING_INDEX =>
  DATA_MAPPING_IND : DATA_MAPPING_INDEX;
```

```
when SET_REFLECTANCE_INDEX =>
  REFLECTANCE_IND : REFLECTANCE_INDEX;

when SET_BACK_INTERIOR_INDEX =>
  BACK_INTERIOR_IND : INTERIOR_INDEX;

when SET_BACK_DATA_MAPPING_INDEX =>
  BACK_DATA_MAPPING_IND : DATA_MAPPING_INDEX;

when SET_BACK_REFLECTANCE_INDEX =>
  BACK_REFLECTANCE_IND : REFLECTANCE_INDEX;

when SET_PARAMETRIC_SURFACE_INDEX =>
  PARAMETRIC_SURFACE_IND : PARAMETRIC_SURFACE_INDEX;

-- PHIGS PLUS Individual Aspect Elements

when SET_POLYLINE_COLOUR =>
  GENERAL_POLYLINE_COLOUR : GENERAL_COLOUR;

when SET_POLYLINE_SHADING_METHOD =>
  POLYLINE_SHADING : POLYLINE_SHADING_METHOD;

when SET_POLYMARKER_COLOUR =>
  GENERAL_POLYMARKER_COLOUR : GENERAL_COLOUR;

when SET_TEXT_COLOUR =>
  GENERAL_TEXT_COLOUR : GENERAL_COLOUR;

when SET_FACET_DISTINGUISHING_MODE =>
  FACET_DISTINGUISHING : FACET_DISTINGUISHING_MODE;

when SET_FACET_CULLING_MODE =>
  FACET_CULLING : FACET_CULLING_MODE;

when SET_INTERIOR_COLOUR =>
  GENERAL_INTERIOR_COLOUR : GENERAL_COLOUR;

when SET_INTERIOR_SHADING_METHOD =>
  INTERIOR_SHADING : INTERIOR_SHADING_METHOD;

when SET_DATA_MAPPING_METHOD =>
  METHOD_OF_DATA_MAPPING : DATA_MAPPING_DATA_RECORD;

when SET_REFLECTANCE_PROPERTIES =>
  REFLECTANCE_PROPERTIES : REFLECTANCE_PROPERTIES_DATA_RECORD;

when SET_REFLECTANCE_MODEL =>
  MODEL_OF_REFLECTANCE : REFLECTANCE_MODEL;
```

```
when SET_BACK_INTERIOR_STYLE =>
  BACK_STYLE_OF_INTERIOR : INTERIOR_STYLE;

when SET_BACK_INTERIOR_STYLE_INDEX =>
  BACK_STYLE_IND : STYLE_INDEX;

when SET_BACK_INTERIOR_COLOUR =>
  BACK_GENERAL_INTERIOR_COLOUR : GENERAL_COLOUR;

when SET_BACK_INTERIOR_SHADING_METHOD =>
  BACK_INTERIOR_SHADING : INTERIOR_SHADING_METHOD;

when SET_BACK_DATA_MAPPING_METHOD =>
  BACK_METHOD_OF_DATA_MAPPING : DATA_MAPPING_METHOD;

when SET_BACK_REFLECTANCE_PROPERTIES =>
  BACK_REFLECTANCE_PROPERTIES
                          : REFLECTANCE_PROPERTIES_DATA_RECORD;

when SET_BACK_REFLECTANCE_MODEL =>
  BACK_REFLECTANCE : REFLECTANCE_MODEL;

when SET_LIGHT_SOURCE_STATE =>
  ACTIVATION_LIST : LIGHT_SOURCE_INDICES.LIST_OF;
  DEACTIVATION_LIST : LIGHT_SOURCE_INDICES.LIST_OF;

when SET_EDGE_COLOUR =>
  GENERAL_EDGE_COLOUR : GENERAL_COLOUR;

when SET_CURVE_APPROX_CRITERIA =>
  CRITERIA_FOR_CURVE_APPROX : CURVE_APPROX_DATA_RECORD;

when SET_SURFACE_APPROX_CRITERIA =>
  CRITERIA_FOR_SURFACE_APPROX
                          : SURFACE_APPROX_DATA_RECORD;

when SET_PARAMETRIC_SURFACE_CHARACTERISTICS =>
  PARAMETRIC_SURFACE_CHARACTERISTICS
                          : PARAMETRIC_SURFACE_DATA_RECORD;

when SET_RENDERING_COLOUR_MODEL =>
  RENDERING_COLOUR_MODEL : COLOUR_MODEL;

when SET_DEPTH_CUE_INDEX =>
  DEPTH_CUE_IND : DEPTH_CUE_INDEX;

when SET_COLOUR_MAPPING_INDEX =>
  COLOUR_MAPPING_IND : COLOUR_MAPPING_INDEX;
```

```
        end case;
    end record;

    -- This type defines the format of structure contents for PHIGS PLUS structures and is used to
    -- return structure elements during inquiry.
```

6.1.6 Additions to alphabetical list of PHIGS type definitions

The following additional types should be added alphabetically to the alphabetical list of type definitions in 4.2.2:

ACCESS_ARRAY_OF_EDGE_FLAG_PAIR

```
    type ACCESS_ARRAY_OF_EDGE_FLAG_PAIR is access ARRAY_OF_EDGE_FLAG_PAIR;

    -- Provides for access variable to two-dimensional arrays of edge flag pairs.  These will be
    -- used when inquiring structure elements which are "Quadrilateral Mesh with Data"
    -- primitives.
```

ACCESS_COLOUR_MAPPING_DATA_RECORD

```
    type ACCESS_COLOUR_MAPPING_DATA_RECORD is
            access COLOUR_MAPPING_DATA_RECORD;

    -- Provides for pointers to colour mapping data records.
```

ACCESS_COLOUR_VALUE_ARRAY

```
    type ACCESS_COLOUR_VALUE_ARRAY is access COLOUR_VALUE_ARRAY;

    -- Provides for pointers to arrays of general colour values.
```

ACCESS_COLOUR_VALUE_LIST

```
    type ACCESS_COLOUR_VALUE_LIST is access COLOUR_VALUE_LIST;

    -- Provides for pointers to lists of general colour values.
```

ACCESS_CURVE_COLOURSPLINE

```
type ACCESS_CURVE_COLOURSPLINE is
        access CURVE_COLOURSPLINE;

-- Provides for pointers to colourspline definitions.
```

ACCESS_DATASPLINE_LIST

```
type ACCESS_DATASPLINE_LIST is
        access DATASPLINE_LIST;

-- Provides for pointers to lists of dataspline definitions.
```

ACCESS_DATA_VALUE_SET

```
type ACCESS_DATA_VALUE_SET is access DATA_VALUE_SET;

-- Provides for pointers to sets of data mapping data values.
```

ACCESS_EDGE_FLAG_LIST

```
type ACCESS_EDGE_FLAG_LIST is access EDGE_FLAG_LIST;

-- Provides for pointers to lists of edge flags.
```

ACCESS_LIST_OF_COLOUR_VALUE_LIST

```
type ACCESS_LIST_OF_COLOUR_VALUE_LIST is
        access LIST_OF_COLOUR_VALUE_LIST;

-- Provides for pointers to lists of lists of colour values values.
```

ACCESS_LIST_OF_EDGE_FLAG_LIST

```
type ACCESS_LIST_OF_EDGE_FLAG_LIST is access LIST_OF_EDGE_FLAG_LIST;

-- Provides for pointers to lists of lists of edge flags.
```

ACCESS_LIST_OF_EDGE_FLAG_TRIPLET

```
type ACCESS_LIST_OF_EDGE_FLAG_TRIPLET is
  access LIST_OF_EDGE_FLAG_TRIPLET;

-- Provides for pointers to lists of triplets of edge flags.
```

ACCESS_LIST_OF_LIST_OF_EDGE_FLAG_LIST

```
type ACCESS_LIST_OF_LIST_OF_EDGE_FLAG_LIST is
      access LIST_OF_LIST_OF_EDGE_FLAG_LIST;

-- Provides for pointers to lists of lists of edge flag lists.
```

ACCESS_LIST_OF_LIST_OF_VERTEX_INDEX_LIST

```
type ACCESS_LIST_OF_LIST_OF_VERTEX_INDEX_LIST is
      access LIST_OF_LIST_OF_VERTEX_INDEX_LIST;

-- Provides for pointers to lists of lists of lists of indices.
```

ACCESS_LIST_OF_TRIMCURVE_LIST

```
type ACCESS_LIST_OF_TRIMCURVE_LIST is access LIST_OF_TRIMCURVE_LIST;

-- Provides for pointers to lists of lists of edge flags.
```

ACCESS_LIST_OF_VERTEX_INDEX_LIST

```
type ACCESS_LIST_OF_VERTEX_INDEX_LIST is
  access LIST_OF_VERTEX_INDEX_LIST;

-- Provides for pointers to lists of lists of vertex indices.
```

ACCESS_LIST_OF_VERTEX_INDEX_TRIPLET

```
type ACCESS_LIST_OF_VERTEX_INDEX_TRIPLET is
  access LIST_OF_VERTEX_INDEX_TRIPLET;

-- Provides for pointers to lists of triplets of vertex indices.
```

ACCESS_SURFACE_COLOURSPLINE

```
type ACCESS_SURFACE_COLOURSPLINE is
        access SURFACE_COLOURSPLINE;

-- Provides for pointers to colourspline definitions.
```

ACCESS_TRIMCURVE_LIST

```
type ACCESS_TRIMCURVE_LIST is access TRIMCURVE_LIST;

-- Provides for pointers to lists of trimming curves.
```

ACCESS_WEIGHT_LIST

```
type ACCESS_WEIGHT_LIST is access WEIGHT_LIST;

-- Provides for pointers to lists of weights.
```

ACCESS_VERTEX_INDEX_LIST

```
type ACCESS_VERTEX_INDEX_LIST is access VERTEX_INDEX_LIST;

-- Provides for pointers to lists of vertex indices.
```

ARRAY_OF_DATA_VALUE_SET

```
type ARRAY_OF_DATA_VALUE_SET is
        array ( VERTEX_SET_DIMENSION range <>,
                VERTEX_SET_DIMENSION range <>,
                DATA_VALUE_INDEX range <>) of DATA_VALUE;

-- Provides for two-dimensional arrays of data mapping data value sets.  These
-- will be used with "Quadrilateral Mesh with Data" primitives.  The first index
-- refers to the width of the array, the second index to the height, and the third
-- index to the individual data values.
```

ARRAY_OF_EDGE_FLAG_PAIR

```
type ARRAY_OF_EDGE_FLAG_PAIR is
        array (POSITIVE range <>,
                POSITIVE range <>) of EDGE_FLAG_PAIR;

-- Provides for two-dimensional arrays of edge flag pairs.  These will be used with
-- "Quadrilateral Mesh with Data" primitives.
```

ATTENUATION_COEFFICIENTS

```
type ATTENUATION_COEFFICIENTS is
  record
    C1  : COLOUR_COEFFICIENT;
    C2  : COLOUR_COEFFICIENT;
  end record;

-- Provides for specification of light source attenuation.
```

CIELUV_COLOUR_VALUE

```
type CIELUV_COLOUR_VALUE is
  record
    L_STAR_CIE : COLOUR_COEFFICIENT;
    U_STAR_CIE : COLOUR_COEFFICIENT;
    V_STAR_CIE : COLOUR_COEFFICIENT;
  end record;

-- Provides for specification of CIELUV colours.
```

CIELUV_HOMOGENEOUS_COLOUR_VALUE

```
type CIELUV_HOMOGENEOUS_COLOUR_VALUE is
  record
    L_STAR_CIE : COLOUR_COEFFICIENT;
    U_STAR_CIE : COLOUR_COEFFICIENT;
    V_STAR_CIE : COLOUR_COEFFICIENT;
    W_CIE      : COLOUR_COEFFICIENT;
  end record;

-- Provides for specification of homogeneous CIELUV colours for use in
-- rational coloursplines.
```

CIELUV_TYPES

```
package CIELUV_TYPES is
  new PHIGS_COLOUR_TYPES( CIELUV_COLOUR_VALUE,
                          CIELUV_HOMOGENEOUS_COLOUR_VALUE);

-- Provides for CIELUV colour types.
```

COLOUR_CONTROL_POINT_ARRAY

```
type COLOUR_CONTROL_POINT_ARRAY
  (MODEL          : COLOUR_MODEL := RGB;
   RATIONALITY    : SPLINE_RATIONALITY := RATIONAL;
   LENGTH_U       : COLOUR_VALUE_SET_DIMENSION := 1;
   LENGTH_V       : COLOUR_VALUE_SET_DIMENSION := 1)  is
    record
      case MODEL is

        when INDIRECT =>
          COLOURS_INDIRECT : INDIRECT_TYPES.CONTROL_POINT_ARRAY
                                (RATIONALITY, LENGTH_U, LENGTH_V);

        when RGB =>
          COLOURS_RGB : RGB_TYPES.CONTROL_POINT_ARRAY
                          (RATIONALITY, LENGTH_U, LENGTH_V);

        when CIELUV =>
          COLOURS_CIELUV : CIELUV_TYPES.CONTROL_POINT_ARRAY
                             (RATIONALITY, LENGTH_U, LENGTH_V);

        when HSV =>
          COLOURS_HSV  : HSV_TYPES.CONTROL_POINT_ARRAY
                           (RATIONALITY, LENGTH_U, LENGTH_V);

        when HLS =>
          COLOURS_HLS : HLS_TYPES.CONTROL_POINT_ARRAY
                          (RATIONALITY, LENGTH_U, LENGTH_V);

        when others =>
          COLOURS_GENERIC : GENERIC_COLOUR_TYPES.CONTROL_POINT_ARRAY
                              (RATIONALITY, LENGTH_U, LENGTH_V);
```

```
        end case;
      end record;

  -- Provides for specification of two-dimensional arrays of colour control point values
  -- all of which are the same colour model.
```

COLOUR_CONTROL_POINT_LIST

```
  type COLOUR_CONTROL_POINT_LIST
    (MODEL          : COLOUR_MODEL := RGB;
     RATIONALITY    : SPLINE_RATIONALITY := RATIONAL;
     LENGTH : COLOUR_VALUE_SET_DIMENSION := 0) is
      record
        case MODEL is

          when INDIRECT =>
            COLOURS_INDIRECT : INDIRECT_TYPES.CONTROL_POINT_LIST
                                    (RATIONALITY, LENGTH);

          when RGB =>
            COLOURS_RGB : RGB_TYPES.CONTROL_POINT_LIST
                                    (RATIONALITY, LENGTH);

          when CIELUV =>
            COLOURS_CIELUV  : CIELUV_TYPES.CONTROL_POINT_LIST
                                    (RATIONALITY, LENGTH);

          when HSV =>
            COLOURS_HSV : HSV_TYPES.CONTROL_POINT_LIST
                                    (RATIONALITY, LENGTH);

          when HLS =>
            COLOURS_HLS : HLS_TYPES.CONTROL_POINT_LIST
                                    (RATIONALITY, LENGTH);

          when others =>
            COLOURS_GENERIC : GENERIC_COLOUR_TYPES.CONTROL_POINT_LIST
                                    (RATIONALITY, LENGTH);

        end case;
      end record;

  -- Provides for specification of lists of colour control point values
  -- all of which are the same colour model.
```

COLOUR_MAPPING_DATA_RECORD

```
type COLOUR_MAPPING_DATA_RECORD
  (METHOD : COLOUR_MAPPING_METHOD) is
    record
      case METHOD is

        when TRUE_COLOUR_MAPPING =>
          null;

        when PSEUDO_COLOUR_MAPPING =>
          PSEUDO_COLOUR_MODEL  : COLOUR_MODEL;
          PSEUDO_WEIGHTS       : ACCESS_WEIGHT_LIST;
          PSEUDO_COLOURS       : ACCESS_COLOUR_VALUE_LIST;

        when PSEUDO_N_COLOUR_MAPPING =>
          PSEUDO_N_COLOUR_MODEL  : COLOUR_MODEL;
          PSEUDO_N_COLOURS       : ACCESS_LIST_OF_COLOUR_VALUE_LIST;

        when others =>
          null;

      end case;
    end record;

-- Provides for specifying values for different methods of colour mapping.
-- Additional variants may be included when additional registered or unregistered colour
-- mapping methods are supported by an implementation.
```

COLOUR_MAPPING_INDEX

```
type COLOUR_MAPPING_INDEX is new PHIGS_NATURAL;

-- Provides for index values for colour mapping representations.
```

COLOUR_MAPPING_INDICES

```
package COLOUR_MAPPING_INDICES is
        new PHIGS_LIST_UTILITIES
                (COLOUR_MAPPING_INDEX,
                  MAX_COLOUR_MAPPING_INDICES_SUPPORTED);

-- Provides for lists of index values for colour mapping representations.
```

COLOUR_MAPPING_METHOD

```
type COLOUR_MAPPING_METHOD is new PHIGS_INTEGER;

-- Provides for specifying different methods for performing colour mapping.
```

COLOUR_MAPPING_METHOD_FACILITIES

```
type COLOUR_MAPPING_METHOD_FACILITIES
  (METHOD : COLOUR_MAPPING_METHOD := TRUE_COLOUR_MAPPING) is
    record
      case METHOD is

        when TRUE_COLOUR_MAPPING =>
          NUMBER_TRUE_COLOURS_AVAIL : PHIGS_INTEGER;

        when PSEUDO_COLOUR_MAPPING =>
          NUMBER_PSEUDO_COLORS_AVAIL : PHIGS_INTEGER;

        when PSEUDO_N_COLOUR_MAPPING =>
          null;

        when others =>
          null;

      end case;
    end record;

-- Provides for colour mapping availability information.  Additional variants may be included
-- when additional registered or unregistered colour mapping methods are supported by an
-- implementation.
```

COLOUR_MAPPING_METHODS

```
package COLOUR_MAPPING_METHODS is
      new PHIGS_LIST_UTILITIES
            (COLOUR_MAPPING_METHOD,
              MAX_COLOUR_MAPPING_METHODS_SUPPORTED);

-- Provides for lists of colour mapping methods.
```

COLOUR_MAPPING_STATE

```
type COLOUR_MAPPING_STATE
  (METHOD : COLOUR_MAPPING_METHOD := TRUE_COLOUR_MAPPING) is
    record
      case METHOD is

        when TRUE_COLOUR_MAPPING =>
          null;

        when PSEUDO_COLOUR_MAPPING =>
          null;

        when PSEUDO_N_COLOUR_MAPPING =>
          null;

        when others =>
          null;

      end case;
    end record;

-- Provides for colour mapping state information.  Additional variants may be included
-- when additional registered or unregistered colour mapping methods are supported by an
-- implementation.
```

COLOUR_VALUE_ARRAY

```
type COLOUR_VALUE_ARRAY
  (MODEL    : COLOUR_MODEL := RGB;
   LENGTH_M : COLOUR_VALUE_SET_DIMENSION := 1;
   LENGTH_N : COLOUR_VALUE_SET_DIMENSION := 1)  is
    record
      case MODEL is

        when INDIRECT =>
          COLOURS_INDIRECT : INDIRECT_TYPES.COLOUR_VALUE_ARRAY
                                (1..LENGTH_M, 1..LENGTH_N);

        when RGB =>
          COLOURS_RGB : RGB_TYPES.COLOUR_VALUE_ARRAY
                            (1..LENGTH_M, 1..LENGTH_N);

        when CIELUV =>
          COLOURS_CIELUV : CIELUV_TYPES.COLOUR_VALUE_ARRAY
                               (1..LENGTH_M, 1..LENGTH_N);
```

```
            when HSV =>
              COLOURS_HSV  : HSV_TYPES.COLOUR_VALUE_ARRAY
                                (1..LENGTH_M, 1..LENGTH_N);

            when HLS =>
              COLOURS_HLS : HLS_TYPES.COLOUR_VALUE_ARRAY
                                (1..LENGTH_M, 1..LENGTH_N);

            when others =>
              COLOURS_GENERIC : GENERIC_COLOUR_TYPES.COLOUR_VALUE_ARRAY
                                (1..LENGTH_M, 1..LENGTH_N);

          end case;
        end record;

    -- Provides for specification of two-dimensional arrays of colour values all of which
    -- are the same colour model.
```

COLOUR_VALUE_LIST

```
    type COLOUR_VALUE_LIST
      (MODEL   : COLOUR_MODEL := RGB;
       LENGTH  : COLOUR_VALUE_SET_DIMENSION := 0) is
        record
          case MODEL is

            when INDIRECT =>
              COLOURS_INDIRECT : INDIRECT_TYPES.COLOUR_VALUE_LIST(1..LENGTH);

            when RGB =>
              COLOURS_RGB : RGB_TYPES.COLOUR_VALUE_LIST(1..LENGTH);

            when CIELUV =>
              COLOURS_CIELUV  : CIELUV_TYPES.COLOUR_VALUE_LIST(1..LENGTH);

            when HSV =>
              COLOURS_HSV : HSV_TYPES.COLOUR_VALUE_LIST(1..LENGTH);

            when HLS =>
              COLOURS_HLS : HLS_TYPES.COLOUR_VALUE_LIST(1..LENGTH);

            when others =>
              COLOURS_GENERIC : GENERIC_COLOUR_TYPES.COLOUR_VALUE_LIST
                                (1..LENGTH);
```

```
            end case;
        end record;

    -- Provides for specification of lists of colour values all of which are the same colour model.
```

COLOUR VALUE_SET_DIMENSION

```
    subtype COLOUR_VALUE_SET_DIMENSION is
            NATURAL range 0..MAX_COLOUR_VALUES_SUPPORTED;

    -- Provides for specifying the dimensions of colour value lists and arrays.
```

CURVE_APPROX_CRITERIA_TYPE

```
    type CURVE_APPROX_CRITERIA_TYPE is new PHIGS_INTEGER;

    -- Provides for differentiating types of curve approximation.
```

CURVE_APPROX_CRITERIA_TYPES

```
    package CURVE_APPROX_CRITERIA_TYPES is
        new PHIGS_LIST_UTILITIES
            (CURVE_APPROX_CRITERIA_TYPE,
             MAX_CURVE_APPROX_CRITERIA_TYPES_SUPPORTED);

    -- Provides for lists of curve approximation criteria.
```

CURVE_APPROX_DATA_RECORD

```
    type CURVE_APPROX_DATA_RECORD
      (CRITERIA_TYPE : CURVE_APPROX_CRITERIA_TYPE := WS_DEPENDENT_CURVE) is
        record
          case CRITERIA_TYPE is

            when WS_DEPENDENT_CURVE =>
              null;

            when CONSTANT_SUBDIVISION_CURVE =>
              COUNT : PHIGS_INTEGER;

            when WC_CHORDAL_SIZE_CURVE =>
              WC_CHORDAL_SIZE_VALUE : WC.MAGNITUDE;
```

```
            when NPC_CHORDAL_SIZE_CURVE =>
              NPC_CHORDAL_SIZE_VALUE : NPC.MAGNITUDE;

            when DC_CHORDAL_SIZE_CURVE =>
              DC_CHORDAL_SIZE_VALUE : DC.MAGNITUDE;

            when WC_CHORDAL_DEVIATION_CURVE =>
              WC_CHORDAL_DEVIATION_VALUE : WC.MAGNITUDE;

            when NPC_CHORDAL_DEVIATION_CURVE =>
              NPC_CHORDAL_DEVIATION_VALUE : NPC.MAGNITUDE;

            when DC_CHORDAL_DEVIATION_CURVE =>
              DC_CHORDAL_DEVIATION_VALUE : DC.MAGNITUDE;

            when WC_RELATIVE_CURVE =>
              WC_RELATIVE_VALUE : WC.RELATIVE_MAGNITUDE;

            when NPC_RELATIVE_CURVE =>
              NPC_RELATIVE_VALUE : NPC.RELATIVE_MAGNITUDE;

            when DC_RELATIVE_CURVE =>
              DC_RELATIVE_VALUE : DC.RELATIVE_MAGNITUDE;

            when others =>
              null;

          end case;
        end record;

    -- Provides for specifying values for different methods of curve approximation.
    -- Additional variants may be included when additional registered or unregistered curve
    -- approximation criteria types are supported by an implementation.
```

CURVE_COLOURSPLINE

```
    type CURVE_COLOURSPLINE( MODEL      : COLOUR_MODEL;
                             RATIONALITY : SPLINE_RATIONALITY;
                             KNOT_COUNT : SMALL_NATURAL;
                             LENGTH     : COLOUR_VALUE_SET_DIMENSION) is
      record
        ORDER          : SPLINE_ORDER;
        KNOTS          : KNOT_VECTOR(KNOT_COUNT);
        CONTROL_POINTS : COLOUR_CONTROL_POINT_LIST
                           (MODEL, RATIONALITY, LENGTH);
      end record;

    -- Provides for colour splines for use with Non-uniform B-Spline Curve primitives.
```

CURVE_PLACEMENT_TYPE

```
type CURVE_PLACEMENT_TYPE is (UNIFORM, NON_UNIFORM);

-- Provides for different ways of placing curves.
```

CURVE_VISIBILITY_FLAG

```
subtype CURVE_VISIBILITY_FLAG is EDGE_FLAG;

-- Provides control over display of curves on spline surfaces
```

DATA_MAPPING_DATA_RECORD

```
type DATA_MAPPING_DATA_RECORD
  (METHOD : DATA_MAPPING_METHOD := DATA_MAPPING_COLOUR) is
    record
      case METHOD is

        when DATA_MAPPING_COLOUR =>
          COLOUR_SELECTORS: SOURCE_SELECTOR_LIST;

        when SINGLE_VALUE_UNIFORM =>
          SINGLE_UNIFORM_SELECTORS          : SOURCE_SELECTOR_LIST;
          SINGLE_UNIFORM_DATA_VALUE_IND     : DATA_VALUE_INDEX;
          SINGLE_UNIFORM_RANGE              : DATA_VALUE_RANGE;
          SINGLE_UNIFORM_COLOURS            : ACCESS_COLOUR_VALUE_LIST;

        when SINGLE_VALUE_NON_UNIFORM =>
          SINGLE_NON_UNIFORM_SELECTORS      : SOURCE_SELECTOR_LIST;
          SINGLE_NON_UNIFORM_DATA_VALUE_IND : DATA_VALUE_INDEX;
          SINGLE_NON_UNIFORM_BOUNDARIES     : ACCESS_DATA_VALUE_SET;
          SINGLE_NON_UNIFORM_COLOURS        : ACCESS_COLOUR_VALUE_LIST;

        when BI_VALUE_UNIFORM =>
          BI_UNIFORM_SELECTORS : SOURCE_SELECTOR_LIST;
          BI_UNIFORM_DATA_VALUE_IND_PAIR  : DATA_VALUE_INDEX_PAIR;
          BI_UNIFORM_A_RANGE    : DATA_VALUE_RANGE;
          BI_UNIFORM_B_RANGE    : DATA_VALUE_RANGE;
          BI_UNIFORM_COLOURS    : ACCESS_LIST_OF_COLOUR_VALUE_LIST;
```

```
            when BI_VALUE_NON_UNIFORM =>
              BI_NON_UNIFORM_SELECTORS       : SOURCE_SELECTOR_LIST;
              BI_NON_UNIFORM_DATA_VALUE_IND_PAIR : DATA_VALUE_INDEX_PAIR;
              BI_NON_UNIFORM_A_BOUNDARIES : ACCESS_DATA_VALUE_SET;
              BI_NON_UNIFORM_B_BOUNDARIES : ACCESS_DATA_VALUE_SET;
              BI_NON_UNIFORM_COLOURS         : ACCESS_COLOUR_VALUE_ARRAY;

            when others =>
              null;

          end case;
        end record;

    -- Provides for specifying values for different methods of data_mapping.
    -- Additional variants may be included when additional registered or unregistered data
    -- mapping methods are supported by an implementation.
```

DATA_MAPPING_INDEX

```
    type DATA_MAPPING_INDEX is new PHIGS_NATURAL;

    -- Provides for index values for data mapping representations.
```

DATA_MAPPING_INDICES

```
    package DATA_MAPPING_INDICES is
        new PHIGS_LIST_UTILITIES
            (DATA_MAPPING_INDEX,
             MAX_DATA_MAPPING_INDICES_SUPPORTED);

    -- Provides for lists of data mapping indices.
```

DATA_MAPPING_METHOD

```
    type DATA_MAPPING_METHOD is new PHIGS_INTEGER;

    -- Provides for differentiating methods of performing data mapping.
```

DATA_MAPPING_METHODS

```
package DATA_MAPPING_METHODS is
        new PHIGS_LIST_UTILITIES
                (DATA_MAPPING_METHOD,
                 MAX_DATA_MAPPING_METHODS_SUPPORTED);
```

-- Provides for lists of methods of performing data mapping.

DATASPLINE

```
type DATASPLINE( MODEL          : COLOUR_MODEL := RGB;
                 RATIONALITY    : SPLINE_RATIONALITY := RATIONAL;
                 KNOT_COUNT_U   : SMALL_NATURAL := 4;
                 KNOT_COUNT_V   : SMALL_NATURAL := 4;
                 WIDTH          : COLOUR_VALUE_SET_DIMENSION := 4;
                 HEIGHT         : COLOUR_VALUE_SET_DIMENSION := 4) is
  record
    U_ORDER          : SPLINE_ORDER;
    V_ORDER          : SPLINE_ORDER;
    U_KNOTS          : KNOT_VECTOR(KNOT_COUNT_U);
    V_KNOTS          : KNOT_VECTOR(KNOT_COUNT_V);
    CONTROL_POINTS : SPC.CONTROL_POINT_ARRAY_3
                         (RATIONALITY, WIDTH, HEIGHT);
  end record;
```

-- Provides for data splines for use with Non-uniform B-Spline primitives.

DATASPLINE_LIST

```
type DATASPLINE_LIST is
        array (DATA_VALUE_INDEX range <>) of DATASPLINE;
```

-- Provides for lists of data mapping data splines.

DATA_VALUE

```
type DATA_VALUE is digits PHIGS_PRECISION;
```

-- Provides for data mapping input values.

DATA_VALUE_INDEX

```
type DATA_VALUE_INDEX is
        new PHIGS_POSITIVE range 1 .. MAX_DATA_VALUES_PER_VERTEX;

-- Provides for selecting from lists of data mapping input values.
```

DATA_VALUE_INDEX_PAIR

```
type DATA_VALUE_INDEX_PAIR is
  record
    INDEX_A  : DATA_VALUE_INDEX;
    INDEX_B  : DATA_VALUE_INDEX;
  end record;

-- Provides for selecting from bi-valued sets of data mapping input values.
```

DATA_VALUE_RANGE

```
type DATA_VALUE_RANGE is
  record
    UPPER   : DATA_VALUE;
    LOWER   : DATA_VALUE;
  end record;

-- Provides for specifying a range of data mapping input values.
```

DATA_VALUE_SET

```
type DATA_VALUE_SET is
  array (DATA_VALUE_INDEX range <>) of DATA_VALUE;

-- Provides for arrays (ordered lists) of data mapping input values.
```

DEPTH_CUE_INDEX

```
type DEPTH_CUE_INDEX is new PHIGS_NATURAL;

-- Provides for index values for depth cue representations.
```

DEPTH_CUE_INDICES

```
package DEPTH_CUE_INDICES is
        new PHIGS_LIST_UTILITIES
                (DEPTH_CUE_INDEX,
                 MAX_DEPTH_CUE_INDICES_SUPPORTED);

-- Provides for lists of depth cue indices.
```

DEPTH_CUE_MODE

```
type DEPTH_CUE_MODE is (SUPPRESSED, ALLOWED);

-- Provides for activation and deactivation of depth cue operations.
```

DEPTH_CUE_MODES

```
package DEPTH_CUE_MODES is
        new PHIGS_LIST_UTILITIES (DEPTH_CUE_MODE, 2);

-- Provides for lists of depth cue indices.
```

DEPTH_CUE_SCALE_FACTOR

```
subtype DEPTH_CUE_SCALE_FACTOR is SCALE_FACTOR range 0.0 .. 1.0;

-- Provides for specifying scale factors for depth cue operations.
```

DEPTH_CUE_SCALE_FACTORS

```
type DEPTH_CUE_SCALE_FACTORS is
  record
    FRONT : DEPTH_CUE_SCALE_FACTOR;
    BACK  : DEPTH_CUE_SCALE_FACTOR;
  end record;

-- Provides for specifying a pair of scale factors for apportioning colours during depth cue
-- operations.
```

EDGE_FLAG_LIST

```
type EDGE_FLAG_LIST is array (POSITIVE range <>) of EDGE_FLAG;

-- Provides for arrays of edge flags.  These will be used with various "with Data" primitives.
```

EDGE_FLAG_PAIR

```
type EDGE_FLAG_PAIR is array (1..2) of EDGE_FLAG;

-- Provides for pairs of edge flags.  These will be used with "Quadrilateral Mesh with Data"
-- primitives.
```

EDGE_FLAG_TRIPLET

```
type EDGE_FLAG_TRIPLET is array (1..3) of EDGE_FLAG;

-- Provides for triplets of edge flags.  These will be used with "Triangle Set with Data"
-- primitives.
```

FACET_CULLING_MODE

```
type FACET_CULLING_MODE is (  NO_FACET_CULLING,
                              BACKFACING,
                              FRONTFACING);

-- Provides for indicating the type of culling to be applied to facets.
```

FACET_DATA_FLAG

```
type FACET_DATA_FLAG is (  FACET_NONE,
                           FACET_COLOUR,
                           FACET_NORMAL,
                           FACET_DATA,
                           FACET_COLOUR_NORMAL,
                           FACET_COLOUR_DATA,
                           FACET_NORMAL_DATA,
                           FACET_COLOUR_NORMAL_DATA);

-- Provides for various types of facet data flags.
```

FACET_DISTINGUISHING_MODE

```
type FACET_DISTINGUISHING_MODE is new OFF_ON;

-- Provides for enabling and disabling facet distinguishing operations.
```

FACET_SET_DIMENSION

```
subtype FACET_SET_DIMENSION is
        NATURAL range 0..MAX_FACETS_SUPPORTED;

-- Provides for specifying the dimensions of facet lists and arrays.
```

GENERAL_COLOUR

```
type GENERAL_COLOUR
  (MODEL : COLOUR_MODEL := RGB) is
    record
      case MODEL is

        when INDIRECT =>
          INDEX           : COLOUR_INDEX;

        when RGB =>
          COLOUR_RGB      : RGB_COLOUR_VALUE;

        when CIELUV =>
          COLOUR_CIELUV   : CIELUV_COLOUR_VALUE;

        when HSV =>
          COLOUR_HSV      : HSV_COLOUR_VALUE;

        when HLS =>
          COLOUR_HLS      : HLS_COLOUR_VALUE;

        when others =>
          COLOUR_GENERIC  : COLOUR_COEFFICIENT_ARRAY;

      end case;
    end record;

-- Provides for specifying values for colours dependent upon colour models.  Additional
-- variants may be included when additional registered or unregistered colour models are
-- supported by an implementation.
```

GENERIC_COLOUR_TYPES

```
package GENERIC_COLOUR_TYPES is
  new PHIGS_COLOUR_TYPES( COLOUR_COEFFICIENT_ARRAY,
                          COLOUR_COEFFICIENT_ARRAY);

-- Provides for generic (unsupported) colour types.
```

HLS_COLOUR_VALUE

```
type HLS_COLOUR_VALUE is
  record
    HUE_HLS        : INTENSITY;
    LIGHTNESS_HLS  : INTENSITY;
    SATURATION_HLS : INTENSITY;
  end record;

-- Provides for specification of HLS colours.
```

HLS_HOMOGENEOUS_COLOUR_VALUE

```
type HLS_HOMOGENEOUS_COLOUR_VALUE is
  record
    HUE_HLS        : COLOUR_COEFFICIENT;
    LIGHTNESS_HLS  : COLOUR_COEFFICIENT;
    SATURATION_HLS : COLOUR_COEFFICIENT;
    W_HLS          : COLOUR_COEFFICIENT;
  end record;

-- Provides for specification of homogeneous HLS colours for use in
-- rational colourspline.
```

HLS_TYPES

```
package HLS_TYPES is
  new PHIGS_COLOUR_TYPES( HLS_COLOUR_VALUE,
                          HLS_HOMOGENEOUS_COLOUR_VALUE);

-- Provides for HLS colour types.
```

HSV_COLOUR_VALUE

```
type HSV_COLOUR_VALUE is
  record
    HUE_HSV          : INTENSITY;
    SATURATION_HSV   : INTENSITY;
    VALUE_HSV        : INTENSITY;
  end record;

-- Provides for specification of HSV colours.
```

HSV_HOMOGENEOUS_COLOUR_VALUE

```
type HSV_HOMOGENEOUS_COLOUR_VALUE is
  record
    HUE_HSV          : COLOUR_COEFFICIENT;
    SATURATION_HSV   : COLOUR_COEFFICIENT;
    VALUE_HSV        : COLOUR_COEFFICIENT;
    W_HSV            : COLOUR_COEFFICIENT;
  end record;

-- Provides for specification of homogeneous HSV colours for use in
-- rational coloursplines.
```

HSV_TYPES

```
package HSV_TYPES is
  new PHIGS_COLOUR_TYPES( HSV_COLOUR_VALUE,
                          HSV_HOMOGENEOUS_COLOUR_VALUE);

-- Provides for HSV colour types.
```

INDIRECT_HOMOGENEOUS_COLOUR_VALUE

```
type INDIRECT_HOMOGENEOUS_COLOUR_VALUE is
  record
    INDEX_INDIRECT : COLOUR_INDEX;
    W_INDIRECT     : COLOUR_COEFFICIENT;
  end record;

-- Provides for specification of homogeneous indirect colours for use in rational
-- coloursplines.
```

INDIRECT_TYPES

```
package INDIRECT_TYPES is
  new PHIGS_COLOUR_TYPES( COLOUR_INDEX,
                          INDIRECT_HOMOGENEOUS_COLOUR_VALUE);

-- Provides for INDIRECT colour types.
```

INTERIOR_SHADING_METHOD

```
type INTERIOR_SHADING_METHOD is new PHIGS_INTEGER;

-- Provides for differentiating methods of shading interiors of enclosed regions.
```

INTERIOR_SHADING_METHODS

```
package INTERIOR_SHADING_METHODS is
      new PHIGS_LIST_UTILITIES
            (INTERIOR_SHADING_METHOD,
             MAX_INTERIOR_SHADING_METHODS_SUPPORTED);

-- Provides for lists of methods of performing interior shading.
```

KNOT

```
subtype KNOT is SPC.MAGNITUDE;

-- Provides for specifying spline knots.
```

KNOT_VECTOR

```
subtype KNOT_VECTOR is SPC.MAGNITUDE_LIST;

-- Provides for specifying spline knot vectors.
```

LIGHTING_EXPONENT

```
type LIGHTING_EXPONENT is digits PHIGS_PRECISION;

-- Provides for specifying values for specular lighting exponents.
```

LIGHT_SOURCE_DATA_RECORD

```
type LIGHT_SOURCE_DATA_RECORD
  (TYPE_OF_LIGHT : LIGHT_SOURCE_TYPE := AMBIENT_LIGHT) is
    record
      case TYPE_OF_LIGHT is

        when AMBIENT_LIGHT =>
          AMBIENT_COLOUR             : GENERAL_COLOUR;

        when DIRECTIONAL_LIGHT =>
          DIRECTIONAL_COLOUR         : GENERAL_COLOUR;
          DIRECTIONAL_DIRECTION      : WC.VECTOR_3;

        when POSITIONAL_LIGHT =>
          POSITIONAL_COLOUR          : GENERAL_COLOUR;
          POSITIONAL_POSITION        : WC.POINT_3;
          POSITIONAL_ATTENUATION     : ATTENUATION_COEFFICIENTS;

        when SPOT_LIGHT =>
          SPOT_COLOUR                : GENERAL_COLOUR;
          SPOT_POSITION              : WC.POINT_3;
          SPOT_DIRECTION             : WC.VECTOR_3;
          SPOT_CONCENTRATION         : LIGHTING_EXPONENT;
          SPOT_ATTENUATION           : ATTENUATION_COEFFICIENTS;
          SPOT_SPREAD                : SPREAD_ANGLE;

        when others =>
          null;

      end case;
    end record;

-- Provides for descriptions of light sources.  Additional variants may be included when
-- additional registered or unregistered light source types are supported by an
-- implementation.
```

LIGHT_SOURCE_INDEX

```
type LIGHT_SOURCE_INDEX is new PHIGS_POSITIVE;

-- Provides for identifying light sources.
```

LIGHT_SOURCE_INDICES

```
package LIGHT_SOURCE_INDICES is
        new PHIGS_LIST_UTILITIES ( LIGHT_SOURCE_INDEX,
                                   MAX_LIGHT_SOURCE_INDICES_SUPPORTED);

-- Provides for lists of light sources.
```

LIGHT_SOURCE_TYPE

```
type LIGHT_SOURCE_TYPE is new PHIGS_INTEGER;

-- Provides for specifying kinds of light sources.
```

LIGHT_SOURCE_TYPES

```
package LIGHT_SOURCE_TYPES is
        new PHIGS_LIST_UTILITIES ( LIGHT_SOURCE_TYPE,
                                   MAX_LIGHT_SOURCE_TYPES_SUPPORTED);

-- Provides for lists of light source types.
```

LIST_OF_COLOUR_VALUE_LIST

```
type LIST_OF_COLOUR_VALUE_LIST
      (MODEL    : COLOUR_MODEL := RGB;
       LENGTH   : COLOUR_VALUE_SET_DIMENSION := 3) is
  record
    case MODEL is

      when INDIRECT =>
        COLOURS_INDIRECT
            : INDIRECT_TYPES.LIST_OF_COLOUR_VALUE_LIST(1..LENGTH);

      when RGB =>
        COLOURS_RGB
            : RGB_TYPES.LIST_OF_COLOUR_VALUE_LIST(1..LENGTH);

      when CIELUV =>
        COLOURS_CIELUV
            : CIELUV_TYPES.LIST_OF_COLOUR_VALUE_LIST(1..LENGTH);
```

```
            when HSV =>
              COLOURS_HSV
                    : HSV_TYPES.LIST_OF_COLOUR_VALUE_LIST(1..LENGTH);

            when HLS =>
              COLOURS_HLS
                    : HLS_TYPES.LIST_OF_COLOUR_VALUE_LIST(1..LENGTH);

            when others =>
                COLOURS_GENERIC
                    : GENERIC_COLOUR_TYPES.LIST_OF_COLOUR_VALUE_LIST
                                                (1..LENGTH);

          end case;
        end record;

    -- Provides for specification of lists of lists of colour values all of which are the same
    -- colour model.
```

LIST_OF_DATA_VALUE_SET

```
    type LIST_OF_DATA_VALUE_SET is
            array ( VERTEX_SET_DIMENSION range <>,
                    DATA_VALUE_INDEX range <>) of DATA_VALUE;

    -- Provides for lists of data mapping data value sets.  These will be used
    -- various output primitives.  The first index refers to the number of data value lists
    -- and the second index to the individual data values.
```

LIST_OF_EDGE_FLAG_LIST

```
    type LIST_OF_EDGE_FLAG_LIST is
      array (POSITIVE range <>) of ACCESS_EDGE_FLAG_LIST;

    -- Provides for pointers to lists of lists of edge flags.
```

LIST_OF_EDGE_FLAG_TRIPLET

```
    type LIST_OF_EDGE_FLAG_TRIPLET is
      array (POSITIVE range <>) of EDGE_FLAG_TRIPLET;

    -- Provides for lists of triplets of edge flags.
```

LIST_OF_LIST_OF_EDGE_FLAG_LIST

```
type LIST_OF_LIST_OF_EDGE_FLAG_LIST is
  array (POSITIVE range <>) of ACCESS_LIST_OF_EDGE_FLAG_LIST;

-- Provides for pointers to lists of lists of edge flags.
```

LIST_OF_LIST_OF_VERTEX_INDEX_LIST

```
type LIST_OF_LIST_OF_VERTEX_INDEX_LIST is
  array (POSITIVE range <>) of ACCESS_LIST_OF_VERTEX_INDEX_LIST;

-- Provides for pointers to lists of lists of vertex indices.
```

LIST_OF_TRIMCURVE_LIST

```
type LIST_OF_TRIMCURVE_LIST is
  array (POSITIVE range <>) of ACCESS_TRIMCURVE_LIST;

-- Provides for pointers to lists of lists of trimming curves.
```

LIST_OF_VERTEX_INDEX_LIST

```
type LIST_OF_VERTEX_INDEX_LIST is
  array (POSITIVE range <>) of ACCESS_VERTEX_INDEX_LIST;

-- Provides for lists of lists of vertex indices.
```

LIST_OF_VERTEX_INDEX_TRIPLET

```
type LIST_OF_VERTEX_INDEX_TRIPLET is
  array (POSITIVE range <>) of VERTEX_INDEX_TRIPLET;

-- Provides for lists of triplets of vertex indices.
```

PARAMETRIC_SURFACE_CHARACTERISTIC

```
type PARAMETRIC_SURFACE_CHARACTERISTIC is new PHIGS_INTEGER;

-- Provides for differentiating representations of parametric surfaces.
```

PARAMETRIC_SURFACE_CHARACTERISTICS

```
package PARAMETRIC_SURFACE_CHARACTERISTICS is
        new PHIGS_LIST_UTILITIES
                (PARAMETRIC_SURFACE_CHARACTERISTIC,
                 MAX_PARAMETRIC_SURFACE_CHARACTERISTICS_SUPPORTED);

-- Provides for lists of parametric surface characteristics types.
```

PARAMETRIC_SURFACE_DATA_RECORD

```
type PARAMETRIC_SURFACE_DATA_RECORD
  (CHARACTERISTIC : PARAMETRIC_SURFACE_CHARACTERISTIC
                        := NO_PARAMETRIC_SURFACE) is
    record
      case CHARACTERISTIC is

        when NO_PARAMETRIC_SURFACE =>
          null;

        when WS_DEPENDENT_PARAMETRIC_SURFACE =>
          null;

        when ISOPARAMETRIC_CURVES =>
          CURVE_PLACEMENT : CURVE_PLACEMENT_TYPE;
          UCOUNT          : PHIGS_NATURAL;
          VCOUNT          : PHIGS_NATURAL;

        when MC_LEVEL_CURVES =>
          MC_LEVEL_ORIGIN      : MC.POINT_3;
          MC_LEVEL_DIRECTION   : MC.VECTOR_3;
          MC_LEVEL_PARAMETERS  : MC.MAGNITUDE_LIST;

        when WC_LEVEL_CURVES =>
          WC_LEVEL_ORIGIN      : WC.POINT_3;
          WC_LEVEL_DIRECTION   : WC.VECTOR_3;
          WC_LEVEL_PARAMETERS  : WC.MAGNITUDE_LIST;

        when others =>
          null;

      end case;
    end record;

-- Provides for specifying values for different parametric surface characteristics.  Additional
-- variants may be included when additional registered or unregistered parametric surface
-- characteristics are supported by an implementation.
```

PARAMETRIC_SURFACE_INDEX

```
type PARAMETRIC_SURFACE_INDEX is new PHIGS_NATURAL;

-- Provides for index values for parametric surface representations.
```

PARAMETRIC_SURFACE_INDICES

```
package PARAMETRIC_SURFACE_INDICES is
        new PHIGS_LIST_UTILITIES
                (PARAMETRIC_SURFACE_INDEX,
                 MAX_PARAMETRIC_SURFACE_INDICES_SUPPORTED);

-- Provides for lists of parametric surface indices.
```

POLYLINE_SHADING_METHOD

```
type POLYLINE_SHADING_METHOD is new PHIGS_NATURAL;

-- Provides for values for distinguishing between various methods of polyline shading.
```

POLYLINE_SHADING_METHODS

```
package POLYLINE_SHADING_METHODS is
        new PHIGS_LIST_UTILITIES
                (POLYLINE_SHADING_METHOD,
                 MAX_POLYLINE_SHADING_METHODS_SUPPORTED);

-- Provides for lists of polyline shading methods.
```

REFLECTANCE_COEFFICIENT

```
type REFLECTANCE_COEFFICIENT is digits PHIGS_PRECISION range 0.0 .. 1.0;

-- Provides for specifying values for lighting reflectance coefficients.
```

REFLECTANCE_INDEX

```
type REFLECTANCE_INDEX is new PHIGS_NATURAL;

-- Provides for index values for reflectance representations.
```

REFLECTANCE_INDICES

```
package REFLECTANCE_INDICES is
        new PHIGS_LIST_UTILITIES ( REFLECTANCE_INDEX,
                                   MAX_REFLECTANCE_INDICES_SUPPORTED);

-- Provides for lists of reflectance indices.
```

REFLECTANCE_MODEL

```
type REFLECTANCE_MODEL is new PHIGS_INTEGER;

-- Provides for selecting the type of reflectance calculations.
```

REFLECTANCE_MODELS

```
package REFLECTANCE_MODELS is
        new PHIGS_LIST_UTILITIES
                (REFLECTANCE_MODEL,
                 MAX_REFLECTANCE_MODELS_SUPPORTED);

-- Provides for lists of reflectance models.
```

REFLECTANCE_PROPERTIES_DATA_RECORD

```
type REFLECTANCE_PROPERTIES_DATA_RECORD
        (TYPE_OF_REFLECTANCE_PROPERTIES
                : REFLECTANCE_PROPERTIES_TYPE := SIMPLE_REFLECTANCE) is
    record
      case TYPE_OF_REFLECTANCE_PROPERTIES is
```

```
            when SIMPLE_REFLECTANCE =>
              AMBIENT_REFLECTANCE   : REFLECTANCE_COEFFICIENT;
              DIFFUSE_REFLECTANCE   : REFLECTANCE_COEFFICIENT;
              SPECULAR_REFLECTANCE  : REFLECTANCE_COEFFICIENT;
              SPECULAR_COLOUR       : GENERAL_COLOUR;
              SPECULAR_EXPONENT     : LIGHTING_EXPONENT;

            when others =>
              null;

          end case;
        end record;

    -- Provides for specifying values for different types of reflectance properties.  Additional
    -- variants may be included when additional registered or unregistered reflectance
    -- properties types are supported by an implementation.
```

REFLECTANCE_PROPERTIES_TYPE

```
    type REFLECTANCE_PROPERTIES_TYPE is new PHIGS_INTEGER;

    -- Provides for selecting the type of reflectance properties.
```

REFLECTANCE_PROPERTIES_TYPES

```
    package REFLECTANCE_PROPERTIES_TYPES is
          new PHIGS_LIST_UTILITIES
                (REFLECTANCE_PROPERTIES_TYPE,
                 MAX_REFLECTANCE_PROPERTIES_TYPES_SUPPORTED);

    -- Provides for lists of reflectance properties types.
```

RETURN_DEPTH_CUE_INDEX_COUNT

```
    subtype RETURN_DEPTH_CUE_INDEX_COUNT is
              PHIGS_POSITIVE range 2..MAX_DEPTH_CUE_INDICES_SUPPORTED;

    -- Provides for restricting the returned count of predefined depth cue indices.
```

RETURN_LIGHT_SOURCE_COUNT

```
subtype RETURN_LIGHT_SOURCE_COUNT is
        PHIGS_POSITIVE range 2..PHIGS_POSITIVE'last;

-- Provides for restricting spline orders to those acceptable for returning values of the
-- facilities.
```

RETURN_SPLINE_ORDER

```
subtype RETURN_SPLINE_ORDER is
        SPLINE_ORDER range 6..SPLINE_ORDER'last;

-- Provides for restricting spline orders to those acceptable for returning values of the
-- facilities.
```

RETURN_TRIMCURVE_ORDER

```
subtype RETURN_TRIMCURVE_ORDER is
        TRIMCURVE_ORDER range 4..TRIMCURVE_ORDER'last;

-- Provides for restricting trimcurve orders to those acceptable for returning values of the
-- facilities.
```

RGB_COLOUR_VALUE

```
type RGB_COLOUR_VALUE is
  record
    RED_RGB   : INTENSITY;
    GREEN_RGB : INTENSITY;
    BLUE_RGB  : INTENSITY;
  end record;

-- Provides for specification of RGB colours.
```

RGB_HOMOGENEOUS_COLOUR_VALUE

```
type RGB_HOMOGENEOUS_COLOUR_VALUE is
  record
    RED_RGB    : COLOUR_COEFFICIENT;
    GREEN_RGB  : COLOUR_COEFFICIENT;
    BLUE_RGB   : COLOUR_COEFFICIENT;
    W_RGB      : COLOUR_COEFFICIENT;
  end record;

-- Provides for specification of homogeneous RGB colours for use in rational colourspline.
```

RGB_TYPES

```
package RGB_TYPES is
  new PHIGS_COLOUR_TYPES( RGB_COLOUR_VALUE,
                          RGB_HOMOGENEOUS_COLOUR_VALUE);

-- Provides for RGB colour types.
```

SOURCE_SELECTOR

```
type SOURCE_SELECTOR is ( COLOUR_ASPECT,
                          VERTEX_COLOUR,
                          VERTEX_DATA,
                          FACET_COLOUR,
                          FACET_DATA);

-- Provides for specifying sources for colour information.
```

SOURCE_SELECTOR_LIST

```
type SOURCE_SELECTOR_LIST is
  array (SMALL_NATURAL) of SOURCE_SELECTOR;

-- Provides for specification of ordered lists of sources for colour information.
```

SPC

```
package SPC is new PHIGS_COORDINATE_SYSTEM(SPC_TYPE);

-- Defines the Spline Parameter Coordinate System.
```

SPC_TYPE

```
type SPC_TYPE is digits PHIGS_PRECISION;

-- Defines the type of a coordinate in the Spline Parameter Coordinate System.
```

SPLINE_ORDER

```
type SPLINE_ORDER is new POSITIVE;

-- Provides for specifying orders for spline curves and surfaces.
```

SPLINE_RATIONALITY

```
type SPLINE_RATIONALITY is (RATIONAL, NON_RATIONAL);

-- Provides for specifying the rationality of splines.
```

SPREAD_ANGLE

```
subtype SPREAD_ANGLE is ANGLE range 0.0 .. PHIGS_PI;

-- Provides for specifying the amount of spread of spot light sources.
```

SURFACE_APPROX_CRITERIA_TYPE

```
type SURFACE_APPROX_CRITERIA_TYPE is new PHIGS_INTEGER;

-- Provides for differentiating methods of surface approximation.
```

SURFACE_APPROX_CRITERIA_TYPES

```
package SURFACE_APPROX_CRITERIA_TYPES is
        new PHIGS_LIST_UTILITIES
                (SURFACE_APPROX_CRITERIA_TYPE,
                 MAX_SURFACE_APPROX_CRITERIA_TYPES_SUPPORTED);

-- Provides for lists of surface approximation criteria types.
```

SURFACE_APPROX_DATA_RECORD

```
type SURFACE_APPROX_DATA_RECORD
  (CRITERIA_TYPE : SURFACE_APPROX_CRITERIA_TYPE
                   := WS_DEPENDENT_SURFACE) is
    record
      case CRITERIA_TYPE is

        when WS_DEPENDENT_SURFACE =>
          null;

        when CONSTANT_SUBDIVISION_SURFACE_BETWEEN_KNOTS =>
          CONSTANT_SUBDIVISION_UCOUNT : PHIGS_INTEGER;
          CONSTANT_SUBDIVISION_VCOUNT : PHIGS_INTEGER;

        when WC_CHORDAL_SIZE_SURFACE =>
          WC_CHORDAL_SIZE_UVALUE : WC.MAGNITUDE;
          WC_CHORDAL_SIZE_VVALUE : WC.MAGNITUDE;

        when NPC_CHORDAL_SIZE_SURFACE =>
          NPC_CHORDAL_SIZE_UVALUE : NPC.MAGNITUDE;
          NPC_CHORDAL_SIZE_VVALUE : NPC.MAGNITUDE;

        when DC_CHORDAL_SIZE_SURFACE =>
          DC_CHORDAL_SIZE_UVALUE : DC.MAGNITUDE;
          DC_CHORDAL_SIZE_VVALUE : DC.MAGNITUDE;

        when WC_PLANAR_DEVIATION_SURFACE =>
          WC_PLANAR_DEVIATION_VALUE : WC.MAGNITUDE;

        when NPC_PLANAR_DEVIATION_SURFACE =>
          NPC_PLANAR_DEVIATION_VALUE : NPC.MAGNITUDE;

        when DC_PLANAR_DEVIATION_SURFACE =>
          DC_PLANAR_DEVIATION_VALUE : DC.MAGNITUDE;

        when WC_RELATIVE_SURFACE =>
          WC_RELATIVE_VALUE : WC.RELATIVE_MAGNITUDE;

        when NPC_RELATIVE_SURFACE =>
          NPC_RELATIVE_VALUE : NPC.RELATIVE_MAGNITUDE;

        when DC_RELATIVE_SURFACE =>
          DC_RELATIVE_VALUE : DC.RELATIVE_MAGNITUDE;

        when others =>
          null;
```

```
        end case;
      end record;

    -- Provides for specifying values for different methods of surface approximation.
    -- Additional variants may be included when additional registered or unregistered surface
    -- approximation criteria types are supported by an implementation.
```

SURFACE_COLOURSPLINE

```
    type SURFACE_COLOURSPLINE
          (MODEL              : COLOUR_MODEL;
           RATIONALITY        : SPLINE_RATIONALITY;
           KNOT_COUNT_U       : SMALL_NATURAL;
           KNOT_COUNT_V       : SMALL_NATURAL;
           LENGTH_U           : COLOUR_VALUE_SET_DIMENSION;
           LENGTH_V           : COLOUR_VALUE_SET_DIMENSION) is
      record
        U_ORDER              : SPLINE_ORDER;
        V_ORDER              : SPLINE_ORDER;
        U_KNOTS              : KNOT_VECTOR(KNOT_COUNT_U);
        V_KNOTS              : KNOT_VECTOR(KNOT_COUNT_V);
        CONTROL_POINTS       : COLOUR_CONTROL_POINT_ARRAY
                                 (MODEL, RATIONALITY, LENGTH_U, LENGTH_V);
      end record;

    -- Provides for colour splines for use with Non-uniform B-Spline Surface primitives.
```

TRIMCURVE

```
    type TRIMCURVE( CRITERIA_TYPE  : CURVE_APPROX_CRITERIA_TYPE
                                     := CONSTANT_SUBDIVISION_CURVE;
                    RATIONALITY    : SPLINE_RATIONALITY := RATIONAL;
                    KNOT_COUNT     : SMALL_NATURAL := 4;
                    LENGTH         : VERTEX_SET_DIMENSION := 2) is
      record
        CRITERIA_FOR_CURVE_APPROX
                               :  CURVE_APPROX_DATA_RECORD (CRITERIA_TYPE);
        FLAG                   : CURVE_VISIBILITY_FLAG;
        ORDER                  : TRIMCURVE_ORDER;
        KNOTS                  : KNOT_VECTOR(KNOT_COUNT);
        CONTROL_POINTS         : SPC.CONTROL_POINT_LIST_2(RATIONALITY, LENGTH);
        LIMITS                 : SPC.RANGE_OF_MAGNITUDES;
      end record;

    -- Provides for specifying minimum and maximum values of spline parameters.
```

TRIMCURVE_LIST

```
type TRIMCURVE_LIST is array (POSITIVE range <>) of TRIMCURVE;

-- Provides for arrays of trimming curves.  These will be used with NURB surface primitives.
```

TRIMCURVE_ORDER

```
subtype TRIMCURVE_ORDER is SPLINE_ORDER range 2..SPLINE_ORDER'last;

-- Provides for restricting spline orders to those acceptable for trimming curves.
```

VERTEX_COLOUR_FLAG

```
subtype VERTEX_COLOUR_FLAG is
        VERTEX_DATA_FLAG range COORDINATES .. COORDINATES_COLOUR;

-- Provides identifiers for various types of vertex information used in polyline sets.
```

VERTEX_DATA_FLAG

```
type VERTEX_DATA_FLAG is ( COORDINATES,
                           COORDINATES_COLOUR,
                           COORDINATES_NORMAL,
                           COORDINATES_DATA,
                           COORDINATES_COLOUR_NORMAL,
                           COORDINATES_COLOUR_DATA,
                           COORDINATES_NORMAL_DATA,
                           COORDINATES_COLOUR_NORMAL_DATA);

-- Provides identifiers for various types of vertex information.
```

VERTEX_INDEX

```
type VERTEX_INDEX is new PHIGS_NATURAL range 1..MAX_VERTICES_SUPPORTED;

-- Provides for indices into arrays of vertices.  These will be used with various "with Data"
-- primitives.
```

VERTEX_INDEX_LIST

```
type VERTEX_INDEX_LIST is array (POSITIVE range <>) of VERTEX_INDEX;

-- Provides for arrays of indices.  These will be used with various "with Data" primitives.
```

VERTEX_INDEX_TRIPLET

```
subtype VERTEX_INDEX_TRIPLET is VERTEX_INDEX_LIST(1..3);

-- Provides for triplets of indices.  These will be used with "Triangle Set with Data"
-- primitives.
```

VERTEX_SET_COUNT

```
subtype VERTEX_SET_COUNT is
        NATURAL range 0..MAX_VERTEX_SETS_SUPPORTED;

-- Provides for specifying the number of sets in a vertex set.
```

VERTEX_SET_DIMENSION

```
subtype VERTEX_SET_DIMENSION is
        NATURAL range 0..MAX_VERTICES_SUPPORTED;

-- Provides for specifying the dimensions of vertex lists and arrays.
```

WEIGHT_LIST

```
type WEIGHT_LIST
     (LENGTH : COLOUR_COMPONENTS := MAX_COLOUR_COEFFICIENTS) is
   record
     WEIGHTS : WEIGHT_LIST_ARRAY(1..LENGTH);
   end record;

-- Provides for a variable length list of weights to be applied during colour mapping.
```

WEIGHT_LIST_ARRAY

```
type WEIGHT_LIST_ARRAY is
     array (COLOUR_COMPONENTS range <>) of COLOUR_COEFFICIENT;
```

-- Provides for a list of weights to be applied during colour mapping.

6.1.7 Additions to list of constant declarations

The following constant defines the PHIGS PLUS standard additional colour model:

```
INDIRECT                                : constant COLOUR_MODEL := 0;
```

The following constant is for use when specifying the rendering colour model:

```
WS_DEPENDENT_COLOUR_MODEL    : constant COLOUR_MODEL := 0;
```

The following constants define the PHIGS PLUS standard polyline shading methods:

```
NO_POLYLINE_SHADING              : constant POLYLINE_SHADING_METHOD := 1;
POLYLINE_COLOUR_SHADING          : constant POLYLINE_SHADING_METHOD := 2;
```

The following constants define the PHIGS PLUS standard interior shading methods:

```
NO_INTERIOR_SHADING                    : constant INTERIOR_SHADING_METHOD := 1;
INTERIOR_COLOUR_SHADING                : constant INTERIOR_SHADING_METHOD := 2;
INTERIOR_DATA_SHADING                  : constant INTERIOR_SHADING_METHOD := 3;
INTERIOR_DATA_AND_DOT_SHADING          : constant INTERIOR_SHADING_METHOD := 4;
INTERIOR_DATA_AND_NORMAL_SHADING       : constant INTERIOR_SHADING_METHOD := 5;
```

The following constants define the PHIGS PLUS standard data mapping methods:

```
DATA_MAPPING_COLOUR          : constant DATA_MAPPING_METHOD := 1;
SINGLE_VALUE_UNIFORM         : constant DATA_MAPPING_METHOD := 2;
SINGLE_VALUE_NON_UNIFORM     : constant DATA_MAPPING_METHOD := 3;
BI_VALUE_UNIFORM             : constant DATA_MAPPING_METHOD := 4;
BI_VALUE_NON_UNIFORM         : constant DATA_MAPPING_METHOD := 5;
```

The following constant defines the PHIGS PLUS standard reflectance properties type:

```
SIMPLE_REFLECTANCE  : constant REFLECTANCE_PROPERTIES_TYPE := 1;
```

The following constants define the PHIGS PLUS standard reflectance models:

```
NO_REFLECTANCE                          : constant REFLECTANCE_MODEL := 1;
AMBIENT_REFLECTANCE                     : constant REFLECTANCE_MODEL := 2;
AMBIENT_DIFFUSE_REFLECTANCE             : constant REFLECTANCE_MODEL := 3;
AMBIENT_DIFFUSE_SPECULAR_REFLECTANCE    : constant REFLECTANCE_MODEL := 4;
```

The following constants define the PHIGS PLUS standard curve approximation criteria types:

```
WS_DEPENDENT_CURVE          : constant CURVE_APPROX_CRITERIA_TYPE := 1;
CONSTANT_SUBDIVISION_CURVE  : constant CURVE_APPROX_CRITERIA_TYPE := 2;
WC_CHORDAL_SIZE_CURVE       : constant CURVE_APPROX_CRITERIA_TYPE := 3;
NPC_CHORDAL_SIZE_CURVE      : constant CURVE_APPROX_CRITERIA_TYPE := 4;
DC_CHORDAL_SIZE_CURVE       : constant CURVE_APPROX_CRITERIA_TYPE := 5;
WC_CHORDAL_DEVIATION_CURVE  : constant CURVE_APPROX_CRITERIA_TYPE := 6;
```

```
NPC_CHORDAL_DEVIATION_CURVE : constant CURVE_APPROX_CRITERIA_TYPE := 7;
DC_CHORDAL_DEVIATION_CURVE  : constant CURVE_APPROX_CRITERIA_TYPE := 8;
WC_RELATIVE_CURVE           : constant CURVE_APPROX_CRITERIA_TYPE := 9;
NPC_RELATIVE_CURVE          : constant CURVE_APPROX_CRITERIA_TYPE := 10;
DC_RELATIVE_CURVE           : constant CURVE_APPROX_CRITERIA_TYPE := 11;
```

The following constants define the PHIGS PLUS standard surface approximation criteria types:

```
WS_DEPENDENT_SURFACE            : constant SURFACE_APPROX_CRITERIA_TYPE := 1;
CONSTANT_SUBDIVISION_SURFACE_BETWEEN_KNOTS
                                : constant SURFACE_APPROX_CRITERIA_TYPE := 2;
WC_CHORDAL_SIZE_SURFACE         : constant SURFACE_APPROX_CRITERIA_TYPE := 3;
NPC_CHORDAL_SIZE_SURFACE        : constant SURFACE_APPROX_CRITERIA_TYPE := 4;
DC_CHORDAL_SIZE_SURFACE         : constant SURFACE_APPROX_CRITERIA_TYPE := 5;
WC_PLANAR_DEVIATION_SURFACE     : constant SURFACE_APPROX_CRITERIA_TYPE := 6;
NPC_PLANAR_DEVIATION_SURFACE    : constant SURFACE_APPROX_CRITERIA_TYPE := 7;
DC_PLANAR_DEVIATION_SURFACE     : constant SURFACE_APPROX_CRITERIA_TYPE := 8;
WC_RELATIVE_SURFACE             : constant SURFACE_APPROX_CRITERIA_TYPE := 9;
NPC_RELATIVE_SURFACE            : constant SURFACE_APPROX_CRITERIA_TYPE := 10;
DC_RELATIVE_SURFACE             : constant SURFACE_APPROX_CRITERIA_TYPE := 11;
```

The following constants define the PHIGS PLUS standard parametric surface characteristic types:

```
NO_PARAMETRIC_SURFACE
                          : constant PARAMETRIC_SURFACE_CHARACTERISTIC := 1;
WS_DEPENDENT_PARAMETRIC_SURFACE
                          : constant PARAMETRIC_SURFACE_CHARACTERISTIC := 2;
ISOPARAMETRIC_CURVES      : constant PARAMETRIC_SURFACE_CHARACTERISTIC := 3;
MC_LEVEL_CURVES           : constant PARAMETRIC_SURFACE_CHARACTERISTIC := 4;
WC_LEVEL_CURVES           : constant PARAMETRIC_SURFACE_CHARACTERISTIC := 5;
```

The following constants define the PHIGS PLUS standard light source types:

```
AMBIENT_LIGHT             : constant LIGHT_SOURCE_TYPE := 1;
DIRECTIONAL_LIGHT         : constant LIGHT_SOURCE_TYPE := 2;
POSITIONAL_LIGHT          : constant LIGHT_SOURCE_TYPE := 3;
SPOT_LIGHT                : constant LIGHT_SOURCE_TYPE := 4;
```

The following constants define the PHIGS PLUS standard colour mapping methods:

```
TRUE_COLOUR_MAPPING       : constant COLOUR_MAPPING_METHOD := 1;
PSEUDO_COLOUR_MAPPING     : constant COLOUR_MAPPING_METHOD := 2;
PSEUDO_N_COLOUR_MAPPING   : constant COLOUR_MAPPING_METHOD := 3;
```

6.1.8 PHIGS PLUS configuration values

The following configuration values are required by PHIGS PLUS to specify the maximum number of entities of the indicated type which are supported by the implementation:

MAX_COLOUR_MAPPING_INDICES_SUPPORTED
MAX_COLOUR_MAPPING_METHODS_SUPPORTED
MAX_COLOUR_VALUES_SUPPORTED
MAX_CURVE_APPROX_CRITERIA_TYPES_SUPPORTED
MAX_DATA_MAPPING_INDICES_SUPPORTED
MAX_DATA_MAPPING_METHODS_SUPPORTED
MAX_DATA_VALUES_PER_FACET
MAX_DATA_VALUES_PER_VERTEX
MAX_DEPTH_CUE_INDICES_SUPPORTED
MAX_FACETS_SUPPORTED
MAX_INTERIOR_SHADING_METHODS_SUPPORTED
MAX_LIGHT_SOURCE_INDICES_SUPPORTED
MAX_LIGHT_SOURCE_TYPES_SUPPORTED
MAX_PARAMETRIC_SURFACE_CHARACTERISTICS_SUPPORTED
MAX_PARAMETRIC_SURFACE_INDICES_SUPPORTED
MAX_POLYLINE_SHADING_METHODS_SUPPORTED
MAX_REFLECTANCE_INDICES_SUPPORTED
MAX_REFLECTANCE_MODELS_SUPPORTED
MAX_REFLECTANCE_PROPERTIES_TYPES_SUPPORTED
MAX_SURFACE_APPROX_CRITERIA_TYPES_SUPPORTED
MAX_VERTEX_SETS_SUPPORTED
MAX_VERTICES_SUPPORTED

The following constant defines the PHIGS PLUS standard implementation dependent value for π:

PHIGS_PI : constant ANGLE := implementation dependent precision;

7 Functions in the Ada Binding of PHIGS PLUS

7.1 Output primitive functions

POLYLINE SET 3 WITH COLOUR

```
procedure POLYLINE_SET
  (VERTICES: in MC.LIST_OF_VERTEX_COLOUR_LIST_SET_3);
```

FILL AREA SET 3 WITH DATA (without Edge Flags)

```
procedure FILL_AREA_SET_WITH_DATA
  (FACET      : in MC.FACET_DATA_SET;
   VERTICES   : in MC.LIST_OF_VERTEX_DATA_LIST_SET_3);
```

FILL AREA SET 3 WITH DATA (with Edge Flags)

```
procedure FILL_AREA_SET_WITH_DATA
  (FACET      : in MC.FACET_DATA_SET;
   EDGES      : in LIST_OF_EDGE_FLAG_LIST;
   VERTICES   : in MC.LIST_OF_VERTEX_DATA_LIST_SET_3);
```

FILL AREA SET WITH DATA (without Edge Flags)

```
procedure FILL_AREA_SET_WITH_DATA
  (FACET      : in MC.FACET_DATA_SET;
   VERTICES   : in MC.LIST_OF_VERTEX_DATA_LIST_SET_2);
```

FILL AREA SET WITH DATA (with Edge Flags)

```
procedure FILL_AREA_SET_WITH_DATA
  (FACET      : in MC.FACET_DATA_SET;
   EDGES      : in LIST_OF_EDGE_FLAG_LIST;
   VERTICES   : in MC.LIST_OF_VERTEX_DATA_LIST_SET_2);
```

CELL ARRAY 3 PLUS

```
procedure CELL_ARRAY
  (CORNER_P  : in MC.POINT_3;
   CORNER_Q  : in MC.POINT_3;
   CORNER_R  : in MC.POINT_3;
   CELLS     : in COLOUR_VALUE_ARRAY);
```

SET OF FILL AREA SETS 3 WITH DATA (without Edge Flags)

```
procedure SET_OF_FILL_AREA_SET_WITH_DATA
  (FACETS     : in MC.FACET_DATA_LIST_SET;
   VERTICES   : in MC.VERTEX_DATA_LIST_SET_3;
   INDICES    : in LIST_OF_LIST_OF_VERTEX_INDEX_LIST);
```

SET OF FILL AREA SETS 3 WITH DATA (with Edge Flags)

```
procedure SET_OF_FILL_AREA_SET_WITH_DATA
  (FACETS     : in MC.FACET_DATA_LIST_SET;
   EDGES      : in LIST_OF_LIST_OF_EDGE_FLAG_LIST;
   VERTICES   : in MC.VERTEX_DATA_LIST_SET_3;
   INDICES    : in LIST_OF_LIST_OF_VERTEX_INDEX_LIST);
```

SET OF FILL AREA SETS WITH DATA (without Edge Flags)

```
procedure SET_OF_FILL_AREA_SET_WITH_DATA
  (FACETS     : in MC.FACET_DATA_LIST_SET;
   VERTICES   : in MC.VERTEX_DATA_LIST_SET_2;
   INDICES    : in LIST_OF_LIST_OF_VERTEX_INDEX_LIST);
```

SET OF FILL AREA SETS WITH DATA (with Edge Flags)

```
procedure SET_OF_FILL_AREA_SET_WITH_DATA
  (FACETS      : in MC.FACET_DATA_LIST_SET;
   EDGES       : in LIST_OF_LIST_OF_EDGE_FLAG_LIST;
   VERTICES    : in MC.VERTEX_DATA_LIST_SET_2;
   INDICES     : in LIST_OF_LIST_OF_VERTEX_INDEX_LIST);
```

TRIANGLE SET 3 WITH DATA (without Edge Flags)

```
procedure TRIANGLE_SET_WITH_DATA
  (FACETS      : in MC.FACET_DATA_LIST_SET;
   VERTICES    : in MC.VERTEX_DATA_LIST_SET_3;
   INDICES     : in LIST_OF_VERTEX_INDEX_TRIPLET);
```

TRIANGLE SET 3 WITH DATA (with Edge Flags)

```
procedure TRIANGLE_SET_WITH_DATA
  (FACETS      : in MC.FACET_DATA_LIST_SET;
   EDGES       : in LIST_OF_EDGE_FLAG_TRIPLET;
   VERTICES    : in MC.VERTEX_DATA_LIST_SET_3;
   INDICES     : in LIST_OF_VERTEX_INDEX_TRIPLET);
```

TRIANGLE SET WITH DATA (without Edge Flags)

```
procedure TRIANGLE_SET_WITH_DATA
  (FACETS      : in MC.FACET_DATA_LIST_SET;
   VERTICES    : in MC.VERTEX_DATA_LIST_SET_2;
   INDICES     : in LIST_OF_VERTEX_INDEX_TRIPLET);
```

TRIANGLE SET WITH DATA (with Edge Flags)

```
procedure TRIANGLE_SET_WITH_DATA
  (FACETS      : in MC.FACET_DATA_LIST_SET;
   EDGES       : in LIST_OF_EDGE_FLAG_TRIPLET;
   VERTICES    : in MC.VERTEX_DATA_LIST_SET_2;
   INDICES     : in LIST_OF_VERTEX_INDEX_TRIPLET);
```

TRIANGLE STRIP 3 WITH DATA (without Edge Flags)

```
procedure TRIANGLE_STRIP_WITH_DATA
  (FACETS      : in MC.FACET_DATA_LIST_SET;
   VERTICES    : in MC.VERTEX_DATA_LIST_SET_3);
```

TRIANGLE STRIP 3 WITH DATA (with Edge Flags)

```
procedure TRIANGLE_STRIP_WITH_DATA
  (FACETS      : in MC.FACET_DATA_LIST_SET;
   EDGES       : in EDGE_FLAG_LIST;
   VERTICES    : in MC.VERTEX_DATA_LIST_SET_3);
```

TRIANGLE STRIP WITH DATA (without Edge Flags)

```
procedure TRIANGLE_STRIP_WITH_DATA
  (FACETS      : in MC.FACET_DATA_LIST_SET;
   VERTICES    : in MC.VERTEX_DATA_LIST_SET_2);
```

TRIANGLE STRIP WITH DATA (with Edge Flags)

```
procedure TRIANGLE_STRIP_WITH_DATA
  (FACETS      : in MC.FACET_DATA_LIST_SET;
   EDGES       : in EDGE_FLAG_LIST;
   VERTICES    : in MC.VERTEX_DATA_LIST_SET_2);
```

QUADRILATERAL MESH 3 WITH DATA (without Edge Flags)

```
procedure QUADRILATERAL_MESH_WITH_DATA
  (FACETS      : in MC.FACET_DATA_ARRAY_SET;
   VERTICES    : in MC.VERTEX_DATA_ARRAY_SET_3);
```

The FACETS array must have dimension one less than the dimensions of the VERTICES array. If this relationship is violated, the following binding specific error must be generated:

2502 Ignoring function, the parameters have inconsistent dimensions.

QUADRILATERAL MESH 3 WITH DATA (with Edge Flags)

```
procedure QUADRILATERAL_MESH_WITH_DATA
  (FACETS     : in MC.FACET_DATA_ARRAY_SET;
   EDGES      : in ARRAY_OF_EDGE_FLAG_PAIR;
   VERTICES   : in MC.VERTEX_DATA_ARRAY_SET_3);
```

The EDGES and VECTICES arrays must have the same dimensions. The FACETS array must have dimension one less than the dimensions of the VERTICES array. If either of these two relationships is violated, the following binding specific error must be generated:

2502 Ignoring function, the parameters have inconsistent dimensions.

QUADRILATERAL MESH WITH DATA (without Edge Flags)

```
procedure QUADRILATERAL_MESH_WITH_DATA
  (FACETS     : in MC.FACET_DATA_ARRAY_SET;
   VERTICES   : in MC.VERTEX_DATA_ARRAY_SET_2);
```

The FACETS array must have dimension one less than the dimensions of the VERTICES array. If this relationship is violated, the following binding specific error must be generated:

2502 Ignoring function, the parameters have inconsistent dimensions.

QUADRILATERAL MESH WITH DATA (with Edge Flags)

```
procedure QUADRILATERAL_MESH_WITH_DATA
  (FACETS     : in MC.FACET_DATA_ARRAY_SET;
   EDGES      : in ARRAY_OF_EDGE_FLAG_PAIR;
   VERTICES   : in MC.VERTEX_DATA_ARRAY_SET_2);
```

The EDGES and VECTICES arrays must have the same dimensions. The FACETS array must have dimension one less than the dimensions of the VERTICES array. If either of these two relationships is violated, the following binding specific error must be generated:

2502 Ignoring function, the parameters have inconsistent dimensions.

NON-UNIFORM B-SPLINE CURVE

```
procedure NON_UNIFORM_B_SPLINE_CURVE
  (SPLINE    : in MC.CURVE_GEOMETRYSPLINE_3;
   LIMITS    : in SPC.RANGE_OF_MAGNITUDES);
```

NON-UNIFORM B-SPLINE CURVE WITH COLOUR

```
procedure NON_UNIFORM_B_SPLINE_CURVE
  (CURVE_SPLINE  : in MC.CURVE_GEOMETRYSPLINE_3;
   LIMITS        : in SPC.RANGE_OF_MAGNITUDES;
   COLOUR        : in CURVE_COLOURSPLINE);
```

NON-UNIFORM B-SPLINE SURFACE

```
procedure NON_UNIFORM_B_SPLINE_SURFACE
   (SURFACE_SPLINE : in MC.SURFACE_GEOMETRYSPLINE;
   TRIM_CURVES     : in LIST_OF_TRIMCURVE_LIST);
```

NON-UNIFORM B-SPLINE SURFACE WITH DATA

```
procedure NON_UNIFORM_B_SPLINE_SURFACE_WITH_DATA
  (SURFACE_SPLINE : in MC.SURFACE_GEOMETRYSPLINE;
   TRIM_CURVES    : in LIST_OF_TRIMCURVE_LIST;
   COLOUR_SPLINE  : in SURFACE_COLOURSPLINE;
   DATA_SPLINES   : in DATASPLINE_LIST);
```

7.2 Attribute specification functions

SET DATA MAPPING INDEX

```
procedure SET_DATA_MAPPING_INDEX
  (DATA_MAPPING_IND : in DATA_MAPPING_INDEX);
```

SET REFLECTANCE INDEX

```
procedure SET_REFLECTANCE_INDEX
  (REFLECTANCE_IND : in REFLECTANCE_INDEX);
```

SET BACK INTERIOR INDEX

```
procedure SET_BACK_INTERIOR_INDEX
  (BACK_INTERIOR_IND : in INTERIOR_INDEX);
```

SET BACK DATA MAPPING INDEX

```
procedure SET_BACK_DATA_MAPPING_INDEX
  (BACK_DATA_MAPPING_IND : in DATA_MAPPING_INDEX);
```

SET BACK REFLECTANCE INDEX

```
procedure SET_BACK_REFLECTANCE_INDEX
  (BACK_REFLECTANCE_IND : in REFLECTANCE_INDEX);
```

SET PARAMETRIC SURFACE INDEX

```
procedure SET_PARAMETRIC_SURFACE_INDEX
  (PARAMETRIC_SURFACE_IND : in PARAMETRIC_SURFACE_INDEX);
```

SET POLYLINE COLOUR

```
procedure SET_POLYLINE_COLOUR
  (GENERAL_POLYLINE_COLOUR : in GENERAL_COLOUR);
```

SET POLYLINE SHADING METHOD

```
procedure SET_POLYLINE_SHADING_METHOD
  (POLYLINE_SHADING : in POLYLINE_SHADING_METHOD);
```

SET POLYMARKER COLOUR

```
procedure SET_POLYMARKER_COLOUR
  (GENERAL_POLYMARKER_COLOUR : in GENERAL_COLOUR);
```

SET TEXT COLOUR

```
procedure SET_TEXT_COLOUR
  (GENERAL_TEXT_COLOUR : in GENERAL_COLOUR);
```

SET FACET DISTINGUISHING MODE

```
procedure SET_FACET_DISTINGUISHING_MODE
  (FACET_DISTINGUISHING : in FACET_DISTINGUISHING_MODE);
```

SET FACET CULLING MODE

```
procedure SET_FACET_CULLING_MODE
  (FACET_CULLING : in FACET_CULLING_MODE);
```

SET INTERIOR COLOUR

```
procedure SET_INTERIOR_COLOUR
  (GENERAL_INTERIOR_COLOUR : in GENERAL_COLOUR);
```

SET INTERIOR SHADING METHOD

```
procedure SET_INTERIOR_SHADING_METHOD
  (INTERIOR_SHADING : in INTERIOR_SHADING_METHOD);
```

SET DATA MAPPING METHOD

```
procedure SET_DATA_MAPPING_METHOD
  (METHOD_OF_DATA_MAPPING : in DATA_MAPPING_DATA_RECORD);
```

SET REFLECTANCE PROPERTIES

```
procedure SET_REFLECTANCE_PROPERTIES
   (REFLECTANCE_DATA : in REFLECTANCE_PROPERTIES_DATA_RECORD);
```

SET_REFLECTANCE_MODEL

```
procedure SET_REFLECTANCE_MODEL
  (REFLECTANCE : in REFLECTANCE_MODEL);
```

SET BACK INTERIOR STYLE

```
procedure SET_BACK_INTERIOR_STYLE
  (BACK_STYLE_OF_INTERIOR : in INTERIOR_STYLE);
```

SET BACK INTERIOR STYLE INDEX

```
procedure SET_BACK_INTERIOR_STYLE_INDEX
  (BACK_STYLE_IND : in STYLE_INDEX);
```

SET BACK INTERIOR COLOUR

```
procedure SET_BACK_INTERIOR_COLOUR
  (BACK_GENERAL_INTERIOR_COLOUR : in GENERAL_COLOUR);
```

SET BACK INTERIOR SHADING METHOD

```
procedure SET_BACK_INTERIOR_SHADING_METHOD
  (BACK_INTERIOR_SHADING : in INTERIOR_SHADING_METHOD);
```

SET BACK DATA MAPPING METHOD

```
procedure SET_BACK_DATA_MAPPING_METHOD
  (BACK_METHOD_OF_DATA_MAPPING : in DATA_MAPPING_DATA_RECORD);
```

SET_BACK_REFLECTANCE_PROPERTIES

```
procedure SET_BACK_REFLECTANCE_PROPERTIES
   (BACK_REFLECTANCE_DATA : in REFLECTANCE_PROPERTIES_DATA_RECORD);
```

SET_BACK_REFLECTANCE_MODEL

```
procedure SET_BACK_REFLECTANCE_MODEL
  (BACK_REFLECTANCE : in REFLECTANCE_MODEL);
```

SET LIGHT SOURCE STATE

```
procedure SET_LIGHT_SOURCE_STATE
  (ACTIVATION_LIST      : in LIGHT_SOURCE_INDICES.LIST_OF;
   DEACTIVATION_LIST  : in LIGHT_SOURCE_INDICES.LIST_OF);
```

SET EDGE COLOUR

```
procedure SET_EDGE_COLOUR
  (GENERAL_EDGE_COLOUR : GENERAL_COLOUR);
```

SET CURVE APPROXIMATION CRITERIA

```
procedure SET_CURVE_APPROXIMATION_CRITERIA
  (CRITERIA_FOR_CURVE_APPROX : in CURVE_APPROX_DATA_RECORD);
```

SET SURFACE APPROXIMATION CRITERIA

```
procedure SET_SURFACE_APPROXIMATION_CRITERIA
  (CRITERIA_FOR_SURFACE_APPROX :  in  SURFACE_APPROX_DATA_RECORD);
```

SET PARAMETRIC SURFACE CHARACTERISTICS

```
procedure SET_PARAMETRIC_SURFACE_CHARACTERISTICS
  (PARAMETRIC_SURFACE_CHARACTERISTICS
                                    : in PARAMETRIC_SURFACE_DATA_RECORD);
```

SET RENDERING COLOUR MODEL

```
procedure SET_RENDERING_COLOUR_MODEL
  (RENDERING_COLOUR_MODEL : in COLOUR_MODEL);
```

SET DEPTH CUE INDEX

```
procedure SET_DEPTH_CUE_INDEX
  (DEPTH_CUE_IND : in DEPTH_CUE_INDEX);
```

SET COLOUR MAPPING INDEX

```
procedure SET_COLOUR_MAPPING_INDEX
  (COLOUR_MAPPING_IND : in COLOUR_MAPPING_INDEX);
```

SET POLYLINE REPRESENTATION PLUS

```
procedure SET_POLYLINE_REPRESENTATION
  (WS                          : in WS_ID;
   POLYLINE_IND                : in POLYLINE_INDEX;
   TYPE_OF_LINE                : in LINETYPE;
   LINEWIDTH_SF                : in LINEWIDTH;
   GENERAL_LINE_COLOUR         : in GENERAL_COLOUR;
   LINE_SHADING                : in POLYLINE_SHADING_METHOD;
   CRITERIA_FOR_CURVE_APPROX   : in CURVE_APPROX_DATA_RECORD);
```

SET POLYMARKER REPRESENTATION PLUS

```
procedure SET_POLYMARKER_REPRESENTATION
  (WS                          : in WS_ID;
   POLYMARKER_IND              : in POLYMARKER_INDEX;
   TYPE_OF_MARKER              : in MARKER_TYPE;
   SIZE                        : in MARKER_SIZE;
   GENERAL_MARKER_COLOUR       : in GENERAL_COLOUR);
```

SET TEXT REPRESENTATION PLUS

```
procedure SET_TEXT_REPRESENTATION
  (WS                          : in WS_ID;
   TEXT_IND                    : in TEXT_INDEX;
   FONT                        : in TEXT_FONT;
   PRECISION                   : in TEXT_PRECISION;
   EXPANSION                   : in CHAR_EXPANSION;
   SPACING                     : in CHAR_SPACING;
   GENERAL_TEXT_COLOUR         : in GENERAL_COLOUR);
```

SET INTERIOR REPRESENTATION PLUS

```
procedure SET_INTERIOR_REPRESENTATION
  (WS                          : in WS_ID;
   INTERIOR_IND                : in INTERIOR_INDEX;
   STYLE_OF_INTERIOR           : in INTERIOR_STYLE;
   STYLE_IND                   : in STYLE_INDEX;
   GENERAL_INTERIOR_COLOUR     : in GENERAL_COLOUR;
   INTERIOR_SHADING            : in INTERIOR_SHADING_METHOD);
```

SET EDGE REPRESENTATION PLUS

```
procedure SET_EDGE_REPRESENTATION
  (WS                    : in WS_ID;
   EDGE_IND              : in EDGE_INDEX;
   FLAG                  : in EDGE_FLAG;
   TYPE_OF_EDGE          : in EDGETYPE;
   EDGEWIDTH_SF          : in EDGEWIDTH;
   GENERAL_EDGE_COLOUR   : in GENERAL_COLOUR);
```

SET DATA MAPPING REPRESENTATION

```
procedure SET_DATA_MAPPING_REPRESENTATION
  (WS                : in WS_ID;
   DATA_MAPPING_IND  : in DATA_MAPPING_INDEX;
   DATA_MAPPING      : in DATA_MAPPING_DATA_RECORD);
```

SET REFLECTANCE REPRESENTATION

```
procedure SET_REFLECTANCE_REPRESENTATION
  (WS                      : in WS_ID;
   REFLECTANCE_IND         : in REFLECTANCE_INDEX;
   MODEL_OF_REFLECTANCE    : in REFLECTANCE_MODEL;
   REFLECTANCE_PROPERTIES  : in REFLECTANCE_PROPERTIES_DATA_RECORD);
```

SET PARAMETRIC SURFACE REPRESENTATION

```
procedure SET_PARAMETRIC_SURFACE_REPRESENTATION
  (WS                                : in WS_ID;
   PARAMETRIC_SURFACE_IND            : in PARAMETRIC_SURFACE_INDEX;
   CRITERIA_FOR_SURFACE_APPROX       : in SURFACE_APPROX_DATA_RECORD;
   PARAMETRIC_SURFACE_CHARACTERISTICS
                                     : in PARAMETRIC_SURFACE_DATA_RECORD);
```

SET PATTERN REPRESENTATION PLUS

```
procedure SET_PATTERN_REPRESENTATION
  (WS                : in WS_ID;
   PATTERN_IND       : in PATTERN_INDEX;
   GENERAL_PATTERN   : in COLOUR_VALUE_ARRAY);
```

SET LIGHT SOURCE REPRESENTATION

```
procedure SET_LIGHT_SOURCE_REPRESENTATION
  (WS                 : in WS_ID;
   LIGHT_SOURCE_IND   : in LIGHT_SOURCE_INDEX;
   LIGHT_SOURCE       : in LIGHT_SOURCE_DATA_RECORD);
```

SET DEPTH CUE REPRESENTATION

```
procedure SET_DEPTH_CUE_REPRESENTATION
  (WS                 : in WS_ID;
   DEPTH_CUE_IND      : in DEPTH_CUE_INDEX;
   MODE               : in DEPTH_CUE_MODE;
   REFERENCE_PLANES   : in NPC.RANGE_OF_MAGNITUDES;
   DEPTH_CUE_SF       : in DEPTH_CUE_SCALE_FACTORS;
   DEPTH_CUE_COLOUR   : in GENERAL_COLOUR);
```

SET COLOUR MAPPING REPRESENTATION

```
procedure SET_COLOUR_MAPPING_REPRESENTATION
  (WS                 : in WS_ID;
   COLOUR_MAPPING_IND : in COLOUR_MAPPING_INDEX;
   COLOUR_MAPPING     : in COLOUR_MAPPING_DATA_RECORD);
```

7.3 Inquiry functions

The following should be added:

INQUIRE POLYLINE REPRESENTATION PLUS

```
procedure INQ_POLYLINE_REPRESENTATION
  (WS                          : in  WS_ID;
   POLYLINE_IND                : in  POLYLINE_INDEX;
   RETURNED_VALUES             : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR             : out ERROR_NUMBER;
   TYPE_OF_LINE                : out LINETYPE;
   LINEWIDTH_SF                : out LINEWIDTH;
   GENERAL_LINE_COLOUR         : out GENERAL_COLOUR;
   LINE_SHADING                : out POLYLINE_SHADING_METHOD;
   CRITERIA_FOR_CURVE_APPROX   : out CURVE_APPROX_DATA_RECORD);
```

INQUIRE POLYMARKER REPRESENTATION PLUS

```
procedure INQ_POLYMARKER_REPRESENTATION
  (WS                          : in  WS_ID;
   POLYMARKER_IND              : in  POLYMARKER_INDEX;
   RETURNED_VALUES             : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR             : out ERROR_NUMBER;
   TYPE_OF_MARKER              : out MARKER_TYPE;
   SIZE                        : out MARKER_SIZE;
   GENERAL_MARKER_COLOUR       : out GENERAL_COLOUR);
```

INQUIRE TEXT REPRESENTATION PLUS

```
procedure INQ_TEXT_REPRESENTATION
  (WS                          : in  WS_ID;
   TEXT_IND                    : in  TEXT_INDEX;
   RETURNED_VALUES             : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR             : out ERROR_NUMBER;
   FONT                        : out TEXT_FONT;
   PRECISION                   : out TEXT_PRECISION;
   EXPANSION                   : out CHAR_EXPANSION;
   SPACING                     : out CHAR_SPACING;
   GENERAL_TEXT_COLOUR         : out GENERAL_COLOUR);
```

INQUIRE INTERIOR REPRESENTATION PLUS

```
procedure INQ_INTERIOR_REPRESENTATION
  (WS                          : in  WS_ID;
   INTERIOR_IND                : in  INTERIOR_INDEX;
   RETURNED_VALUES             : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR             : out ERROR_NUMBER;
   STYLE_OF_INTERIOR           : out INTERIOR_STYLE;
   STYLE_IND                   : out STYLE_INDEX;
   GENERAL_INTERIOR_COLOUR     : out GENERAL_COLOUR;
   INTERIOR_SHADING            : out INTERIOR_SHADING_METHOD);
```

INQUIRE EDGE REPRESENTATION PLUS

```
procedure INQ_EDGE_REPRESENTATION
  (WS                    : in  WS_ID;
   EDGE_IND              : in  EDGE_INDEX;
   RETURNED_VALUES       : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR       : out ERROR_NUMBER;
   FLAG                  : out EDGE_FLAG;
   TYPE_OF_EDGE          : out EDGETYPE;
   EDGEWIDTH_SF          : out EDGEWIDTH;
   GENERAL_EDGE_COLOUR : out GENERAL_COLOUR);
```

INQUIRE LIST OF DATA MAPPING INDICES

```
procedure INQ_LIST_OF_DATA_MAPPING_INDICES
  (WS                  : in  WS_ID;
   ERROR_INDICATOR: out ERROR_NUMBER;
   INDICES             : out DATA_MAPPING_INDICES.LIST_OF);
```

INQUIRE DATA MAPPING REPRESENTATION

```
procedure INQ_DATA_MAPPING_REPRESENTATION
  (WS                    : in  WS_ID;
   DATA_MAPPING_IND  : in  DATA_MAPPING_INDEX;
   RETURNED_VALUES   : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   DATA_MAPPING      : out DATA_MAPPING_DATA_RECORD);
```

INQUIRE LIST OF REFLECTANCE INDICES

```
procedure INQ_LIST_OF_REFLECTANCE_INDICES
  (WS                  : in  WS_ID;
   ERROR_INDICATOR : out ERROR_NUMBER;
   INDICES             : out REFLECTANCE_INDICES.LIST_OF);
```

INQUIRE REFLECTANCE REPRESENTATION

```
procedure INQ_REFLECTANCE_REPRESENTATION
  (WS                    : in  WS_ID;
   REFLECTANCE_IND       : in  REFLECTANCE_INDEX;
   RETURNED_VALUES       : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR       : out ERROR_NUMBER;
   MODEL                 : out REFLECTANCE_MODEL;
   REFLECTANCE_PROPERTIES : out REFLECTANCE_PROPERTIES_DATA_RECORD);
```

INQUIRE LIST OF PARAMETRIC SURFACE INDICES

```
procedure INQ_LIST_OF_PARAMETRIC_SURFACE_INDICES
  (WS              : in  WS_ID;
   ERROR_INDICATOR : out ERROR_NUMBER;
   INDICES         : out PARAMETRIC_SURFACE_INDICES.LIST_OF);
```

INQUIRE PARAMETRIC SURFACE REPRESENTATION

```
procedure INQ_PARAMETRIC_SURFACE_REPRESENTATION
  (WS                          : in  WS_ID;
   PARAMETRIC_SURFACE_IND      : in  PARAMETRIC_SURFACE_INDEX;
   RETURNED_VALUES             : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR             : out ERROR_NUMBER;
   CRITERIA_FOR_SURFACE_APPROX : out SURFACE_APPROX_DATA_RECORD;
   PARAMETRIC_SURFACE_CHARACTERISTICS
                               : out PARAMETRIC_SURFACE_DATA_RECORD);
```

INQUIRE PATTERN REPRESENTATION PLUS

```
procedure INQ_PATTERN_REPRESENTATION
  (WS                : in  WS_ID;
   PATTERN_IND       : in  PATTERN_INDEX;
   RETURNED_VALUES   : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   GENERAL_PATTERN   : out ACCESS_COLOUR_VALUE_ARRAY);
```

INQUIRE LIST OF LIGHT SOURCE INDICES

```
procedure INQ_LIST_OF_LIGHT_SOURCE_INDICES
  (WS                 : in  WS_ID;
   ERROR_INDICATOR    : out ERROR_NUMBER;
   INDICES            : out LIGHT_SOURCE_INDICES.LIST_OF);
```

INQUIRE LIGHT SOURCE REPRESENTATION

```
procedure INQ_LIGHT_SOURCE_REPRESENTATION
  (WS                 : in  WS_ID;
   LIGHT_SOURCE_IND   : in  LIGHT_SOURCE_INDEX;
   RETURNED_VALUES    : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR    : out ERROR_NUMBER;
   LIGHT_SOURCE       : out LIGHT_SOURCE_DATA_RECORD);
```

INQUIRE LIST OF DEPTH CUE INDICES

```
procedure INQ_LIST_OF_DEPTH_CUE_INDICES
  (WS                 : in  WS_ID;
   ERROR_INDICATOR    : out ERROR_NUMBER;
   INDICES            : out DEPTH_CUE_INDICES.LIST_OF);
```

INQUIRE DEPTH CUE REPRESENTATION

```
procedure INQ_DEPTH_CUE_REPRESENTATION
  (WS                 : in  WS_ID;
   DEPTH_CUE_IND      : in  DEPTH_CUE_INDEX;
   RETURNED_VALUES    : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR    : out ERROR_NUMBER;
   MODE               : out DEPTH_CUE_MODE;
   REFERENCE_PLANES   : out NPC.RANGE_OF_MAGNITUDES;
   DEPTH_CUE_SF       : out DEPTH_CUE_SCALE_FACTORS;
   DEPTH_CUE_COLOUR   : out GENERAL_COLOUR);
```

INQUIRE COLOUR MAPPING STATE

```
procedure INQ_COLOUR_MAPPING_STATE
  (WS                 : in  WS_ID;
   METHOD             : in  COLOUR_MAPPING_METHOD;
   ERROR_INDICATOR    : out ERROR_NUMBER;
   STATE              : out COLOUR_MAPPING_STATE);
```

INQUIRE LIST OF COLOUR MAPPING INDICES

```
procedure INQ_LIST_OF_COLOUR_MAPPING_INDICES
  (WS                 : in  WS_ID;
   ERROR_INDICATOR    : out ERROR_NUMBER;
   INDICES            : out COLOUR_MAPPING_INDICES.LIST_OF);
```

INQUIRE COLOUR MAPPING REPRESENTATION

```
procedure INQ_COLOUR_MAPPING_REPRESENTATION
  (WS                     : in  WS_ID;
   COLOUR_MAPPING_IND     : in  COLOUR_MAPPING_INDEX;
   RETURNED_VALUES        : in  RETURN_VALUE_TYPE;
   ERROR_INDICATOR        : out ERROR_NUMBER;
   COLOUR_MAPPING         : out ACCESS_COLOUR_MAPPING_DATA_RECORD);
```

INQUIRE DIRECT COLOUR MODEL FACILITIES

```
procedure INQ_DIRECT_COLOUR_MODEL_FACILITIES
  (TYPE_OF_WS         : in  WS_TYPE;
   ERROR_INDICATOR    : out ERROR_NUMBER;
   LIST_OF_MODELS     : out COLOUR_MODELS.LIST_OF);
```

INQUIRE RENDERING COLOUR MODEL FACILITIES

```
procedure INQ_RENDERING_COLOUR_MODEL_FACILITIES
  (TYPE_OF_WS         : in  WS_TYPE;
   ERROR_INDICATOR    : out ERROR_NUMBER;
   LIST_OF_MODELS     : out COLOUR_MODELS.LIST_OF);
```

INQUIRE DYNAMICS OF WORKSTATION ATTRIBUTES PLUS

```
procedure INQ_DYNAMICS_OF_WS_ATTRIBUTES
  (TYPE_OF_WS                          : in  WS_TYPE;
   ERROR_INDICATOR                     : out ERROR_NUMBER;
   DATA_MAPPING_REPRESENTATION         : out DYNAMIC_MODIFICATION;
   REFLECTANCE_REPRESENTATION          : out DYNAMIC_MODIFICATION;
   PARAMETRIC_SURFACE_REPRESENTATION   : out DYNAMIC_MODIFICATION;
   WS_LIGHT_SOURCE_REPRESENTATION      : out DYNAMIC_MODIFICATION;
   DEPTH_CUE_REPRESENTATION            : out DYNAMIC_MODIFICATION;
   COLOUR_MAPPING_REPRESENTATION       : out DYNAMIC_MODIFICATION);
```

INQUIRE POLYLINE FACILITIES PLUS

```
procedure INQ_POLYLINE_FACILITIES
  (TYPE_OF_WS                  : in  WS_TYPE;
   ERROR_INDICATOR             : out ERROR_NUMBER;
   NUMBER_OF_TYPES             : out PHIGS_INTEGER;
   LIST_OF_TYPES               : out LINETYPES.LIST_OF;
   NUMBER_OF_WIDTHS            : out PHIGS_NATURAL;
   NOMINAL_WIDTH               : out DC.MAGNITUDE;
   RANGE_OF_WIDTHS             : out DC.RANGE_OF_MAGNITUDES;
   LIST_OF_SHADING_METHODS     : out POLYLINE_SHADING_METHODS.LIST_OF;
   NUMBER_OF_INDICES           : out PHIGS_NATURAL);
```

INQUIRE PREDEFINED POLYLINE REPRESENTATION PLUS

```
procedure INQ_PREDEFINED_POLYLINE_REPRESENTATION
  (TYPE_OF_WS                  : in  WS_TYPE;
   POLYLINE_IND                : in  POLYLINE_INDEX;
   ERROR_INDICATOR             : out ERROR_NUMBER;
   TYPE_OF_LINE                : out LINETYPE;
   LINEWIDTH_SF                : out LINEWIDTH;
   GENERAL_LINE_COLOUR         : out GENERAL_COLOUR;
   LINE_SHADING                : out POLYLINE_SHADING_METHOD;
   CRITERIA_FOR_CURVE_APPROX   : out CURVE_APPROX_DATA_RECORD);
```

INQUIRE PREDEFINED POLYMARKER REPRESENTATION PLUS

```
procedure INQ_PREDEFINED_POLYMARKER_REPRESENTATION
  (TYPE_OF_WS                  : in  WS_TYPE;
   POLYMARKER_IND              : in  POLYMARKER_INDEX;
   ERROR_INDICATOR             : out ERROR_NUMBER;
   TYPE_OF_MARKER              : out MARKER_TYPE;
   SIZE                        : out MARKER_SIZE;
   GENERAL_MARKER_COLOUR       : out GENERAL_COLOUR);
```

INQUIRE PREDEFINED TEXT REPRESENTATION PLUS

```
proccdurc INQ_PREDEFINED_TEXT_REPRESENTATION
  (TYPE_OF_WS              : in  WS_TYPE;
   TEXT_IND                : in  TEXT_INDEX;
   ERROR_INDICATOR         : out ERROR_NUMBER;
   FONT                    : out TEXT_FONT;
   PRECISION               : out TEXT_PRECISION;
   EXPANSION               : out CHAR_EXPANSION;
   SPACING                 : out CHAR_SPACING;
   GENERAL_TEXT_COLOUR     : out GENERAL_COLOUR);
```

INQUIRE INTERIOR FACILITIES PLUS

```
procedure INQ_INTERIOR_FACILITIES
  (TYPE_OF_WS              : in  WS_TYPE;
   ERROR_INDICATOR         : out ERROR_NUMBER;
   LIST_OF_INTERIOR_STYLES : out INTERIOR_STYLES.LIST_OF;
   NUMBER_OF_HATCH_STYLES  : out PHIGS_NATURAL;
   LIST_OF_HATCH_STYLES    : out HATCH_STYLES.LIST_OF;
   LIST_OF_SHADING_METHODS : out INTERIOR_SHADING_METHODS.LIST_OF;
   NUMBER_OF_INDICES       : out PHIGS_NATURAL);
```

INQUIRE PREDEFINED INTERIOR REPRESENTATION PLUS

```
procedure INQ_PREDEFINED_INTERIOR_REPRESENTATION
  (TYPE_OF_WS              : in  WS_TYPE;
   INTERIOR_IND            : in  INTERIOR_INDEX;
   ERROR_INDICATOR         : out ERROR_NUMBER;
   STYLE_OF_INTERIOR       : out INTERIOR_STYLE;
   STYLE_IND               : out STYLE_INDEX;
   GENERAL_INTERIOR_COLOUR : out GENERAL_COLOUR;
   INTERIOR_SHADING        : out INTERIOR_SHADING_METHOD);
```

INQUIRE PREDEFINED EDGE REPRESENTATION PLUS

```
procedure INQ_PREDEFINED_EDGE_REPRESENTATION
  (TYPE_OF_WS              : in  WS_TYPE;
   EDGE_IND                : in  EDGE_INDEX;
   ERROR_INDICATOR         : out ERROR_NUMBER;
   FLAG                    : out EDGE_FLAG;
   TYPE_OF_EDGE            : out EDGETYPE;
   EDGEWIDTH_SF            : out EDGEWIDTH;
   GENERAL_EDGE_COLOUR     : out GENERAL_COLOUR);
```

INQUIRE DATA MAPPING FACILITIES

```
procedure INQ_DATA_MAPPING_FACILITIES
  (TYPE_OF_WS              : in  WS_TYPE;
   ERROR_INDICATOR         : out ERROR_NUMBER;
   LIST_OF_METHODS         : out DATA_MAPPING_METHODS.LIST_OF;
   NUMBER_OF_INDICES       : out PHIGS_POSITIVE);
```

INQUIRE PREDEFINED DATA MAPPING REPRESENTATION

```
procedure INQ_PREDEFINED_DATA_MAPPING_REPRESENTATION
  (TYPE_OF_WS              : in  WS_TYPE;
   DATA_MAPPING_IND        : in  DATA_MAPPING_INDEX;
   ERROR_INDICATOR         : out ERROR_NUMBER;
   DATA_MAPPING            : out DATA_MAPPING_DATA_RECORD);
```

INQUIRE REFLECTANCE FACILITIES

```
procedure INQ_REFLECTANCE_FACILITIES
  (TYPE_OF_WS              : in  WS_TYPE;
   ERROR_INDICATOR         : out ERROR_NUMBER;
   MODELS                  : out REFLECTANCE_MODELS.LIST_OF;
   PROPERTIES              : out REFLECTANCE_PROPERTIES_TYPES.LIST_OF;
   NUMBER_OF_INDICES       : out PHIGS_POSITIVE);
```

INQUIRE PREDEFINED REFLECTANCE REPRESENTATION

```
procedure INQ_PREDEFINED_REFLECTANCE_REPRESENTATION
  (TYPE_OF_WS                 : in  WS_TYPE;
   REFLECTANCE_IND            : in  REFLECTANCE_INDEX;
   ERROR_INDICATOR            : out ERROR_NUMBER;
   MODEL                      : out REFLECTANCE_MODEL;
   REFLECTANCE_PROPERTIES     : out REFLECTANCE_PROPERTIES_DATA_RECORD);
```

INQUIRE CURVE AND SURFACE FACILITIES

```
procedure INQ_CURVE_AND_SURFACE_FACILITIES
  (TYPE_OF_WS                 : in  WS_TYPE;
   ERROR_INDICATOR            : out ERROR_NUMBER;
   MAX_CURVE_ORDER            : out RETURN_SPLINE_ORDER;
   MAX_SURFACE_ORDER          : out RETURN_SPLINE_ORDER;
   MAX_TRIM_ORDER             : out RETURN_TRIMCURVE_ORDER;
   LIST_OF_CURVE_APPROX_CRITERIA_TYPES
                              : out CURVE_APPROX_CRITERIA_TYPES.LIST_OF;
   LIST_OF_SURFACE_APPROX_CRITERIA_TYPES
                              : out SURFACE_APPROX_CRITERIA_TYPES.LIST_OF;
   LIST_OF_TRIM_APPROX_CRITERIA_TYPES
                              : out CURVE_APPROX_CRITERIA_TYPES.LIST_OF;
   LIST_OF_CHARACTERISTICS
                              : out PARAMETRIC_SURFACE_CHARACTERISTICS.LIST_OF;
   NUMBER_OF_INDICES          : out PHIGS_POSITIVE);
```

INQUIRE PREDEFINED PARAMETRIC SURFACE REPRESENTATION

```
procedure INQ_PREDEFINED_PARAMETRIC_SURFACE_REPRESENTATION
  (TYPE_OF_WS                 : in  WS_TYPE;
   PARAMETRIC_SURFACE_IND     : in  PARAMETRIC_SURFACE_INDEX;
   ERROR_INDICATOR            : out ERROR_NUMBER;
   CRITERIA_FOR_SURFACE_APPROX : out SURFACE_APPROX_DATA_RECORD;
   PARAMETRIC_SURFACE_CHARACTERISTICS
                              : out PARAMETRIC_SURFACE_DATA_RECORD);
```

INQUIRE PREDEFINED PATTERN REPRESENTATION PLUS

```
procedure INQ_PREDEFINED_PATTERN_REPRESENTATION
  (TYPE_OF_WS          : in  WS_TYPE;
   PATTERN_IND         : in  PATTERN_INDEX;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   PATTERN             : out ACCESS_COLOUR_VALUE_ARRAY);
```

INQUIRE LIGHT SOURCE FACILITIES

```
procedure INQ_LIGHT_SOURCE_FACILITIES
  (TYPE_OF_WS                  : in  WS_TYPE;
   ERROR_INDICATOR             : out ERROR_NUMBER;
   LIST_OF_TYPES               : out LIGHT_SOURCE_TYPES.LIST_OF;
   MAX_SIMULTANEOUS_LIGHTS     : out PHIGS_POSITIVE;
   NUMBER_OF_INDICES           : out RETURN_LIGHT_SOURCE_COUNT);
```

INQUIRE PREDEFINED LIGHT SOURCE REPRESENTATION

```
procedure INQ_PREDEFINED_LIGHT_SOURCE_REPRESENTATION
  (TYPE_OF_WS          : in  WS_TYPE;
   LIGHT_SOURCE_IND    : in  LIGHT_SOURCE_INDEX;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   LIGHT_SOURCE        : out LIGHT_SOURCE_DATA_RECORD);
```

INQUIRE DEPTH CUE FACILITIES

```
procedure INQ_DEPTH_CUE_FACILITIES
  (TYPE_OF_WS          : in  WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   NUMBER_OF_INDICES   : out RETURN_DEPTH_CUE_INDEX_COUNT;
   AVAILABLE_MODES     : out DEPTH_CUE_MODES.LIST_OF);
```

INQUIRE PREDEFINED DEPTH CUE REPRESENTATION

```
procedure INQ_PREDEFINED_DEPTH_CUE_REPRESENTATION
  (TYPE_OF_WS           : in  WS_TYPE;
   DEPTH_CUE_IND        : in  DEPTH_CUE_INDEX;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   MODE                 : out DEPTH_CUE_MODE;
   REFERENCE_PLANES     : out NPC.RANGE_OF_MAGNITUDES;
   DEPTH_CUE_SF         : out DEPTH_CUE_SCALE_FACTORS;
   DEPTH_CUE_COLOUR     : out GENERAL_COLOUR);
```

INQUIRE COLOUR MAPPING FACILITIES

```
procedure INQ_COLOUR_MAPPING_FACILITIES
  (TYPE_OF_WS           : in  WS_TYPE;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   LIST_OF_METHODS      : out COLOUR_MAPPING_METHODS.LIST_OF;
   NUMBER_OF_INDICES    : out PHIGS_POSITIVE);
```

INQUIRE COLOUR MAPPING METHOD FACILITIES

```
procedure INQ_COLOUR_MAPPING_METHOD_FACILITIES
  (TYPE_OF_WS           : in  WS_TYPE;
   METHOD               : in  COLOUR_MAPPING_METHOD;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   FACILITIES           : out COLOUR_MAPPING_METHOD_FACILITIES);
```

INQUIRE PREDEFINED COLOUR MAPPING REPRESENTATION

```
procedure INQ_PREDEFINED_COLOUR_MAPPING_REPRESENTATION
  (TYPE_OF_WS           : in  WS_TYPE;
   COLOUR_MAPPING_IND   : in  COLOUR_MAPPING_INDEX;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   COLOUR_MAPPING       : out ACCESS_COLOUR_MAPPING_DATA_RECORD);
```

INQUIRE WORKSTATION STATE TABLE LENGTHS PLUS

```
procedure INQ_WS_STATE_TABLE_LENGTHS
  (TYPE_OF_WS                        : in  WS_TYPE;
   ERROR_INDICATOR                   : out ERROR_NUMBER;
   MAX_DATA_MAPPING_ENTRIES          : out PHIGS_NATURAL;
   MAX_REFLECTANCE_ENTRIES           : out PHIGS_NATURAL;
   MAX_PARAMETRIC_SURFACE_ENTRIES    : out PHIGS_NATURAL;
   MAX_LIGHT_SOURCE_ENTRIES          : out PHIGS_NATURAL;
   MAX_DEPTH_CUE_ENTRIES             : out PHIGS_NATURAL;
   MAX_COLOUR_MAPPING_ENTRIES        : out PHIGS_NATURAL);
```

7.4 Additional functions

7.4.1 Changes to PHIGS generic coordinate system package

The following type declaration should replace the declaration for type MAGNITUDE in the PHIGS generic coordinate system package to support PHIGS PLUS.

```
subtype MAGNITUDE is MAGNITUDE_BASE_TYPE
   range 0.0 .. COORDINATE_COMPONENT_TYPE'SAFE_LARGE;
```

7.4.1 Additions to PHIGS generic coordinate system package

The following additional data types should be added to the PHIGS generic coordinate system package to support PHIGS PLUS:

```
type POINT_4 is
  record
    X : COORDINATE_COMPONENT_TYPE;
    Y : COORDINATE_COMPONENT_TYPE;
    Z : COORDINATE_COMPONENT_TYPE;
    W : COORDINATE_COMPONENT_TYPE;
  end record;

type POINT_LIST_4 is array (POSITIVE range <>) of POINT_4;

type ACCESS_POINT_LIST_4 is access POINT_LIST_4;

type LIST_OF_POINT_LIST_4 is
  array (POSITIVE range <>) of ACCESS_POINT_LIST_4;
```

```
type ACCESS_LIST_OF_POINT_LIST_4 is access LIST_OF_POINT_LIST_4;

type POINT_ARRAY_4 is
  array (POSITIVE range <>, POSITIVE range <>) of POINT_4;

type ACCESS_POINT_ARRAY_4 is access POINT_ARRAY_4;

type POINT_ARRAY_3 is
  array (POSITIVE range <>, POSITIVE range <>) of POINT_3;

type ACCESS_POINT_ARRAY_3 is access POINT_ARRAY_3;

type POINT_ARRAY_2 is
  array (POSITIVE range <>, POSITIVE range <>) of POINT_2;

type ACCESS_POINT_ARRAY_2 is access POINT_ARRAY_2;

type CONTROL_POINT_LIST_3
        (TYPE_OF_RATIONALITY : SPLINE_RATIONALITY;
         LENGTH              : VERTEX_SET_DIMENSION) is
  record
    case TYPE_OF_RATIONALITY is

      when RATIONAL =>
        RATIONAL_POINTS : POINT_LIST_4(1..LENGTH);

      when NON_RATIONAL =>
        NON_RATIONAL_POINTS : POINT_LIST_3(1..LENGTH);

    end case;
  end record;

type ACCESS_CONTROL_POINT_LIST_3 is access CONTROL_POINT_LIST_3;

type CONTROL_POINT_LIST_2
        (TYPE_OF_RATIONALITY : SPLINE_RATIONALITY;
         LENGTH              : VERTEX_SET_DIMENSION) is
  record
    case TYPE_OF_RATIONALITY is

      when RATIONAL =>
        RATIONAL_POINTS : POINT_LIST_3(1..LENGTH);

      when NON_RATIONAL =>
        NON_RATIONAL_POINTS : POINT_LIST_2(1..LENGTH);

    end case;
  end record;

type ACCESS_CONTROL_POINT_LIST_2 is access CONTROL_POINT_LIST_2;
```

```
type CONTROL_POINT_ARRAY_3
        (TYPE_OF_RATIONALITY : SPLINE_RATIONALITY;
         LENGTH_U            : VERTEX_SET_DIMENSION;
         LENGTH_V            : VERTEX_SET_DIMENSION) is
  record
    case TYPE_OF_RATIONALITY is

      when RATIONAL =>
        RATIONAL_POINTS : POINT_ARRAY_4(1..LENGTH_U, 1..LENGTH_V);

      when NON_RATIONAL =>
        NON_RATIONAL_POINTS : POINT_ARRAY_3(1..LENGTH_U, 1..LENGTH_V);

    end case;
  end record;

type ACCESS_CONTROL_POINT_ARRAY_3 is access CONTROL_POINT_ARRAY_3;

type VECTOR_ARRAY_3 is
  array (POSITIVE range <>, POSITIVE range <>) of VECTOR_3;

type ACCESS_VECTOR_ARRAY_3 is access VECTOR_ARRAY_3;

type VECTOR_LIST_3 is
  array (POSITIVE range <>) of VECTOR_3;

type ACCESS_VECTOR_LIST_3 is access VECTOR_LIST_3;

subtype RELATIVE_MAGNITUDE is MAGNITUDE range 0.0 .. 1.0;

type MAGNITUDE_LIST_ARRAY is
        array (SMALL_NATURAL range <>) of MAGNITUDE;

type MAGNITUDE_LIST
        (LENGTH : SMALL_NATURAL := 1) is
  record
    MAGNITUDES : MAGNITUDE_LIST_ARRAY(1..LENGTH);
  end record;

type ACCESS_MAGNITUDE_LIST is access MAGNITUDE_LIST;

type ARRAY_OF_MAGNITUDE_LISTS is
  array ( POSITIVE range <>,
          POSITIVE range <>) of ACCESS_MAGNITUDE_LIST;

type ACCESS_ARRAY_OF_MAGNITUDE_LISTS is
  acccss ARRAY_OF_MAGNITUDE_LISTS;

type CURVE_GEOMETRYSPLINE_3
        (RATIONALITY : SPLINE_RATIONALITY;
```

```
        KNOT_COUNT : SMALL_NATURAL;
        LENGTH     : VERTEX_SET_DIMENSION) is
  record
    ORDER          : SPLINE_ORDER;
    KNOTS          : MAGNITUDE_LIST(KNOT_COUNT);
    CONTROL_POINTS : CONTROL_POINT_LIST_3(RATIONALITY, LENGTH);
  end record;

type ACCESS_CURVE_GEOMETRYSPLINE_3 is
  access CURVE_GEOMETRYSPLINE_3;

type CURVE_GEOMETRYSPLINE_2
      (RATIONALITY : SPLINE_RATIONALITY;
       KNOT_COUNT : SMALL_NATURAL;
       LENGTH     : VERTEX_SET_DIMENSION) is
  record
    ORDER          : SPLINE_ORDER;
    KNOTS          : MAGNITUDE_LIST(KNOT_COUNT);
    CONTROL_POINTS : CONTROL_POINT_LIST_2(RATIONALITY, LENGTH);
  end record;

type ACCESS_CURVE_GEOMETRYSPLINE_2 is
  access CURVE_GEOMETRYSPLINE_2;

type SURFACE_GEOMETRYSPLINE
      (RATIONALITY   : SPLINE_RATIONALITY;
       KNOT_COUNT_U  : SMALL_NATURAL;
       KNOT_COUNT_V  : SMALL_NATURAL;
       LENGTH_U      : VERTEX_SET_DIMENSION;
       LENGTH_V      : VERTEX_SET_DIMENSION) is
  record
    U_ORDER        : SPLINE_ORDER;
    V_ORDER        : SPLINE_ORDER;
    U_KNOTS        : MAGNITUDE_LIST(KNOT_COUNT_U);
    V_KNOTS        : MAGNITUDE_LIST(KNOT_COUNT_V);
    CONTROL_POINTS : CONTROL_POINT_ARRAY_3( RATIONALITY,
                                            LENGTH_U,
                                            LENGTH_V);
  end record;

type ACCESS_SURFACE_GEOMETRYSPLINE is
  access SURFACE_GEOMETRYSPLINE;

type FACET_COLOUR_ITEM (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
                        MODEL               : COLOUR_MODEL) is
  record
    case FACET_DATA_PROVIDED is
```

```
        when FACET_COLOUR |
             FACET_COLOUR_NORMAL |
             FACET_COLOUR_DATA |
             FACET_COLOUR_NORMAL_DATA =>
          COLOUR : GENERAL_COLOUR(MODEL);
        when others =>
          null;
      end case;
    end record;

  type FACET_NORMAL_ITEM (FACET_DATA_PROVIDED : FACET_DATA_FLAG) is
    record
      case FACET_DATA_PROVIDED is
        when FACET_NORMAL |
             FACET_COLOUR_NORMAL |
             FACET_NORMAL_DATA |
             FACET_COLOUR_NORMAL_DATA =>
          NORMAL : VECTOR_3;
        when others =>
          null;
      end case;
    end record;

  type FACET_DATA_ITEM
         (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
          PER_FACET           : DATA_VALUE_INDEX) is
    record
      case FACET_DATA_PROVIDED is
        when FACET_DATA |
             FACET_COLOUR_DATA |
             FACET_NORMAL_DATA |
             FACET_COLOUR_NORMAL_DATA =>
          DATA_ITEMS : DATA_VALUE_SET(1..PER_FACET);
        when others =>
          null;
      end case;
    end record;

  type FACET_DATA_SET
    (FACET_DATA_PROVIDED  : FACET_DATA_FLAG;
     MODEL                : COLOUR_MODEL;
     PER_FACET            : DATA_VALUE_INDEX) is
      record
        COLOUR      : FACET_COLOUR_ITEM (FACET_DATA_PROVIDED, MODEL);
        NORMAL      : FACET_NORMAL_ITEM (FACET_DATA_PROVIDED);
        DATA_ITEMS  : FACET_DATA_ITEM (FACET_DATA_PROVIDED, PER_FACET);
```

```
    end record;


type ACCESS_FACET_DATA_SET is access FACET_DATA_SET;

type FACET_COLOUR_ARRAY
      (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
       WIDTH               : FACET_SET_DIMENSION;
       HEIGHT              : FACET_SET_DIMENSION;
       MODEL               : COLOUR_MODEL) is
  record
    case FACET_DATA_PROVIDED is
      when FACET_COLOUR |
           FACET_COLOUR_NORMAL |
           FACET_COLOUR_DATA |
           FACET_COLOUR_NORMAL_DATA =>
        COLOURS : COLOUR_VALUE_ARRAY( MODEL,
                                      WIDTH,
                                      HEIGHT);

      when others =>
        null;
    end case;
  end record;

type FACET_NORMAL_ARRAY
      (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
       WIDTH               : FACET_SET_DIMENSION;
       HEIGHT              : FACET_SET_DIMENSION) is
  record
    case FACET_DATA_PROVIDED is
      when FACET_NORMAL |
           FACET_COLOUR_NORMAL |
           FACET_NORMAL_DATA |
           FACET_COLOUR_NORMAL_DATA =>
        NORMALS : VECTOR_ARRAY_3( 1..WIDTH,
                                  1..HEIGHT);
      when others =>
        null;
    end case;
  end record;

type FACET_DATA_ARRAY
      (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
       WIDTH               : FACET_SET_DIMENSION;
       HEIGHT              : FACET_SET_DIMENSION;
       PER_FACET           : DATA_VALUE_INDEX) is
  record
    case FACET_DATA_PROVIDED is
```

```
        when FACET_DATA |
             FACET_COLOUR_DATA |
             FACET_NORMAL_DATA |
             FACET_COLOUR_NORMAL_DATA =>
                  DATA_ITEMS : ARRAY_OF_DATA_VALUE_SET(1..WIDTH,
                                                       1..HEIGHT,
                                                       1..PER_FACET);
        when others =>
          null;
      end case;
    end record;

type FACET_DATA_ARRAY_SET
       (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
        WIDTH               : FACET_SET_DIMENSION;
        HEIGHT              : FACET_SET_DIMENSION;
        MODEL               : COLOUR_MODEL;
        PER_FACET           : DATA_VALUE_INDEX) is
    record
      COLOURS     : FACET_COLOUR_ARRAY( FACET_DATA_PROVIDED,
                                        WIDTH,
                                        HEIGHT,
                                        MODEL);
      NORMALS     : FACET_NORMAL_ARRAY( FACET_DATA_PROVIDED,
                                        WIDTH,
                                        HEIGHT);
      DATA_ITEMS : FACET_DATA_ARRAY( FACET_DATA_PROVIDED,
                                     WIDTH,
                                     HEIGHT,
                                     PER_FACET);
    end record;

type ACCESS_FACET_DATA_ARRAY_SET is access FACET_DATA_ARRAY_SET;

type FACET_COLOUR_LIST
       (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
        LENGTH              : FACET_SET_DIMENSION;
        MODEL               : COLOUR_MODEL) is
  record
    case FACET_DATA_PROVIDED is
```

```
        when FACET_COLOUR |
             FACET_COLOUR_NORMAL |
             FACET_COLOUR_DATA |
             FACET_COLOUR_NORMAL_DATA =>
          COLOURS : COLOUR_VALUE_LIST( MODEL,
                                       LENGTH);
        when others =>
          null;
      end case;
    end record;

  type FACET_NORMAL_LIST
        (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
         LENGTH              : FACET_SET_DIMENSION) is
    record
      case FACET_DATA_PROVIDED is
        when FACET_NORMAL |
             FACET_COLOUR_NORMAL |
             FACET_NORMAL_DATA |
             FACET_COLOUR_NORMAL_DATA =>
          NORMALS : VECTOR_LIST_3(1..LENGTH);
        when others =>
          null;
      end case;
    end record;

  type FACET_DATA_LIST
        (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
         LENGTH              : FACET_SET_DIMENSION;
         PER_FACET           : DATA_VALUE_INDEX) is
    record
      case FACET_DATA_PROVIDED is
        when FACET_DATA |
             FACET_COLOUR_DATA |
             FACET_NORMAL_DATA |
             FACET_COLOUR_NORMAL_DATA =>
          DATA_ITEMS : LIST_OF_DATA_VALUE_SET( 1..LENGTH,
                                               1..PER_FACET);
        when others =>
          null;
      end case;
    end record;

  type FACET_DATA_LIST_SET
        (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
         LENGTH              : FACET_SET_DIMENSION;
         MODEL               : COLOUR_MODEL;
         PER_FACET           : DATA_VALUE_INDEX) is
      record
```

```
        COLOURS      : FACET_COLOUR_LIST( FACET_DATA_PROVIDED,
                                          LENGTH,
                                          MODEL);
        NORMALS      : FACET_NORMAL_LIST( FACET_DATA_PROVIDED,
                                          LENGTH);
        DATA_ITEMS : FACET_DATA_LIST ( FACET_DATA_PROVIDED,
                                       LENGTH,
                                       PER_FACET);
    end record;

type ACCESS_FACET_DATA_LIST_SET is access FACET_DATA_LIST_SET;

type VERTEX_COLOUR_ARRAY
      (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
       WIDTH                : VERTEX_SET_DIMENSION;
       HEIGHT               : VERTEX_SET_DIMENSION;
       MODEL                : COLOUR_MODEL) is
  record
    case VERTEX_DATA_PROVIDED is
      when COORDINATES_COLOUR |
           COORDINATES_COLOUR_NORMAL |
           COORDINATES_COLOUR_DATA |
           COORDINATES_COLOUR_NORMAL_DATA =>
        COLOURS : COLOUR_VALUE_ARRAY( MODEL,
                                      WIDTH,
                                      HEIGHT);
      when others =>
        null;
    end case;
  end record;

type VERTEX_NORMAL_ARRAY
      (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
       WIDTH                : VERTEX_SET_DIMENSION;
       HEIGHT               : VERTEX_SET_DIMENSION) is
  record
    case VERTEX_DATA_PROVIDED is
```

```
        when COORDINATES_NORMAL |
             COORDINATES_COLOUR_NORMAL |
             COORDINATES_NORMAL_DATA |
             COORDINATES_COLOUR_NORMAL_DATA =>
          NORMALS : VECTOR_ARRAY_3( 1..WIDTH,
                                    1..HEIGHT);
        when others =>
          null;
      end case;
    end record;

  type VERTEX_DATA_ARRAY
        (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
         WIDTH                : VERTEX_SET_DIMENSION;
         HEIGHT               : VERTEX_SET_DIMENSION;
         PER_VERTEX           : DATA_VALUE_INDEX) is
    record
      case VERTEX_DATA_PROVIDED is
        when COORDINATES_DATA |
             COORDINATES_COLOUR_DATA |
             COORDINATES_NORMAL_DATA |
             COORDINATES_COLOUR_NORMAL_DATA =>
          DATA_ITEMS : ARRAY_OF_DATA_VALUE_SET( 1..WIDTH,
                                                1..HEIGHT,
                                                1..PER_VERTEX);
        when others =>
          null;
      end case;
    end record;

  type VERTEX_DATA_ARRAY_SET_3
        (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
         WIDTH                : VERTEX_SET_DIMENSION;
         HEIGHT               : VERTEX_SET_DIMENSION;
         MODEL                : COLOUR_MODEL;
         PER_VERTEX           : DATA_VALUE_INDEX) is
      record
        POINTS      : POINT_ARRAY_3( 1..WIDTH,
                                     1..HEIGHT);

        COLOURS     : VERTEX_COLOUR_ARRAY( VERTEX_DATA_PROVIDED,
                                           WIDTH,
                                           HEIGHT,
                                           MODEL);
        NORMALS     : VERTEX_NORMAL_ARRAY( VERTEX_DATA_PROVIDED,
                                           WIDTH,
                                           HEIGHT);
        DATA_ITEMS : VERTEX_DATA_ARRAY( VERTEX_DATA_PROVIDED,
                                        WIDTH,
```

```
                                                HEIGHT,
                                                PER_VERTEX);
    end record;

type ACCESS_VERTEX_DATA_ARRAY_SET_3 is access VERTEX_DATA_ARRAY_SET_3;

type VERTEX_DATA_ARRAY_SET_2
      (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
       WIDTH                : VERTEX_SET_DIMENSION;
       HEIGHT               : VERTEX_SET_DIMENSION;
       MODEL                : COLOUR_MODEL;
       PER_VERTEX           : DATA_VALUE_INDEX) is
    record
      POINTS      : POINT_ARRAY_2( 1..WIDTH,
                                   1..HEIGHT);

      COLOURS     : VERTEX_COLOUR_ARRAY( VERTEX_DATA_PROVIDED,
                                         WIDTH,
                                         HEIGHT,
                                         MODEL);
      NORMALS     : VERTEX_NORMAL_ARRAY( VERTEX_DATA_PROVIDED,
                                         WIDTH,
                                         HEIGHT);
      DATA_ITEMS : VERTEX_DATA_ARRAY( VERTEX_DATA_PROVIDED,
                                      WIDTH,
                                      HEIGHT,
                                      PER_VERTEX);
    end record;

type ACCESS_VERTEX_DATA_ARRAY_SET_2 is access VERTEX_DATA_ARRAY_SET_2;

type VERTEX_COLOUR_LIST
      (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
       LENGTH               : VERTEX_SET_DIMENSION;
       MODEL                : COLOUR_MODEL) is
  record
    case VERTEX_DATA_PROVIDED is
      when COORDINATES_COLOUR |
           COORDINATES_COLOUR_NORMAL |
           COORDINATES_COLOUR_DATA |
           COORDINATES_COLOUR_NORMAL_DATA =>
        COLOURS : COLOUR_VALUE_LIST( MODEL,
                                     LENGTH);
      when others =>
        null;
    end case;
  end record;

type VERTEX_NORMAL_LIST
```

```
      (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
       LENGTH               : VERTEX_SET_DIMENSION) is
  record
    case VERTEX_DATA_PROVIDED is
      when COORDINATES_NORMAL |
           COORDINATES_COLOUR_NORMAL |
           COORDINATES_NORMAL_DATA |
           COORDINATES_COLOUR_NORMAL_DATA =>
        NORMALS : VECTOR_LIST_3(1..LENGTH);
      when others =>
        null;
    end case;
  end record;

type VERTEX_DATA_LIST
      (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
       LENGTH               : VERTEX_SET_DIMENSION;
       PER_VERTEX           : DATA_VALUE_INDEX) is
  record
    case VERTEX_DATA_PROVIDED is
      when COORDINATES_DATA |
           COORDINATES_COLOUR_DATA |
           COORDINATES_NORMAL_DATA |
           COORDINATES_COLOUR_NORMAL_DATA =>
        DATA_ITEMS : LIST_OF_DATA_VALUE_SET( 1..LENGTH,
                                             1..PER_VERTEX);
      when others =>
        null;
    end case;
  end record;

type VERTEX_DATA_LIST_SET_3
      (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
       LENGTH               : VERTEX_SET_DIMENSION;
       MODEL                : COLOUR_MODEL;
       PER_VERTEX           : DATA_VALUE_INDEX) is
    record
      POINTS      : POINT_LIST_3(1..LENGTH);

      COLOURS     : VERTEX_COLOUR_LIST( VERTEX_DATA_PROVIDED,
                                        LENGTH,
                                        MODEL);
      NORMALS     : VERTEX_NORMAL_LIST( VERTEX_DATA_PROVIDED,
                                        LENGTH);
      DATA_ITEMS : VERTEX_DATA_LIST( VERTEX_DATA_PROVIDED,
                                     LENGTH,
                                     PER_VERTEX);
```

```
    end record;

type ACCESS_VERTEX_DATA_LIST_SET_3 is access VERTEX_DATA_LIST_SET_3;

type LIST_OF_VERTEX_DATA_LIST_SET_3 is array (VERTEX_SET_COUNT) of
        ACCESS_VERTEX_DATA_LIST_SET_3;

type ACCESS_LIST_OF_VERTEX_DATA_LIST_SET_3 is
        access LIST_OF_VERTEX_DATA_LIST_SET_3;

type VERTEX_DATA_LIST_SET_2
        (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
         LENGTH               : VERTEX_SET_DIMENSION;
         MODEL                : COLOUR_MODEL;
         PER_VERTEX           : DATA_VALUE_INDEX) is
    record
      POINTS      : POINT_LIST_2(1..LENGTH);
      COLOURS     : VERTEX_COLOUR_LIST( VERTEX_DATA_PROVIDED,
                                        LENGTH,
                                        MODEL);
      NORMALS     : VERTEX_NORMAL_LIST( VERTEX_DATA_PROVIDED,
                                        LENGTH);
      DATA_ITEMS : VERTEX_DATA_LIST( VERTEX_DATA_PROVIDED,
                                     LENGTH,
                                     PER_VERTEX);
    end record;

type ACCESS_VERTEX_DATA_LIST_SET_2 is access VERTEX_DATA_LIST_SET_2;

type LIST_OF_VERTEX_DATA_LIST_SET_2 is array (VERTEX_SET_COUNT) of
        ACCESS_VERTEX_DATA_LIST_SET_2;

type ACCESS_LIST_OF_VERTEX_DATA_LIST_SET_2 is
        access LIST_OF_VERTEX_DATA_LIST_SET_2;

type VERTEX_COLOUR_LIST_SET_3
        (VERTEX_COLOUR_PROVIDED   : VERTEX_COLOUR_FLAG;
         LENGTH                   : VERTEX_SET_DIMENSION;
         MODEL                    : COLOUR_MODEL) is
    record
      POINTS      : POINT_LIST_3(1..LENGTH);
      COLOURS     : VERTEX_COLOUR_LIST( VERTEX_COLOUR_PROVIDED,
                                        LENGTH,
                                        MODEL);
    end record;

type ACCESS_VERTEX_COLOUR_LIST_SET_3 is access VERTEX_COLOUR_LIST_SET_3;

type LIST_OF_VERTEX_COLOUR_LIST_SET_3 is array (VERTEX_SET_COUNT) of
        ACCESS_VERTEX_COLOUR_LIST_SET_3;
```

```
type ACCESS_LIST_OF_VERTEX_COLOUR_LIST_SET_3 is
        access LIST_OF_VERTEX_COLOUR_LIST_SET_3;

procedure DEALLOCATE (OBJECT: in out LIST_OF_VERTEX_DATA_LIST_SET_3);

procedure DEALLOCATE (OBJECT: in out LIST_OF_VERTEX_DATA_LIST_SET_2);

procedure DEALLOCATE (OBJECT: in out LIST_OF_VERTEX_COLOUR_LIST_SET_3);
```

7.4.2 PHIGS PLUS generic colour package

The generic package PHIGS_COLOUR_TYPES is instantiated in the PHIGS_TYPES package once for each colour model defined in PHIGS and once for a generic undefined colour model. It may also be instantiated once for each additional colour model supported by the implementation of this binding. The package defines the representation of the following data types which are to be replicated for each supported colour model:

COLOUR_VALUE_LIST
ACCESS_COLOUR_VALUE_LIST
HOMOGENEOUS_COLOUR_VALUE_LIST
ACCESS_HOMOGENEOUS_COLOUR_VALUE_LIST
LIST_OF_COLOUR_VALUE_LIST
ACCESS_LIST_OF_COLOUR_VALUE_LIST
COLOUR_VALUE_ARRAY
ACCESS_COLOUR_VALUE_ARRAY
HOMOGENEOUS_COLOUR_VALUE_ARRAY
ACCESS_HOMOGENEOUS_COLOUR_VALUE_ARRAY
CONTROL_POINT_LIST
ACCESS_CONTROL_POINT_LIST
CONTROL_POINT_ARRAY
ACCESS_CONTROL_POINT_ARRAY

The declaration for this generic package is shown below:

```
generic

    type COLOUR_VALUE is private;

    type HOMOGENEOUS_COLOUR_VALUE is private;

package PHIGS_COLOUR_TYPES is
```

```
type COLOUR_VALUE_LIST is
  array (COLOUR_VALUE_SET_DIMENSION range <>)
    of COLOUR_VALUE;

type ACCESS_COLOUR_VALUE_LIST is access COLOUR_VALUE_LIST;

type HOMOGENEOUS_COLOUR_VALUE_LIST is
  array (COLOUR_VALUE_SET_DIMENSION range <>)
    of HOMOGENEOUS_COLOUR_VALUE;

type ACCESS_HOMOGENEOUS_COLOUR_VALUE_LIST is
  access HOMOGENEOUS_COLOUR_VALUE_LIST;

type LIST_OF_COLOUR_VALUE_LIST is
  array (COLOUR_VALUE_SET_DIMENSION range <>)
    of ACCESS_COLOUR_VALUE_LIST;

type ACCESS_LIST_OF_COLOUR_VALUE_LIST is
  access LIST_OF_COLOUR_VALUE_LIST;

type COLOUR_VALUE_ARRAY is
  array ( COLOUR_VALUE_SET_DIMENSION range <>,
          COLOUR_VALUE_SET_DIMENSION range <> )
    of COLOUR_VALUE;

type ACCESS_COLOUR_VALUE_ARRAY is access COLOUR_VALUE_ARRAY;

type HOMOGENEOUS_COLOUR_VALUE_ARRAY is
  array ( COLOUR_VALUE_SET_DIMENSION range <>,
          COLOUR_VALUE_SET_DIMENSION range <> )
    of HOMOGENEOUS_COLOUR_VALUE;

type ACCESS_HOMOGENEOUS_COLOUR_VALUE_ARRAY is
  access HOMOGENEOUS_COLOUR_VALUE_ARRAY;

type CONTROL_POINT_LIST
       (TYPE_OF_RATIONALITY : SPLINE_RATIONALITY;
        LENGTH              : COLOUR_VALUE_SET_DIMENSION) is
  record
    case TYPE_OF_RATIONALITY is

      when RATIONAL =>
        RATIONAL_POINTS : HOMOGENEOUS_COLOUR_VALUE_LIST(1..LENGTH);

      when NON_RATIONAL =>
        NON_RATIONAL_POINTS : COLOUR_VALUE_LIST(1..LENGTH);

    end case;
  end record;
```

```
type ACCESS_CONTROL_POINT_LIST is access CONTROL_POINT_LIST;

type CONTROL_POINT_ARRAY
        (TYPE_OF_RATIONALITY : SPLINE_RATIONALITY;
         LENGTH_U            : COLOUR_VALUE_SET_DIMENSION;
         LENGTH_V            : COLOUR_VALUE_SET_DIMENSION) is
  record
    case TYPE_OF_RATIONALITY is

      when RATIONAL =>
        RATIONAL_POINTS : HOMOGENEOUS_COLOUR_VALUE_ARRAY
                              (1..LENGTH_U, 1..LENGTH_V);

      when NON_RATIONAL =>
        NON_RATIONAL_POINTS : COLOUR_VALUE_ARRAY
                                (1..LENGTH_U, 1..LENGTH_V);

    end case;
  end record;

type ACCESS_CONTROL_POINT_ARRAY is access CONTROL_POINT_ARRAY;

end PHIGS_COLOUR_TYPES;
```

7.4.3 Deallocation of PHIGS PLUS structure element records

The following additional DEALLOCATE functions are defined:

DEALLOCATE

```
procedure DEALLOCATE (OBJECT: in out COLOUR_MAPPING_DATA_RECORD);
```

DEALLOCATE

```
procedure DEALLOCATE (OBJECT: in out COLOUR_VALUE_LIST);
```

DEALLOCATE

```
procedure DEALLOCATE (OBJECT: in out CURVE_COLOURSPLINE);
```

DEALLOCATE

procedure DEALLOCATE (OBJECT: in out DATA_MAPPING_DATA_RECORD);

DEALLOCATE

procedure DEALLOCATE (OBJECT: in out DATASPLINE_LIST);

DEALLOCATE

procedure DEALLOCATE (OBJECT: in out DATA_VALUE_SET);

DEALLOCATE

procedure DEALLOCATE (OBJECT: in out LIST_OF_COLOUR_VALUE_LIST);

DEALLOCATE

procedure DEALLOCATE (OBJECT: in out LIST_OF_EDGE_FLAG_LIST);

DEALLOCATE

procedure DEALLOCATE (OBJECT: in out LIST_OF_LIST_OF_EDGE_FLAG_LIST);

DEALLOCATE

procedure DEALLOCATE (OBJECT: in out LIST_OF_LIST_OF_VERTEX_INDEX_LIST);

DEALLOCATE

procedure DEALLOCATE (OBJECT: in out LIST_OF_TRIMCURVE_LIST);

DEALLOCATE

procedure DEALLOCATE (OBJECT: in out LIST_OF_VERTEX_INDEX_LIST);

DEALLOCATE

procedure DEALLOCATE (OBJECT: in out SURFACE_COLOURSPLINE);

DEALLOCATE

```
procedure DEALLOCATE (OBJECT: in out TRIMCURVE_LIST);
```

Annex A

(informative)

Compilable PHIGS Specification

The following should replace the existing content:

This annex is intended to show one example of a compilable PHIGS PLUS binding to Ada. It is not the only possible such example. It is expected that implementations of a PHIGS PLUS binding to Ada will need to add additional constructs to the various packages or rearrange the constructs in a manner sutiable to the implementation. However, all names defined in this binding shall be visible as dictated in this standard.


```
-- PHIGS Configuration Package


    package PHIGS_CONFIGURATION is

-- The following define example values for the PHIGS configuration names.
-- Note that the values specified are implementation dependent.  The values
-- shown below are arbitrary and have been chosen solely as examples.  More
-- appropriate values should be chosen by an implementation.  Note also that
-- only the technique of defining the names as constants was used in this
-- example.  An implementation is free to choose the most appropriate means
-- of providing the value as discussed in 4.2.5.

  DEFAULT_ERROR_FILE                          : constant STRING := "ERROR.FIL";
  DEFAULT_LIST_SIZE                           : constant := 256;
  DEFAULT_MEMORY_UNITS                        : constant := 0;
  MAX_ANNOTATION_STYLES_SUPPORTED             : constant := 2;
  MAX_APPLICATION_DATA                        : constant := 256;
  MAX_ARCHIVE_IDS_SUPPORTED                   : constant := 4;
  MAX_CHAR_SETS_SUPPORTED                     : constant := 1;
  MAX_CHOICE_PROMPTS_SUPPORTED                : constant := 32;
  MAX_CHOICE_PROMPT_ECHO_TYPES_SUPPORTED      : constant := 10;
  MAX_CHOICE_SMALL_NATURAL                    : constant := 64;
  MAX_COLOUR_COEFFICIENTS                     : constant := 4;
  MAX_COLOUR_INDICES_SUPPORTED                : constant := 4096;
```

```
MAX_COLOUR_MATRIX_SMALL_NATURAL                : constant := 128;
MAX_COLOUR_MODELS_SUPPORTED                    : constant := 2;
MAX_EDGE_INDICES_SUPPORTED                     : constant := 5;
MAX_EDGETYPES_SUPPORTED                        : constant := 1;
MAX_ESCAPE_IDS_SUPPORTED                       : constant := 16;
MAX_FILE_IDS_SUPPORTED                         : constant := 10;
MAX_FILE_SMALL_NATURAL                         : constant := 80;
MAX_GDP_3_IDS_SUPPORTED                        : constant := 24;
MAX_GDP_IDS_SUPPORTED                          : constant := 24;
MAX_GSE_IDS_SUPPORTED                          : constant := 8;
MAX_HATCH_STYLES_SUPPORTED                     : constant := 64;
MAX_HLHSR_IDS_SUPPORTED                        : constant := 128;
MAX_HLHSR_MODES_SUPPORTED                      : constant := 4;
MAX_INPUT_STRING_SMALL_NATURAL                 : constant := 80;
MAX_INTERIOR_INDICES_SUPPORTED                 : constant := 5;
MAX_LINETYPES_SUPPORTED                        : constant := 5;
MAX_LOCATOR_PROMPT_ECHO_TYPES_SUPPORTED        : constant := 8;
MAX_MARKER_TYPES_SUPPORTED                     : constant := 256;
MAX_MEMORY_UNITS                               : constant := 0;
MAX_METAFILE_ITEM_LENGTH                       : constant := 1024;
MAX_METAFILE_ITEM_TYPE                         : constant := 256;
MAX_MODELLING_CLIP_OPERATION_TYPES_SUPPORTED   : constant := 2;
MAX_MODELLING_CLIP_DISTINCT_PLANES_SUPPORTED   : constant := 6;
MAX_NAME_SUPPORTED                             : constant := 63;
MAX_NAME_SET_FILTER_LIST_LENGTH_SUPPORTED      : constant := 4;
MAX_OPEN_WS_SUPPORTED                          : constant := 4;
MAX_PATH_DEPTH_SUPPORTED                       : constant := 64;
MAX_PATTERN_INDICES_SUPPORTED                  : constant := 4;
MAX_PICK_IDS_SUPPORTED                         : constant := 64536;
MAX_PICK_PROMPT_ECHO_TYPES_SUPPORTED           : constant := 2;
MAX_POLYLINE_INDICES_SUPPORTED                 : constant := 5;
MAX_POLYMARKER_INDICES_SUPPORTED               : constant := 5;
MAX_POSTED_STRUCTURES_SUPPORTED                : constant := 4096;
MAX_REFERENCE_PATHS_SUPPORTED                  : constant := 64;
MAX_SMALL_NATURAL                              : constant := 255;
MAX_STRING_PROMPT_ECHO_TYPES_SUPPORTED         : constant := 2;
MAX_STRING_SMALL_NATURAL                       : constant := 255;
MAX_STROKE_PROMPT_ECHO_TYPES_SUPPORTED         : constant := 2;
MAX_STRUCTURE_IDS_SUPPORTED                    : constant := 65535;
MAX_TEXT_FONT_PRECISION_PAIRS_SUPPORTED        : constant := 32;
MAX_TEXT_INDICES_SUPPORTED                     : constant := 5;
MAX_VALUATOR_PROMPT_ECHO_TYPES_SUPPORTED       : constant := 4;
MAX_VIEW_INDICES_SUPPORTED                     : constant := 20;
MAX_WS_IDS_SUPPORTED                           : constant := 16383;
MAX_WS_TYPES_SUPPORTED                         : constant := 64;
MIN_METAFILE_ITEM_TYPE                         : constant := 0;
MIN_INTEGER_VALUE                              : constant := -2147483648;
MAX_INTEGER_VALUE                              : constant := 2147483647;
PHIGS_PRECISION                                : constant := 7;
```

```
  PHIGS_PI                                                    : constant := 3.1415926;

  MAX_COLOUR_MAPPING_INDICES_SUPPORTED                        : constant := 6;
  MAX_COLOUR_MAPPING_METHODS_SUPPORTED                        : constant := 3;
  MAX_COLOUR_VALUES_SUPPORTED                                 : constant := 4096;
  MAX_CURVE_APPROX_CRITERIA_TYPES_SUPPORTED                   : constant := 4;
  MAX_DATA_MAPPING_INDICES_SUPPORTED                          : constant := 20;
  MAX_DATA_MAPPING_METHODS_SUPPORTED                          : constant := 2;
  MAX_DATA_VALUES_PER_FACET                                   : constant := 10;
  MAX_DATA_VALUES_PER_VERTEX                                  : constant := 10;
  MAX_DEPTH_CUE_INDICES_SUPPORTED                             : constant := 20;
  MAX_FACETS_SUPPORTED                                        : constant := 2048;
  MAX_INTERIOR_SHADING_METHODS_SUPPORTED                      : constant := 5;
  MAX_LIGHT_SOURCE_INDICES_SUPPORTED                          : constant := 8;
  MAX_LIGHT_SOURCE_TYPES_SUPPORTED                            : constant := 4;
  MAX_PARAMETRIC_SURFACE_INDICES_SUPPORTED                    : constant := 6;
  MAX_PARAMETRIC_SURFACE_CHARACTERISTICS_SUPPORTED : constant := 2;
  MAX_POLYLINE_SHADING_METHODS_SUPPORTED                      : constant := 2;
  MAX_REFLECTANCE_MODELS_SUPPORTED                            : constant := 6;
  MAX_REFLECTANCE_INDICES_SUPPORTED                           : constant := 20;
  MAX_REFLECTANCE_PROPERTIES_TYPES_SUPPORTED                  : constant := 2;
  MAX_SURFACE_APPROX_CRITERIA_TYPES_SUPPORTED                 : constant := 4;
  MAX_VERTICES_SUPPORTED                                      : constant := 4096;

end PHIGS_CONFIGURATION;
```

```
-- PHIGS List Utilities Generic Package

    with PHIGS_CONFIGURATION;

    generic

        type ELEMENT_TYPE is private;

        MAX_LIST_SIZE : NATURAL := PHIGS_CONFIGURATION.DEFAULT_LIST_SIZE;

    package PHIGS_LIST_UTILITIES is

        subtype LIST_SIZE is NATURAL range 0..MAX_LIST_SIZE;

        type LIST_OF (SIZE : LIST_SIZE := 0) is private;

        type LIST_VALUES is array (POSITIVE range <>) of ELEMENT_TYPE;

        function NULL_LIST return LIST_OF;

          -- This function returns an empty LIST_OF object.  This list is
          -- intended primarily for use by implementors.

        function SIZE_OF_LIST (LIST : in LIST_OF) return NATURAL;

          -- This function returns the number of element type values stored
          -- in the list object.

        function IS_IN_LIST (ELEMENT : in ELEMENT_TYPE;
                             LIST    : in LIST_OF) return BOOLEAN;

        -- This function returns the value TRUE if the element parameter
        -- value is in the list object.  Otherwise it returns the value
        -- FALSE.

        function LIST_ELEMENT ( INDEX : in POSITIVE;
                                LIST  : in  LIST_OF) return ELEMENT_TYPE;

        -- This function returns the element value in the list object that
        -- has an associated index value equal to the index parameter.  The
        -- PHIGS_ERROR 2502 is generated if the index parameter exceeds the
        -- current size of the list parameter object.

        function LIST (VALUES : in LIST_VALUES) return LIST_OF;

        -- This function returns a valid LIST_OF object.  If the VALUES
        -- parameter is a null array, an empty LIST_OF object is returned.
        -- If the values parameter is not null, this function returns a
```

```
-- LIST_OF object containing all the values in the VALUES parameter.
-- The PHIGS_ERROR 2502 is generated if the number of element values
-- exceeds the specified maximum size of the LIST_OF object.

procedure ADD_TO_LIST ( ELEMENT : in ELEMENT_TYPE;
                        LIST    : in out LIST_OF);

-- This procedure stores the element parameter value in the list
-- parameter object, and increases the size of the list object
-- by one. An index value equal to the incremented list size
-- is associated with the stored element value. The ADD_TO_LIST
-- procedure will generate PHIGS_ERROR 2502 if it is called when
-- the list parameter has a size equal to the specified maximum
-- size. If desired, the user can ensure duplicate values are not
-- stored. This is accomplised by calling ADD_TO_LIST with a
-- particular element value only if the function IS_IN_LIST returns
-- false for that element value.


procedure DELETE_FROM_LIST ( ELEMENT : in ELEMENT_TYPE;
                             LIST    : in out LIST_OF);

-- If the list parameter object does not contain the element
-- parameter value, this procedure has no effect. Otherwise, the
-- first occurrence of the element value is deleted. The size of
-- the list object is decreased by one, and the indices associated
-- with the remaining element values are adjusted so that the indices
-- begin at one and increment in steps of one. If desired, the user
-- can delete all occurrences of an element value. This is accomplished
-- by calling DELETE_FROM_LIST repeatedly with a particular element
-- value while the function IS_IN_LIST returns TRUE for that value.

private

-- The declaration of the LIST_OF type is implementation dependent.
-- However, the operations implicitly declared by the LIST_OF declaration,
-- including both assignment and the predefined comparison for equality
-- and inequality, shall produce the correct results. This requirement
-- precludes the use of access types for the implementation of the
-- LIST_OF type. The recommended implementation is given below:

type LIST_OF (SIZE : LIST_SIZE := 0) is
  record
    ELEMENTS : LIST_VALUES(1..SIZE);
  end record;
--
-- Note that declaring unconstrained LIST_OF objects by using the default
-- discriminant value allows dynamic modification of the size of the
-- element array.
```

```
end PHIGS_LIST_UTILITIES;


package body PHIGS_LIST_UTILITIES is

-- This package body contains stubs for each of the subprograms
-- contained in the PHIGS_LIST_UTILITIES package.  The purpose of
-- these stubs is to show compilability for the PHIGS/Ada binding.
-- An implementation would replace these stubs with code appropriate
-- to implement the functionality.


   function NULL_LIST return LIST_OF is
     EMPTY_LIST : LIST_OF;
   begin
     return EMPTY_LIST;
   end NULL_LIST;

   function SIZE_OF_LIST (LIST : in LIST_OF) return NATURAL is
   begin
     return 0;
   end SIZE_OF_LIST;

   function IS_IN_LIST ( ELEMENT : in ELEMENT_TYPE;
                         LIST    : in LIST_OF) return BOOLEAN is
   begin
     return FALSE;
   end IS_IN_LIST;

   function LIST_ELEMENT ( INDEX : in POSITIVE;
                           LIST  : in  LIST_OF) return ELEMENT_TYPE is
     EXAMPLE_ELEMENT : ELEMENT_TYPE;
   begin
     return EXAMPLE_ELEMENT;
   end LIST_ELEMENT;

   function LIST (VALUES : in LIST_VALUES) return LIST_OF is
   begin
     return NULL_LIST;
   end LIST;


   procedure ADD_TO_LIST ( ELEMENT : in ELEMENT_TYPE;
                           LIST    : in out LIST_OF) is
   begin
     null;
   end ADD_TO_LIST;

   procedure DELETE_FROM_LIST ( ELEMENT : in ELEMENT_TYPE;
```

```
                                        LIST        : in out LIST_OF) is
    begin
      null;
    end DELETE_FROM_LIST;

end PHIGS_LIST_UTILITIES;
```

```
-- PHIGS Name Set Facility Package

with PHIGS_CONFIGURATION, PHIGS_LIST_UTILITIES;

package PHIGS_NAME_SET_FACILITY is

   -- PHIGS_CONFIGURATION.MAX_NAME_SUPPORTED

   -- An implementation dependent constant which defines the maximum
   -- name supported. The minimum value allowed by the PHIGS
   -- specification is 63 thus supporting 64 classes. A larger
   -- value is strongly encouraged.

   type NAME_SET is private;

   -- Defines a name set data element.

   -- NAME_SET_ACCEPTANCE

   type NAME_SET_ACCEPTANCE is (REJECTED, ACCEPTED);

   -- Used to indicate the results of applying a name set
   -- against a filter pair.

   -- NAME

   type NAME is range 0..PHIGS_CONFIGURATION.MAX_NAME_SUPPORTED;

   -- Provides for names for each of the name set classes.

   -- NAMES

   package NAMES is new PHIGS_LIST_UTILITIES (NAME);

   -- Provides for lists of names.

   -- NAME_SET_FILTER

   type NAME_SET_FILTER is
     record
       INCLUSION   : NAME_SET;
       EXCLUSION   : NAME_SET;
     end record;

   -- Used to define name set pairs used as filters.

   -- NAME_SET_MEMBERSHIP
```

```
type NAME_SET_MEMBERSHIP is (NON_MEMBER, MEMBER);

-- Provides for indicating whether a name is a member of a
-- name set.

-- Subprograms for Manipulating Name Sets

procedure BUILD_NAME_SET
  (MEMBERS : in   NAMES.LIST_OF;
   SET     : out  NAME_SET);

-- Constructs a name set from a list of names.

procedure BUILD_NAME_SET_UNION
  (LEFT_SET  : in   NAME_SET;
   RIGHT_SET : in   NAME_SET;
   UNION     : out  NAME_SET);

-- Constructs the logical union of two name sets.

procedure BUILD_NAME_SET_INTERSECTION
  (LEFT_SET    : in   NAME_SET;
   RIGHT_SET   : in   NAME_SET;
   INTERSECTION: out  NAME_SET);

-- Constructs the logical intersection of two name sets.

procedure BUILD_NAME_SET_DIFFERENCE
  (LEFT_SET   : in   NAME_SET;
   RIGHT_SET  : in   NAME_SET;
   DIFFERENCE : out  NAME_SET);

-- Constructs the logical difference of two name sets.

procedure COPY_NAME_SET
  (ORIGINAL : in   NAME_SET;
   COPY     : out  NAME_SET);

-- Duplicates the provided name set.

function IS_EQUAL
  (LEFT_SET  : in NAME_SET;
   RIGHT_SET : in NAME_SET)
  return BOOLEAN;

-- Compares two name sets.
```

```
    procedure EXTRACT_NAMES
      (SET          : in    NAME_SET;
       MEMBERS      : out   NAMES.LIST_OF);

    -- Extracts the names from the provided name set.

    procedure ADD_NAMES
      (MEMBERS      : in      NAMES.LIST_OF;
       SET          : in out  NAME_SET);

    -- Adds names to the name set if not already there.

    procedure REMOVE_NAMES
      (MEMBERS      : in      NAMES.LIST_OF;
       SET          : in out  NAME_SET);

    -- Removes names from the name set if they are there.

    function APPLY_FILTER
      (SET      : in NAME_SET;
       FILTER   : in NAME_SET_FILTER)
      return NAME_SET_ACCEPTANCE;

    -- Applies a filter to a name set and indicates whether the name
    -- set passes through the filter.

    function INQUIRE_MEMBERSHIP
      (MEMBER   : in NAME;
       SET      : in NAME_SET)
      return NAME_SET_MEMBERSHIP;

    -- Returns an indication of whether the indicated name is a member
    -- of the provided name set.

    procedure DEALLOCATE
      (OBJECT   : in out NAME_SET);

    -- Deallocates name set objects.

    -- PRIVATE TYPE DEFINITIONS

private

-- The following types define the specifications for private
-- types used in the PHIGS_NAME_SET_FACILITY.  Null records are
-- used soley for the purposes of demonstrating compilability.
-- An implementation would use some more appropriate declaration.
```

```
        type NAME_SET is
          record
            null;
          end record;

    end PHIGS_NAME_SET_FACILITY;
```

```
-- Base types package

with PHIGS_CONFIGURATION;

  package PHIGS_BASE_TYPES is

    -- This package is used to encapsulate the base derived types of PHIGS
    -- since they are used as the parent of several other derived types.
    -- In Ada, if the parent of a derived type is itself a derived type,
    -- then this parent type cannot be declared immediately in the visible
    -- part of the same package.

    use PHIGS_CONFIGURATION;

    -- PHIGS_INTEGER

    type PHIGS_INTEGER is range MIN_INTEGER_VALUE .. MAX_INTEGER_VALUE;

    --  Base type for PHIGS integer types.

    -- PHIGS_NATURAL


    subtype PHIGS_NATURAL is
      PHIGS_INTEGER range 0..PHIGS_INTEGER'last;

    --  Base type for PHIGS natural types.

    -- PHIGS POSITIVE

    subtype PHIGS_POSITIVE is
      PHIGS_INTEGER range 1..PHIGS_INTEGER'last;

    --  Base type for PHIGS positive types.

    -- SCALE_FACTOR

    type SCALE_FACTOR is digits PHIGS_PRECISION;

    -- The type used for unitless scaling factors.

    -- PHIGS_STRING

    type PHIGS_STRING is
      array (PHIGS_INTEGER range <>) of CHARACTER;

    -- Base type for PHIGS string types.
```

```
-- SMALL_NATURAL

subtype SMALL_NATURAL is
  PHIGS_NATURAL range 0..MAX_SMALL_NATURAL;

-- This is a subtype declaration which allows for unconstrained record
-- objects for various record types without causing the exception
-- STORAGE_ERROR to be raised.

-- COLOUR_VALUE_SET_DIMENSION

subtype COLOUR_VALUE_SET_DIMENSION is
        NATURAL range 0..MAX_COLOUR_VALUES_SUPPORTED;

-- Provides for specifying the dimensions of colour value lists and arrays.

-- FACET_SET_DIMENSION

subtype FACET_SET_DIMENSION is
        NATURAL range 0..MAX_FACETS_SUPPORTED;

-- Provides for specifying the dimensions of facet lists and arrays.

-- VERTEX_SET_COUNT

subtype VERTEX_SET_COUNT is
        NATURAL range 0..MAX_VERTEX_SETS_SUPPORTED;

-- Provides for specifying the number of sets in a vertex set.

-- VERTEX_SET_DIMENSION

subtype VERTEX_SET_DIMENSION is
        NATURAL range 0..MAX_VERTICES_SUPPORTED;

-- Provides for specifying the dimensions of vertex lists and arrays.

-- SPLINE_ORDER

type SPLINE_ORDER is new POSITIVE;

-- Provides for specifying orders for spline curves and surfaces.

-- SPLINE_RATIONALITY

type SPLINE_RATIONALITY is (RATIONAL, NON_RATIONAL);

-- Provides for specifying the rationality of splines.
```

```
end PHIGS_BASE_TYPES;
```

```
-- The PHIGS_COLOUR_TYPES Package

generic

   type COLOUR_VALUE is private;

   type HOMOGENEOUS_COLOUR_VALUE is private;

package PHIGS_COLOUR_TYPES is


   type COLOUR_VALUE_LIST is
     array (COLOUR_VALUE_SET_DIMENSION range <>)
       of COLOUR_VALUE;

   type ACCESS_COLOUR_VALUE_LIST is access COLOUR_VALUE_LIST;

   type HOMOGENEOUS_COLOUR_VALUE_LIST is
     array (COLOUR_VALUE_SET_DIMENSION range <>)
       of HOMOGENEOUS_COLOUR_VALUE;

   type ACCESS_HOMOGENEOUS_COLOUR_VALUE_LIST is
     access HOMOGENEOUS_COLOUR_VALUE_LIST;

   type LIST_OF_COLOUR_VALUE_LIST is
     array (COLOUR_VALUE_SET_DIMENSION range <>)
       of ACCESS_COLOUR_VALUE_LIST;

   type ACCESS_LIST_OF_COLOUR_VALUE_LIST is
     access LIST_OF_COLOUR_VALUE_LIST;

   type COLOUR_VALUE_ARRAY is
     array ( COLOUR_VALUE_SET_DIMENSION range <>,
             COLOUR_VALUE_SET_DIMENSION range <> )
       of COLOUR_VALUE;

   type ACCESS_COLOUR_VALUE_ARRAY is access COLOUR_VALUE_ARRAY;

   type HOMOGENEOUS_COLOUR_VALUE_ARRAY is
     array ( COLOUR_VALUE_SET_DIMENSION range <>,
             COLOUR_VALUE_SET_DIMENSION range <> )
       of HOMOGENEOUS_COLOUR_VALUE;

   type ACCESS_HOMOGENEOUS_COLOUR_VALUE_ARRAY is
     access HOMOGENEOUS_COLOUR_VALUE_ARRAY;

   type CONTROL_POINT_LIST
         (TYPE_OF_RATIONALITY : SPLINE_RATIONALITY;
          LENGTH              : COLOUR_VALUE_SET_DIMENSION) is
```

```
        record
          case TYPE_OF_RATIONALITY is

            when RATIONAL =>
              RATIONAL_POINTS : HOMOGENEOUS_COLOUR_VALUE_LIST(1..LENGTH);

            when NON_RATIONAL =>
              NON_RATIONAL_POINTS : COLOUR_VALUE_LIST(1..LENGTH);

          end case;
        end record;

      type ACCESS_CONTROL_POINT_LIST is access CONTROL_POINT_LIST;

      type CONTROL_POINT_ARRAY
            (TYPE_OF_RATIONALITY : SPLINE_RATIONALITY;
             LENGTH_U            : COLOUR_VALUE_SET_DIMENSION;
             LENGTH_V            : COLOUR_VALUE_SET_DIMENSION) is
        record
          case TYPE_OF_RATIONALITY is

            when RATIONAL =>
              RATIONAL_POINTS : HOMOGENEOUS_COLOUR_VALUE_ARRAY
                                  (1..LENGTH_U, 1..LENGTH_V);

            when NON_RATIONAL =>
              NON_RATIONAL_POINTS : COLOUR_VALUE_ARRAY
                                      (1..LENGTH_U, 1..LENGTH_V);

          end case;
        end record;

      type ACCESS_CONTROL_POINT_ARRAY is access CONTROL_POINT_ARRAY;

    end PHIGS_COLOUR_TYPES;
```

```
-- The PHIGS_STANDARD_TYPES Package

with PHIGS_CONFIGURATION, PHIGS_BASE_TYPES;


package PHIGS_STANDARD_TYPES is


-- This package contains all the standard data type definitions required by
-- the PHIGS coordinate system generic package.

-- Configuration names

use PHIGS_CONFIGURATION, PHIGS_BASE_TYPES;

    --  Make PHIGS_BASE_TYPES visible externally.

    subtype PHIGS_INTEGER is PHIGS_BASE_TYPES.PHIGS_INTEGER;
    subtype PHIGS_NATURAL is PHIGS_BASE_TYPES.PHIGS_NATURAL;
    subtype PHIGS_POSITIVE is PHIGS_BASE_TYPES.PHIGS_POSITIVE;
    subtype PHIGS_STRING is PHIGS_BASE_TYPES.PHIGS_STRING;
    subtype SCALE_FACTOR is PHIGS_BASE_TYPES.SCALE_FACTOR;
    subtype SMALL_NATURAL is BASE_TYPE.SMALL_NATURAL;
    subtype COLOUR_VALUE_SET_DIMENSION is
                PHIGS_BASE_TYPES.COLOUR_VALUE_SET_DIMENSION;
    subtype FACET_SET_DIMENSION is PHIGS_BASE_TYPES.FACET_SET_DIMENSION;
    subtype VERTEX_SET_COUNT is PHIGS_BASE_TYPES.VERTEX_SET_COUNT;
    subtype                                VERTEX_SET_DIMENSION                                is
PHIGS_BASE_TYPES.VERTEX_SET_DIMENSION;
    subtype SPLINE_ORDER is PHIGS_BASE_TYPES.SPLINE_ORDER;
    subtype SPLINE_RATIONALITY is PHIGS_BASE_TYPES.SPLINE_RATIONALITY;

    -- DATA_VALUE

    type DATA_VALUE is digits PHIGS_PRECISION;

    -- Provides for data mapping input values.

    -- DATA_VALUE_INDEX

    type DATA_VALUE_INDEX is
            new PHIGS_POSITIVE range 1 .. MAX_DATA_VALUES_PER_VERTEX;

    -- Provides for selecting from lists of data mapping input values.

    -- DATA_VALUE_SET

    type DATA_VALUE_SET is
      array (DATA_VALUE_INDEX range <>) of DATA_VALUE;
```

```
-- Provides for arrays (ordered lists) of data mapping input values.


-- ACCESS_DATA_VALUE_SET

type ACCESS_DATA_VALUE_SET is access DATA_VALUE_SET;

-- Provides for pointers to sets of data mapping data values.
-- DATA_MAPPING_INDEX

type DATA_MAPPING_INDEX is new PHIGS_NATURAL;

-- Provides for index values for data mapping representations.
-- LIST_OF_DATA_VALUE_SET

type LIST_OF_DATA_VALUE_SET is
        array ( VERTEX_SET_DIMENSION range <>,
                DATA_VALUE_INDEX range <>) of DATA_VALUE;

-- Provides for lists of data mapping data value sets.  These will be used
-- various output primitives.  The first index refers to the number of data value lists
-- and the second index to the individual data values.

-- ARRAY_OF_DATA_VALUE_SET

type ARRAY_OF_DATA_VALUE_SET is
        array ( VERTEX_SET_DIMENSION range <>,
                VERTEX_SET_DIMENSION range <>,
                DATA_VALUE_INDEX range <>) of DATA_VALUE;

-- Provides for two-dimensional arrays of data mapping data value sets.  These
-- will be used with "Quadrilateral Mesh with Data" primitives.  The first index
-- refers to the width of the array, the second index to the height, and the third
-- index to the individual data values.
-- COLOUR_COMPONENTS

type COLOUR_COMPONENTS is range 1..MAX_COLOUR_COEFFICIENTS;

-- Defines a type for the size of colour component arrays.

-- COLOUR_INDEX

type COLOUR_INDEX is new PHIGS_NATURAL;

-- Indices into colour tables are of this type.

-- COLOUR_MODEL
```

```
type COLOUR_MODEL is new PHIGS_INTEGER;

-- Indicates the colour models available in PHIGS.

-- The following constants define the colour models specified by PHIGS:

INDIRECT   : constant COLOUR_MODEL := 0;
RGB        : constant COLOUR_MODEL := 1;
CIELUV     : constant COLOUR_MODEL := 2;
HSV        : constant COLOUR_MODEL := 3;
HLS        : constant COLOUR_MODEL := 4;

-- COLOUR_COEFFICIENT

type COLOUR_COEFFICIENT is digits PHIGS_PRECISION;

-- Defines a general type for colour components.

-- COLOUR_COEFFICIENT_ARRAY

type COLOUR_COEFFICIENT_ARRAY is
  array (COLOUR_COMPONENTS) of COLOUR_COEFFICIENT;

-- Defines an array type for colour components.

-- INTENSITY

subtype INTENSITY is COLOUR_COEFFICIENT range 0.0..1.0;

-- Defines the restricted range of colour components required
-- by some colour models.

-- CIELUV COLOUR VALUE

type CIELUV_COLOUR_VALUE is
  record
    L_STAR_CIE : COLOUR_COEFFICIENT;
    U_STAR_CIE : COLOUR_COEFFICIENT;
    V_STAR_CIE : COLOUR_COEFFICIENT;
  end record;

-- Provides for specification of CIELUV colours.

-- CIELUV HOMOGENEOUS COLOUR VALUE
```

```
type CIELUV_HOMOGENEOUS_COLOUR_VALUE is
  record
    L_STAR_CIE : COLOUR_COEFFICIENT;
    U_STAR_CIE : COLOUR_COEFFICIENT;
    V_STAR_CIE : COLOUR_COEFFICIENT;
    W_CIE      : COLOUR_COEFFICIENT;
  end record;

-- Provides for specification of homogeneous CIELUV colours for use in
-- rational coloursplines.

-- CIELUV COLOUR PACKAGE

package CIELUV_TYPES is
  new PHIGS_COLOUR_TYPES( CIELUV_COLOUR_VALUE,
                          CIELUV_HOMOGENEOUS_COLOUR_VALUE);

-- Provides for CIELUV colour types.

-- GENERIC COLOUR PACKAGE

package GENERIC_COLOUR_TYPES is
  new PHIGS_COLOUR_TYPES( COLOUR_COEFFICIENT_ARRAY,
                          COLOUR_COEFFICIENT_ARRAY);

-- Provides for generic (unsupported) colour types.

-- HLS COLOUR VALUE

type HLS_COLOUR_VALUE is
  record
    HUE_HLS        : INTENSITY;
    LIGHTNESS_HLS  : INTENSITY;
    SATURATION_HLS : INTENSITY;
  end record;

-- Provides for specification of HLS colours.

-- HLS HOMOGENEOUS COLOUR VALUE
```

```
type HLS_HOMOGENEOUS_COLOUR_VALUE is
  record
    HUE_HLS          : COLOUR_COEFFICIENT;
    LIGHTNESS_HLS    : COLOUR_COEFFICIENT;
    SATURATION_HLS   : COLOUR_COEFFICIENT;
    W_HLS            : COLOUR_COEFFICIENT;
  end record;

-- Provides for specification of homogeneous HLS colours for use in
-- rational coloursplines.

-- HLS COLOUR PACKAGE

package HLS_TYPES is
  new PHIGS_COLOUR_TYPES( HLS_COLOUR_VALUE,
                          HLS_HOMOGENEOUS_COLOUR_VALUE);

-- Provides for HLS colour types.

-- HSV_COLOUR_VALUE

type HSV_COLOUR_VALUE is
  record
    HUE_HSV          : INTENSITY;
    SATURATION_HSV   : INTENSITY;
    VALUE_HSV        : INTENSITY;
  end record;

-- Provides for specification of HSV colours.

-- HSV HOMOGENEOUS COLOUR VALUE

type HSV_HOMOGENEOUS_COLOUR_VALUE is
  record
    HUE_HSV          : COLOUR_COEFFICIENT;
    SATURATION_HSV   : COLOUR_COEFFICIENT;
    VALUE_HSV        : COLOUR_COEFFICIENT;
    W_HSV            : COLOUR_COEFFICIENT;
  end record;

-- Provides for specification of homogeneous HSV colours for use in
-- rational coloursplines.

-- HSV COLOUR PACKAGE

package HSV_TYPES is
  new PHIGS_COLOUR_TYPES( HSV_COLOUR_VALUE,
                          HSV_HOMOGENEOUS_COLOUR_VALUE);
```

```
-- Provides for HSV colour types.

-- INDIRECT HOMOGENEOUS COLOUR VALUE

type INDIRECT_HOMOGENEOUS_COLOUR_VALUE is
  record
    INDEX_INDIRECT : COLOUR_INDEX;
    W_INDIRECT     : COLOUR_COEFFICIENT;
  end record;

-- Provides for specification of homogeneous indirect colours for use in rational
-- coloursplines.

-- INDIRECT COLOUR PACKAGE

package INDIRECT_TYPES is
  new PHIGS_COLOUR_TYPES( COLOUR_INDEX,
                          INDIRECT_HOMOGENEOUS_COLOUR_VALUE);

-- Provides for INDIRECT colour types.

-- RGB_COLOUR_VALUE

type RGB_COLOUR_VALUE is
  record
    RED_RGB   : INTENSITY;
    GREEN_RGB : INTENSITY;
    BLUE_RGB  : INTENSITY;
  end record;

-- Provides for specification of RGB colours.

-- RGB HOMOGENEOUS COLOUR VALUE

type RGB_HOMOGENEOUS_COLOUR_VALUE is
  record
    RED_RGB   : COLOUR_COEFFICIENT;
    GREEN_RGB : COLOUR_COEFFICIENT;
    BLUE_RGB  : COLOUR_COEFFICIENT;
    W_RGB     : COLOUR_COEFFICIENT;
  end record;

-- Provides for specification of homogeneous RGB colours for use in rational coloursplines.

-- RGB COLOUR PACKAGE
```

```
package RGB_TYPES is
  new PHIGS_COLOUR_TYPES( RGB_COLOUR_VALUE,
                          RGB_HOMOGENEOUS_COLOUR_VALUE);

-- Provides for RGB colour types.

-- GENERAL_COLOUR

type GENERAL_COLOUR
  (MODEL : COLOUR_MODEL := RGB) is
    record
      case MODEL is

        when INDIRECT =>
          INDEX              : COLOUR_INDEX;

        when RGB =>
          COLOUR_RGB         : RGB_COLOUR_VALUE;

        when CIELUV =>
          COLOUR_CIELUV      : CIELUV_COLOUR_VALUE;

        when HSV =>
          COLOUR_HSV         : HSV_COLOUR_VALUE;

        when HLS =>
          COLOUR_HLS         : HLS_COLOUR_VALUE;

        when others =>
          COLOUR_GENERIC     : COLOUR_COEFFICIENT_ARRAY;

      end case;
    end record;

-- Provides for specifying values for colours dependent upon colour models.  Additional
-- variants may be included when additional registered or unregistered colour models are
-- supported by an implementation.

-- COLOUR_VALUE_ARRAY

type COLOUR_VALUE_ARRAY
   (MODEL      : COLOUR_MODEL := RGB;
    LENGTH_M : COLOUR_VALUE_SET_DIMENSION := 1;
    LENGTH_N  : COLOUR_VALUE_SET_DIMENSION := 1)  is
     record
       case MODEL is
```

```
            when INDIRECT =>
              COLOURS_INDIRECT : INDIRECT_TYPES.COLOUR_VALUE_ARRAY
                                   (1..LENGTH_M, 1..LENGTH_N);

            when RGB =>
              COLOURS_RGB : RGB_TYPES.COLOUR_VALUE_ARRAY
                              (1..LENGTH_M, 1..LENGTH_N);

            when CIELUV =>
              COLOURS_CIELUV : CIELUV_TYPES.COLOUR_VALUE_ARRAY
                                 (1..LENGTH_M, 1..LENGTH_N);

            when HSV =>
              COLOURS_HSV  : HSV_TYPES.COLOUR_VALUE_ARRAY
                               (1..LENGTH_M, 1..LENGTH_N);

            when HLS =>
              COLOURS_HLS : HLS_TYPES.COLOUR_VALUE_ARRAY
                              (1..LENGTH_M, 1..LENGTH_N);

            when others =>
              COLOURS_GENERIC : GENERIC_COLOUR_TYPES.COLOUR_VALUE_ARRAY
                                  (1..LENGTH_M, 1..LENGTH_N);

        end case;
      end record;

-- Provides for specification of two-dimensional arrays of colour values all of which
-- are the same colour model.
-- COLOUR_VALUE_LIST

type COLOUR_VALUE_LIST
  (MODEL  : COLOUR_MODEL := RGB;
   LENGTH : COLOUR_VALUE_SET_DIMENSION := 0) is
    record
      case MODEL is

            when INDIRECT =>
              COLOURS_INDIRECT : INDIRECT_TYPES.COLOUR_VALUE_LIST(1..LENGTH);

            when RGB =>
              COLOURS_RGB : RGB_TYPES.COLOUR_VALUE_LIST(1..LENGTH);

            when CIELUV =>
              COLOURS_CIELUV  : CIELUV_TYPES.COLOUR_VALUE_LIST(1..LENGTH);

            when HSV =>
              COLOURS_HSV : HSV_TYPES.COLOUR_VALUE_LIST(1..LENGTH);
```

```
            when HLS =>
              COLOURS_HLS : HLS_TYPES.COLOUR_VALUE_LIST(1..LENGTH);

            when others =>
              COLOURS_GENERIC : GENERIC_COLOUR_TYPES.COLOUR_VALUE_LIST
                                        (1..LENGTH);

        end case;
      end record;

-- Provides for specification of lists of colour values all of which are the same colour model.
-- LIST_OF_COLOUR_VALUE_LIST

type LIST_OF_COLOUR_VALUE_LIST
        (MODEL    : COLOUR_MODEL := RGB;
         LENGTH   : COLOUR_VALUE_SET_DIMENSION := 3) is
    record
      case MODEL is

        when INDIRECT =>
          COLOURS_INDIRECT
                : INDIRECT_TYPES.LIST_OF_COLOUR_VALUE_LIST(1..LENGTH);

        when RGB =>
          COLOURS_RGB
                : RGB_TYPES.LIST_OF_COLOUR_VALUE_LIST(1..LENGTH);

        when CIELUV =>
          COLOURS_CIELUV
                : CIELUV_TYPES.LIST_OF_COLOUR_VALUE_LIST(1..LENGTH);

        when HSV =>
          COLOURS_HSV
                : HSV_TYPES.LIST_OF_COLOUR_VALUE_LIST(1..LENGTH);

        when HLS =>
          COLOURS_HLS
                : HLS_TYPES.LIST_OF_COLOUR_VALUE_LIST(1..LENGTH);

        when others =>
            COLOURS_GENERIC
                : GENERIC_COLOUR_TYPES.LIST_OF_COLOUR_VALUE_LIST
                                        (1..LENGTH);
```

```
      end case;
    end record;

-- Provides for specification of lists of lists of colour values all of which are the same
-- colour model.
-- FACET_DATA_FLAG

type FACET_DATA_FLAG is ( FACET_NONE,
                          FACET_COLOUR,
                          FACET_NORMAL,
                          FACET_DATA,
                          FACET_COLOUR_NORMAL,
                          FACET_COLOUR_DATA,
                          FACET_NORMAL_DATA,
                          FACET_COLOUR_NORMAL_DATA);

-- Provides for various types of facet data flags.

-- VERTEX_DATA_FLAG

type VERTEX_DATA_FLAG is ( COORDINATES,
                           COORDINATES_COLOUR,
                           COORDINATES_NORMAL,
                           COORDINATES_DATA,
                           COORDINATES_COLOUR_NORMAL,
                           COORDINATES_COLOUR_DATA,
                           COORDINATES_NORMAL_DATA,
                           COORDINATES_COLOUR_NORMAL_DATA);

-- Provides identifiers for various types of vertex information.

-- VERTEX_COLOUR_FLAG

subtype VERTEX_COLOUR_FLAG is
        VERTEX_DATA_FLAG range COORDINATES .. COORDINATES_COLOUR;

-- Provides identifiers for various types of vertex information used in polyline sets.

end PHIGS_STANDARD_TYPES;
```

```
-- The PHIGS_COORDINATE_SYSTEM Generic Package

with  PHIGS_CONFIGURATION,
      PHIGS_BASE_TYPES,
      PHIGS_STANDARD_TYPES;

generic

    type COORDINATE_COMPONENT_TYPE is digits <>;

package PHIGS_COORDINATE_SYSTEM is

    use PHIGS_CONFIGURATION, PHIGS_BASE_TYPES, PHIGS_STANDARD_TYPES;

    type POINT_4 is
      record
        X : COORDINATE_COMPONENT_TYPE;
        Y : COORDINATE_COMPONENT_TYPE;
        Z : COORDINATE_COMPONENT_TYPE;
        W : COORDINATE_COMPONENT_TYPE;
      end record;

    type POINT_3 is
      record
        X : COORDINATE_COMPONENT_TYPE;
        Y : COORDINATE_COMPONENT_TYPE;
        Z : COORDINATE_COMPONENT_TYPE;
      end record;

    type POINT_2 is
      record
        X : COORDINATE_COMPONENT_TYPE;
        Y : COORDINATE_COMPONENT_TYPE;
      end record;

    type POINT_LIST_4 is array (POSITIVE range <>) of POINT_4;

    type POINT_LIST_3 is array (POSITIVE range <>) of POINT_3;

    type POINT_LIST_2 is array (POSITIVE range <>) of POINT_2;

    type ACCESS_POINT_LIST_4 is access POINT_LIST_4;

    type ACCESS_POINT_LIST_3 is access POINT_LIST_3;
```

```
type ACCESS_POINT_LIST_2 is access POINT_LIST_2;

type LIST_OF_POINT_LIST_4 is
  array (POSITIVE range <>) of ACCESS_POINT_LIST_4;

type LIST_OF_POINT_LIST_3 is array (POSITIVE range <>) of
  ACCESS_POINT_LIST_3;

type LIST_OF_POINT_LIST_2 is array (POSITIVE range <>) of
  ACCESS_POINT_LIST_2;

type ACCESS_LIST_OF_POINT_LIST_4 is access LIST_OF_POINT_LIST_4;

type ACCESS_LIST_OF_POINT_LIST_3 is access LIST_OF_POINT_LIST_3;

type ACCESS_LIST_OF_POINT_LIST_2 is access LIST_OF_POINT_LIST_2;

type POINT_ARRAY_4 is
  array (POSITIVE range <>, POSITIVE range <>) of POINT_4;

type ACCESS_POINT_ARRAY_4 is access POINT_ARRAY_4;

type POINT_ARRAY_3 is
  array (POSITIVE range <>, POSITIVE range <>) of POINT_3;

type ACCESS_POINT_ARRAY_3 is access POINT_ARRAY_3;

type POINT_ARRAY_2 is
  array (POSITIVE range <>, POSITIVE range <>) of POINT_2;

type ACCESS_POINT_ARRAY_2 is access POINT_ARRAY_2;

type CONTROL_POINT_LIST_3
        (TYPE_OF_RATIONALITY : SPLINE_RATIONALITY;
         LENGTH              : VERTEX_SET_DIMENSION) is
  record
    case TYPE_OF_RATIONALITY is

      when RATIONAL =>
        RATIONAL_POINTS : POINT_LIST_4(1..LENGTH);

      when NON_RATIONAL =>
        NON_RATIONAL_POINTS : POINT_LIST_3(1..LENGTH);

    end case;
  end record;

type ACCESS_CONTROL_POINT_LIST_3 is access CONTROL_POINT_LIST_3;

type CONTROL_POINT_LIST_2
```

```
        (TYPE_OF_RATIONALITY : SPLINE_RATIONALITY;
         LENGTH              : VERTEX_SET_DIMENSION) is
  record
    case TYPE_OF_RATIONALITY is

      when RATIONAL =>
        RATIONAL_POINTS : POINT_LIST_3(1..LENGTH);

      when NON_RATIONAL =>
        NON_RATIONAL_POINTS : POINT_LIST_2(1..LENGTH);

    end case;
  end record;

type ACCESS_CONTROL_POINT_LIST_2 is access CONTROL_POINT_LIST_2;

type CONTROL_POINT_ARRAY_3
        (TYPE_OF_RATIONALITY : SPLINE_RATIONALITY;
         LENGTH_U            : VERTEX_SET_DIMENSION;
         LENGTH_V            : VERTEX_SET_DIMENSION) is
  record
    case TYPE_OF_RATIONALITY is

      when RATIONAL =>
        RATIONAL_POINTS : POINT_ARRAY_4(1..LENGTH_U, 1..LENGTH_V);

      when NON_RATIONAL =>
        NON_RATIONAL_POINTS : POINT_ARRAY_3(1..LENGTH_U, 1..LENGTH_V);

    end case;
  end record;

type ACCESS_CONTROL_POINT_ARRAY_3 is access CONTROL_POINT_ARRAY_3;

type VECTOR_3 is new POINT_3;

type VECTOR_PAIR_3 is array (1..2) of VECTOR_3;

type VECTOR_2 is new POINT_2;

type VECTOR_PAIR_2 is array (1..2) of VECTOR_2;

type VECTOR_ARRAY_3 is
  array (POSITIVE range <>, POSITIVE range <>) of VECTOR_3;

type ACCESS_VECTOR_ARRAY_3 is access VECTOR_ARRAY_3;

type VECTOR_LIST_3 is
  array (POSITIVE range <>) of VECTOR_3;
```

```
type ACCESS_VECTOR_LIST_3 is access VECTOR_LIST_3;

type RECTANGULAR_REGION_3 is
  record
    XMIN  : COORDINATE_COMPONENT_TYPE;
    XMAX : COORDINATE_COMPONENT_TYPE;
    YMIN  : COORDINATE_COMPONENT_TYPE;
    YMAX : COORDINATE_COMPONENT_TYPE;
    ZMIN  : COORDINATE_COMPONENT_TYPE;
    ZMAX : COORDINATE_COMPONENT_TYPE;
  end record;

type RECTANGULAR_REGION_2 is
  record
    XMIN  : COORDINATE_COMPONENT_TYPE;
    XMAX : COORDINATE_COMPONENT_TYPE;
    YMIN  : COORDINATE_COMPONENT_TYPE;
    YMAX : COORDINATE_COMPONENT_TYPE;
  end record;

type HALF_SPACE_3 is
  record
    REFERENCE_POSITION   : POINT_3;
    ACCEPTANCE_NORMAL  : VECTOR_3;
  end record;

type HALF_SPACE_2 is
  record
    REFERENCE_POSITION   : POINT_2;
    ACCEPTANCE_NORMAL  : VECTOR_2;
  end record;

type HALF_SPACE_LIST_3 is
  array (POSITIVE range <>) of HALF_SPACE_3;

type HALF_SPACE_LIST_2 is
  array (POSITIVE range <>) of HALF_SPACE_2;

type ACCESS_HALF_SPACE_LIST_3 is access HALF_SPACE_LIST_3;

type ACCESS_HALF_SPACE_LIST_2 is access HALF_SPACE_LIST_2;

type MAGNITUDE_BASE_TYPE is digits PHIGS_PRECISION;

subtype MAGNITUDE is MAGNITUDE_BASE_TYPE
    range 0.0 .. COORDINATE_COMPONENT_TYPE'SAFE_LARGE;

subtype RELATIVE_MAGNITUDE is MAGNITUDE range 0.0 .. 1.0;

type SIZE_3 is
```

```
    record
      XAXIS : MAGNITUDE;
      YAXIS : MAGNITUDE;
      ZAXIS : MAGNITUDE;
    end record;

  type SIZE_2 is
    record
      XAXIS : MAGNITUDE;
      YAXIS : MAGNITUDE;
    end record;

  type MAGNITUDE_LIST_ARRAY is
          array (SMALL_NATURAL range <>) of MAGNITUDE;

  type MAGNITUDE_LIST
           (LENGTH : SMALL_NATURAL := 1) is
    record
      MAGNITUDES : MAGNITUDE_LIST_ARRAY(1..LENGTH);
    end record;

  type ACCESS_MAGNITUDE_LIST is access MAGNITUDE_LIST;

  type ARRAY_OF_MAGNITUDE_LISTS is
    array ( POSITIVE range <>,
            POSITIVE range <>) of ACCESS_MAGNITUDE_LIST;

  type ACCESS_ARRAY_OF_MAGNITUDE_LISTS is
    access ARRAY_OF_MAGNITUDE_LISTS;

  type RANGE_OF_MAGNITUDES is
    record
      MIN  : MAGNITUDE;
      MAX : MAGNITUDE;
    end record;

  type CURVE_GEOMETRYSPLINE_3
         (RATIONALITY : SPLINE_RATIONALITY;
          KNOT_COUNT : SMALL_NATURAL;
          LENGTH          : VERTEX_SET_DIMENSION) is
    record
      ORDER                : SPLINE_ORDER;
      KNOTS                : MAGNITUDE_LIST(KNOT_COUNT);
      CONTROL_POINTS : CONTROL_POINT_LIST_3(RATIONALITY, LENGTH);
    end record;

  type ACCESS_CURVE_GEOMETRYSPLINE_3 is
    access CURVE_GEOMETRYSPLINE_3;
```

```
type CURVE_GEOMETRYSPLINE_2
      (RATIONALITY : SPLINE_RATIONALITY;
       KNOT_COUNT : SMALL_NATURAL;
       LENGTH      : VERTEX_SET_DIMENSION) is
  record
    ORDER          : SPLINE_ORDER;
    KNOTS          : MAGNITUDE_LIST(KNOT_COUNT);
    CONTROL_POINTS : CONTROL_POINT_LIST_2(RATIONALITY, LENGTH);
  end record;

type ACCESS_CURVE_GEOMETRYSPLINE_2 is
  access CURVE_GEOMETRYSPLINE_2;

type SURFACE_GEOMETRYSPLINE
      (RATIONALITY     : SPLINE_RATIONALITY;
       KNOT_COUNT_U    : SMALL_NATURAL;
       KNOT_COUNT_V    : SMALL_NATURAL;
       LENGTH_U        : VERTEX_SET_DIMENSION;
       LENGTH_V        : VERTEX_SET_DIMENSION) is
  record
    U_ORDER          : SPLINE_ORDER;
    V_ORDER          : SPLINE_ORDER;
    U_KNOTS          : MAGNITUDE_LIST(KNOT_COUNT_U);
    V_KNOTS          : MAGNITUDE_LIST(KNOT_COUNT_V);
    CONTROL_POINTS : CONTROL_POINT_ARRAY_3( RATIONALITY,
                                            LENGTH_U,
                                            LENGTH_V);
  end record;

type ACCESS_SURFACE_GEOMETRYSPLINE is
  access SURFACE_GEOMETRYSPLINE;

type FACET_COLOUR_ITEM (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
                        MODEL               : COLOUR_MODEL) is
  record
    case FACET_DATA_PROVIDED is
      when FACET_COLOUR |
           FACET_COLOUR_NORMAL |
           FACET_COLOUR_DATA |
           FACET_COLOUR_NORMAL_DATA =>
        COLOUR : GENERAL_COLOUR(MODEL);
      when others =>
        null;
    end case;
  end record;

type FACET_NORMAL_ITEM (FACET_DATA_PROVIDED : FACET_DATA_FLAG) is
  record
    case FACET_DATA_PROVIDED is
```

```
        when FACET_NORMAL |
             FACET_COLOUR_NORMAL |
             FACET_NORMAL_DATA |
             FACET_COLOUR_NORMAL_DATA =>
          NORMAL : VECTOR_3;
        when others =>
          null;
      end case;
    end record;

type FACET_DATA_ITEM
       (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
        PER_FACET           : DATA_VALUE_INDEX) is
    record
      case FACET_DATA_PROVIDED is
        when FACET_DATA |
             FACET_COLOUR_DATA |
             FACET_NORMAL_DATA |
             FACET_COLOUR_NORMAL_DATA =>
          DATA_ITEMS : DATA_VALUE_SET(1..PER_FACET);
        when others =>
          null;
      end case;
    end record;

type FACET_DATA_SET
    (FACET_DATA_PROVIDED   : FACET_DATA_FLAG;
     MODEL                 : COLOUR_MODEL;
     PER_FACET             : DATA_VALUE_INDEX) is
      record
        COLOUR       : FACET_COLOUR_ITEM (FACET_DATA_PROVIDED, MODEL);
        NORMAL       : FACET_NORMAL_ITEM (FACET_DATA_PROVIDED);
        DATA_ITEMS   : FACET_DATA_ITEM (FACET_DATA_PROVIDED, PER_FACET);
      end record;

type ACCESS_FACET_DATA_SET is access FACET_DATA_SET;

type FACET_COLOUR_ARRAY
       (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
        WIDTH               : FACET_SET_DIMENSION;
        HEIGHT              : FACET_SET_DIMENSION;
        MODEL               : COLOUR_MODEL) is
    record
      case FACET_DATA_PROVIDED is
```

```
        when FACET_COLOUR |
             FACET_COLOUR_NORMAL |
             FACET_COLOUR_DATA |
             FACET_COLOUR_NORMAL_DATA =>
          COLOURS : COLOUR_VALUE_ARRAY( MODEL,
                                        WIDTH,
                                        HEIGHT);
        when others =>
          null;
      end case;
    end record;

  type FACET_NORMAL_ARRAY
        (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
         WIDTH               : FACET_SET_DIMENSION;
         HEIGHT              : FACET_SET_DIMENSION) is
    record
      case FACET_DATA_PROVIDED is
        when FACET_NORMAL |
             FACET_COLOUR_NORMAL |
             FACET_NORMAL_DATA |
             FACET_COLOUR_NORMAL_DATA =>
          NORMALS : VECTOR_ARRAY_3( 1..WIDTH,
                                    1..HEIGHT);
        when others =>
          null;
      end case;
    end record;

  type FACET_DATA_ARRAY
        (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
         WIDTH               : FACET_SET_DIMENSION;
         HEIGHT              : FACET_SET_DIMENSION;
         PER_FACET           : DATA_VALUE_INDEX) is
    record
      case FACET_DATA_PROVIDED is
```

```
        when FACET_DATA |
             FACET_COLOUR_DATA |
             FACET_NORMAL_DATA |
             FACET_COLOUR_NORMAL_DATA =>
                  DATA_ITEMS : ARRAY_OF_DATA_VALUE_SET(1..WIDTH,
                                                       1..HEIGHT,
                                                       1..PER_FACET);
        when others =>
          null;
      end case;
    end record;

type FACET_DATA_ARRAY_SET
      (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
       WIDTH               : FACET_SET_DIMENSION;
       HEIGHT              : FACET_SET_DIMENSION;
       MODEL               : COLOUR_MODEL;
       PER_FACET           : DATA_VALUE_INDEX) is
    record
      COLOURS     : FACET_COLOUR_ARRAY( FACET_DATA_PROVIDED,
                                        WIDTH,
                                        HEIGHT,
                                        MODEL);
      NORMALS     : FACET_NORMAL_ARRAY( FACET_DATA_PROVIDED,
                                        WIDTH,
                                        HEIGHT);
      DATA_ITEMS : FACET_DATA_ARRAY(  FACET_DATA_PROVIDED,
                                      WIDTH,
                                      HEIGHT,
                                      PER_FACET);
    end record;

type ACCESS_FACET_DATA_ARRAY_SET is access FACET_DATA_ARRAY_SET;

type FACET_COLOUR_LIST
      (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
       LENGTH              : FACET_SET_DIMENSION;
       MODEL               : COLOUR_MODEL) is
  record
    case FACET_DATA_PROVIDED is
```

```
        when FACET_COLOUR |
             FACET_COLOUR_NORMAL |
             FACET_COLOUR_DATA |
             FACET_COLOUR_NORMAL_DATA =>
          COLOURS : COLOUR_VALUE_LIST( MODEL,
                                       LENGTH);
        when others =>
          null;
      end case;
    end record;

  type FACET_NORMAL_LIST
        (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
         LENGTH              : FACET_SET_DIMENSION) is
    record
      case FACET_DATA_PROVIDED is
        when FACET_NORMAL |
             FACET_COLOUR_NORMAL |
             FACET_NORMAL_DATA |
             FACET_COLOUR_NORMAL_DATA =>
          NORMALS : VECTOR_LIST_3(1..LENGTH);
        when others =>
          null;
      end case;
    end record;

  type FACET_DATA_LIST
        (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
         LENGTH              : FACET_SET_DIMENSION;
         PER_FACET           : DATA_VALUE_INDEX) is
    record
      case FACET_DATA_PROVIDED is
        when FACET_DATA |
             FACET_COLOUR_DATA |
             FACET_NORMAL_DATA |
             FACET_COLOUR_NORMAL_DATA =>
          DATA_ITEMS : LIST_OF_DATA_VALUE_SET( 1..LENGTH,
                                               1..PER_FACET);
        when others =>
          null;
      end case;
    end record;

  type FACET_DATA_LIST_SET
        (FACET_DATA_PROVIDED : FACET_DATA_FLAG;
         LENGTH              : FACET_SET_DIMENSION;
         MODEL               : COLOUR_MODEL;
         PER_FACET           : DATA_VALUE_INDEX) is
      record
```

```
        COLOURS      : FACET_COLOUR_LIST( FACET_DATA_PROVIDED,
                                          LENGTH,
                                          MODEL);
        NORMALS      : FACET_NORMAL_LIST( FACET_DATA_PROVIDED,
                                          LENGTH);
        DATA_ITEMS : FACET_DATA_LIST ( FACET_DATA_PROVIDED,
                                       LENGTH,
                                       PER_FACET);
    end record;

type ACCESS_FACET_DATA_LIST_SET is access FACET_DATA_LIST_SET;

type VERTEX_COLOUR_ARRAY
      (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
       WIDTH                : VERTEX_SET_DIMENSION;
       HEIGHT               : VERTEX_SET_DIMENSION;
       MODEL                : COLOUR_MODEL) is
  record
    case VERTEX_DATA_PROVIDED is
      when COORDINATES_COLOUR |
           COORDINATES_COLOUR_NORMAL |
           COORDINATES_COLOUR_DATA |
           COORDINATES_COLOUR_NORMAL_DATA =>
        COLOURS : COLOUR_VALUE_ARRAY(MODEL,
                                     WIDTH,
                                     HEIGHT);
      when others =>
        null;
    end case;
  end record;

type VERTEX_NORMAL_ARRAY
      (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
       WIDTH                : VERTEX_SET_DIMENSION;
       HEIGHT               : VERTEX_SET_DIMENSION) is
  record
    case VERTEX_DATA_PROVIDED is
```

```
        when COORDINATES_NORMAL |
             COORDINATES_COLOUR_NORMAL |
             COORDINATES_NORMAL_DATA |
             COORDINATES_COLOUR_NORMAL_DATA =>
          NORMALS : VECTOR_ARRAY_3( 1..WIDTH,
                                    1..HEIGHT);
        when others =>
          null;
      end case;
    end record;

  type VERTEX_DATA_ARRAY
        (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
         WIDTH                : VERTEX_SET_DIMENSION;
         HEIGHT               : VERTEX_SET_DIMENSION;
         PER_VERTEX           : DATA_VALUE_INDEX) is
    record
      case VERTEX_DATA_PROVIDED is
        when COORDINATES_DATA |
             COORDINATES_COLOUR_DATA |
             COORDINATES_NORMAL_DATA |
             COORDINATES_COLOUR_NORMAL_DATA =>
          DATA_ITEMS : ARRAY_OF_DATA_VALUE_SET( 1..WIDTH,
                                                1..HEIGHT,
                                                1..PER_VERTEX);
        when others =>
          null;
      end case;
    end record;

  type VERTEX_DATA_ARRAY_SET_3
        (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
         WIDTH                : VERTEX_SET_DIMENSION;
         HEIGHT               : VERTEX_SET_DIMENSION;
         MODEL                : COLOUR_MODEL;
         PER_VERTEX           : DATA_VALUE_INDEX) is
      record
        POINTS      : POINT_ARRAY_3( 1..WIDTH,
                                     1..HEIGHT);

        COLOURS     : VERTEX_COLOUR_ARRAY( VERTEX_DATA_PROVIDED,
                                           WIDTH,
                                           HEIGHT,
                                           MODEL);
        NORMALS     : VERTEX_NORMAL_ARRAY( VERTEX_DATA_PROVIDED,
                                           WIDTH,
                                           HEIGHT);
        DATA_ITEMS : VERTEX_DATA_ARRAY( VERTEX_DATA_PROVIDED,
                                        WIDTH,
```

```
                                                HEIGHT,
                                                PER_VERTEX);
    end record;

type ACCESS_VERTEX_DATA_ARRAY_SET_3 is access VERTEX_DATA_ARRAY_SET_3;

type VERTEX_DATA_ARRAY_SET_2
      (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
       WIDTH                : VERTEX_SET_DIMENSION;
       HEIGHT               : VERTEX_SET_DIMENSION;
       MODEL                : COLOUR_MODEL;
       PER_VERTEX           : DATA_VALUE_INDEX) is
    record
      POINTS      : POINT_ARRAY_2( 1..WIDTH,
                                   1..HEIGHT);

      COLOURS     : VERTEX_COLOUR_ARRAY( VERTEX_DATA_PROVIDED,
                                         WIDTH,
                                         HEIGHT,
                                         MODEL);
      NORMALS     : VERTEX_NORMAL_ARRAY( VERTEX_DATA_PROVIDED,
                                         WIDTH,
                                         HEIGHT);
      DATA_ITEMS : VERTEX_DATA_ARRAY( VERTEX_DATA_PROVIDED,
                                      WIDTH,
                                      HEIGHT,
                                      PER_VERTEX);
    end record;

type ACCESS_VERTEX_DATA_ARRAY_SET_2 is access VERTEX_DATA_ARRAY_SET_2;

type VERTEX_COLOUR_LIST
      (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
       LENGTH               : VERTEX_SET_DIMENSION;
       MODEL                : COLOUR_MODEL) is
  record
    case VERTEX_DATA_PROVIDED is
      when COORDINATES_COLOUR |
           COORDINATES_COLOUR_NORMAL |
           COORDINATES_COLOUR_DATA |
           COORDINATES_COLOUR_NORMAL_DATA =>
        COLOURS : COLOUR_VALUE_LIST( MODEL,
                                     LENGTH);
      when others =>
        null;
    end case;
  end record;

type VERTEX_NORMAL_LIST
```

```
        (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
         LENGTH               : VERTEX_SET_DIMENSION) is
    record
      case VERTEX_DATA_PROVIDED is
        when COORDINATES_NORMAL |
             COORDINATES_COLOUR_NORMAL |
             COORDINATES_NORMAL_DATA |
             COORDINATES_COLOUR_NORMAL_DATA =>
          NORMALS : VECTOR_LIST_3(1..LENGTH);
        when others =>
          null;
      end case;
    end record;

  type VERTEX_DATA_LIST
        (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
         LENGTH               : VERTEX_SET_DIMENSION;
         PER_VERTEX           : DATA_VALUE_INDEX) is
    record
      case VERTEX_DATA_PROVIDED is
        when COORDINATES_DATA |
             COORDINATES_COLOUR_DATA |
             COORDINATES_NORMAL_DATA |
             COORDINATES_COLOUR_NORMAL_DATA =>
          DATA_ITEMS : LIST_OF_DATA_VALUE_SET( 1..LENGTH,
                                               1..PER_VERTEX);
        when others =>
          null;
      end case;
    end record;

  type VERTEX_DATA_LIST_SET_3
        (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
         LENGTH               : VERTEX_SET_DIMENSION;
         MODEL                : COLOUR_MODEL;
         PER_VERTEX           : DATA_VALUE_INDEX) is
      record
        POINTS      : POINT_LIST_3(1..LENGTH);

        COLOURS     : VERTEX_COLOUR_LIST( VERTEX_DATA_PROVIDED,
                                          LENGTH,
                                          MODEL);
        NORMALS     : VERTEX_NORMAL_LIST( VERTEX_DATA_PROVIDED,
                                          LENGTH);
        DATA_ITEMS : VERTEX_DATA_LIST( VERTEX_DATA_PROVIDED,
                                       LENGTH,
                                       PER_VERTEX);
```

```
    end record;

type ACCESS_VERTEX_DATA_LIST_SET_3 is access VERTEX_DATA_LIST_SET_3;

type LIST_OF_VERTEX_DATA_LIST_SET_3 is array (VERTEX_SET_COUNT) of
        ACCESS_VERTEX_DATA_LIST_SET_3;

type ACCESS_LIST_OF_VERTEX_DATA_LIST_SET_3 is
        access LIST_OF_VERTEX_DATA_LIST_SET_3;

type VERTEX_DATA_LIST_SET_2
       (VERTEX_DATA_PROVIDED : VERTEX_DATA_FLAG;
        LENGTH               : VERTEX_SET_DIMENSION;
        MODEL                : COLOUR_MODEL;
        PER_VERTEX           : DATA_VALUE_INDEX) is
    record
      POINTS      : POINT_LIST_2(1..LENGTH);
      COLOURS     : VERTEX_COLOUR_LIST( VERTEX_DATA_PROVIDED,
                                        LENGTH,
                                        MODEL);
      NORMALS     : VERTEX_NORMAL_LIST( VERTEX_DATA_PROVIDED,
                                        LENGTH);
      DATA_ITEMS : VERTEX_DATA_LIST( VERTEX_DATA_PROVIDED,
                                     LENGTH,
                                     PER_VERTEX);
    end record;

type ACCESS_VERTEX_DATA_LIST_SET_2 is access VERTEX_DATA_LIST_SET_2;

type LIST_OF_VERTEX_DATA_LIST_SET_2 is array (VERTEX_SET_COUNT) of
        ACCESS_VERTEX_DATA_LIST_SET_2;

type ACCESS_LIST_OF_VERTEX_DATA_LIST_SET_2 is
        access LIST_OF_VERTEX_DATA_LIST_SET_2;

type VERTEX_COLOUR_LIST_SET_3
       (VERTEX_COLOUR_PROVIDED   : VERTEX_COLOUR_FLAG;
        LENGTH                   : VERTEX_SET_DIMENSION;
        MODEL                    : COLOUR_MODEL) is
    record
      POINTS      : POINT_LIST_3(1..LENGTH);
      COLOURS     : VERTEX_COLOUR_LIST( VERTEX_COLOUR_PROVIDED,
                                        LENGTH,
                                        MODEL);
    end record;

type ACCESS_VERTEX_COLOUR_LIST_SET_3 is access VERTEX_COLOUR_LIST_SET_3;

type LIST_OF_VERTEX_COLOUR_LIST_SET_3 is array (VERTEX_SET_COUNT) of
        ACCESS_VERTEX_COLOUR_LIST_SET_3;
```

```
    type ACCESS_LIST_OF_VERTEX_COLOUR_LIST_SET_3 is
            access LIST_OF_VERTEX_COLOUR_LIST_SET_3;

    procedure DEALLOCATE (OBJECT: in out LIST_OF_VERTEX_DATA_LIST_SET_3);

    procedure DEALLOCATE (OBJECT: in out LIST_OF_VERTEX_DATA_LIST_SET_2);

    procedure DEALLOCATE (OBJECT: in out LIST_OF_VERTEX_COLOUR_LIST_SET_3);

end PHIGS_COORDINATE_SYSTEM;


package body PHIGS_COORDINATE_SYSTEM is

-- This package body contains stubs for each of the subprograms contained in the
-- PHIGS_COORDINATE_SYSTEM package.  The purpose of these stubs is to show
-- compilability for the PHIGS/Ada binding.  An implementation would replace these
-- stubs with code appropriate to implement the functionality.

    procedure DEALLOCATE (OBJECT   : in out LIST_OF_VERTEX_DATA_LIST_SET_3) is
    begin
      null;
    end DEALLOCATE;

    procedure DEALLOCATE (OBJECT   : in out LIST_OF_VERTEX_DATA_LIST_SET_2) is
    begin
      null;
    end DEALLOCATE;

    procedure DEALLOCATE (OBJECT   : in out LIST_OF_VERTEX_COLOUR_LIST_SET_3) is
    begin
      null;
    end DEALLOCATE;

end PHIGS_COORDINATE_SYSTEM;
```

```
-- The PHIGS Types Package

with  PHIGS_CONFIGURATION,
      PHIGS_BASE_TYPES,
      PHIGS_COLOUR_TYPES,
      PHIGS_STANDARD_TYPES,
      PHIGS_COORDINATE_SYSTEM,
      PHIGS_LIST_UTILITIES,
      PHIGS_NAME_SET_FACILITY;


package PHIGS_TYPES is

-- This package contains all the data type definitions used to define
-- the Ada binding to PHIGS.

-- Configuration names

use PHIGS_CONFIGURATION;

   -- Make PHIGS_BASE_TYPES visible externally.

   subtype PHIGS_INTEGER is PHIGS_BASE_TYPES.PHIGS_INTEGER;
   subtype PHIGS_NATURAL is PHIGS_BASE_TYPES.PHIGS_NATURAL;
   subtype PHIGS_POSITIVE is PHIGS_BASE_TYPES.PHIGS_POSITIVE;
   subtype PHIGS_STRING is PHIGS_BASE_TYPES.PHIGS_STRING;
   subtype SCALE_FACTOR is PHIGS_BASE_TYPES.SCALE_FACTOR;
   subtype SMALL_NATURAL is PHIGS_BASE_TYPES.SMALL_NATURAL;
   subtype COLOUR_VALUE_SET_DIMENSION is
                              PHIGS_BASE_TYPES.COLOUR_VALUE_SET_DIMENSION;
   subtype FACET_SET_DIMENSION is PHIGS_BASE_TYPES.FACET_SET_DIMENSION;
   subtype VERTEX_SET_COUNT is PHIGS_BASE_TYPES.VERTEX_SET_COUNT;
   subtype VERTEX_SET_DIMENSION is PHIGS_BASE_TYPES.VERTEX_SET_DIMENSION;
   subtype SPLINE_ORDER is PHIGS_BASE_TYPES.SPLINE_ORDER;
   subtype SPLINE_RATIONALITY is PHIGS_BASE_TYPES.SPLINE_RATIONALITY;

   -- Make PHIGS_STANDARD_TYPES visible externally.

   subtype DATA_VALUE is PHIGS_STANDARD_TYPES.DATA_VALUE;
   subtype DATA_VALUE_INDEX is PHIGS_STANDARD_TYPES.DATA_VALUE_INDEX;
   subtype DATA_VALUE_SET is PHIGS_STANDARD_TYPES.DATA_VALUE_SET;
   subtype ACCESS_DATA_VALUE_SET is
                              PHIGS_STANDARD_TYPES.ACCESS_DATA_VALUE_SET;
   subtype DATA_MAPPING_INDEX is PHIGS_STANDARD_TYPES.DATA_MAPPING_INDEX;
   subtype LIST_OF_DATA_VALUE_SET is
                              PHIGS_STANDARD_TYPES.LIST_OF_DATA_VALUE_SET;
   subtype ARRAY_OF_DATA_VALUE_SET is
                              PHIGS_STANDARD_TYPES.ARRAY_OF_DATA_VALUE_SET;
   subtype COLOUR_COMPONENTS is PHIGS_STANDARD_TYPES.COLOUR_COMPONENTS;
```

```
subtype COLOUR_INDEX is PHIGS_STANDARD_TYPES.COLOUR_INDEX;
subtype COLOUR_MODEL is PHIGS_STANDARD_TYPES.COLOUR_MODEL;

-- The following constants define the colour models specified by PHIGS:

INDIRECT    : constant COLOUR_MODEL := 0;
RGB         : constant COLOUR_MODEL := 1;
CIELUV      : constant COLOUR_MODEL := 2;
HSV         : constant COLOUR_MODEL := 3;
HLS         : constant COLOUR_MODEL := 4;

subtype COLOUR_COEFFICIENT is PHIGS_STANDARD_TYPES.COLOUR_COEFFICIENT;
subtype COLOUR_COEFFICIENT_ARRAY is
                        PHIGS_STANDARD_TYPES.COLOUR_COEFFICIENT_ARRAY;
subtype INTENSITY is PHIGS_STANDARD_TYPES.INTENSITY;

-- Types for CIELUV Colour Model

subtype CIELUV_COLOUR_VALUE is
                              PHIGS_STANDARD_TYPES.CIELUV_COLOUR_VALUE;
subtype CIELUV_HOMOGENEOUS_COLOUR_VALUE is
            PHIGS_STANDARD_TYPES.CIELUV_HOMOGENEOUS_COLOUR_VALUE;
package CIELUV_TYPES renames PHIGS_STANDARD_TYPES.CIELUV_TYPES;

-- Type for Generic Colour Models

package GENERIC_COLOUR_TYPES renames
                              PHIGS_STANDARD_TYPES.GENERIC_COLOUR_TYPES;

-- Types for HLS Colour Model

subtype HLS_COLOUR_VALUE is PHIGS_STANDARD_TYPES.HLS_COLOUR_VALUE;
subtype HLS_HOMOGENEOUS_COLOUR_VALUE is
                  PHIGS_STANDARD_TYPES.HLS_HOMOGENEOUS_COLOUR_VALUE;
package HLS_TYPES renames PHIGS_STANDARD_TYPES.HLS_TYPES;

-- Types for HSV Colour Model

subtype HSV_COLOUR_VALUE is PHIGS_STANDARD_TYPES.HSV_COLOUR_VALUE;
subtype HSV_HOMOGENEOUS_COLOUR_VALUE is
                  PHIGS_STANDARD_TYPES.HSV_HOMOGENEOUS_COLOUR_VALUE;
package HSV_TYPES renames PHIGS_STANDARD_TYPES.HSV_TYPES;

-- Types for Indirect Colour Model

subtype INDIRECT_HOMOGENEOUS_COLOUR_VALUE is
            PHIGS_STANDARD_TYPES.INDIRECT_HOMOGENEOUS_COLOUR_VALUE;
package INDIRECT_TYPES renames PHIGS_STANDARD_TYPES.INDIRECT_TYPES;
```

```
-- Types for RGB Colour Model

subtype RGB_COLOUR_VALUE is PHIGS_STANDARD_TYPES.RGB_COLOUR_VALUE;
subtype RGB_HOMOGENEOUS_COLOUR_VALUE is
                    PHIGS_STANDARD_TYPES.RGB_HOMOGENOUS_COLOUR_VALUE;
package RGB_TYPES renames PHIGS_STANDARD_TYPES.RGB_TYPES;

-- GENERAL_COLOUR

subtype GENERAL_COLOUR is PHIGS_STANDARD_TYPES.GENERAL_COLOUR;

-- COLOUR_VALUE_ARRAY

subtype COLOUR_VALUE_ARRAY is
                                PHIGS_STANDARD_TYPES.COLOUR_VALUE_ARRAY;

-- COLOUR_VALUE_LIST

subtype COLOUR_VALUE_LIST is PHIGS_STANDARD_TYPES.COLOUR_VALUE_LIST;
subtype LIST_OF_COLOUR_VALUE_LIST is
                        PHIGS_STANDARD_TYPES.LIST_OF_COLOUR_VALUE_LIST;

-- FACET_DATA_FLAG

subtype FACET_DATA_FLAG is PHIGS_STANDARD_TYPES.FACET_DATA_FLAG;

-- VERTEX_DATA_FLAG

subtype VERTEX_DATA_FLAG is PHIGS_STANDARD_TYPES.VERTEX_DATA_FLAG;

-- VERTEX_COLOUR_FLAG

subtype VERTEX_COLOUR_FLAG is PHIGS_STANDARD_TYPES.VERTEX_COLOUR_FLAG;

-- CHOICE_SMALL_NATURAL

subtype CHOICE_SMALL_NATURAL is
  PHIGS_NATURAL range 0..MAX_CHOICE_SMALL_NATURAL;

-- This is a subtype declaration which allows for
-- unconstrained record objects for CHOICE_PROMPT_STRING_LIST
-- type without causing the exception STORAGE_ERROR to be raised.

-- COLOUR_MATRIX_SMALL_NATURAL

subtype COLOUR_MATRIX_SMALL_NATURAL is
  PHIGS_NATURAL range 0..MAX_COLOUR_MATRIX_SMALL_NATURAL;

-- This is a subtype declaration which allows for unconstrained record
```

```
-- objects of VARIABLE_COLOUR_MATRIX type without causing the
-- exception STORAGE_ERROR to be raised.

-- FILE_SMALL_NATURAL

subtype FILE_SMALL_NATURAL is
  PHIGS_NATURAL range 0..MAX_FILE_SMALL_NATURAL;

-- This is a subtype declaration which allows for
-- unconstrained record objects for VARIABLE_FILE_ID type
-- without causing the exception STORAGE_ERROR to be raised.

-- INPUT_STRING_SMALL_NATURAL

subtype INPUT_STRING_SMALL_NATURAL is
  PHIGS_NATURAL range 0..MAX_INPUT_STRING_SMALL_NATURAL;

-- This is a subtype declaration which allows for
-- unconstrained record objects for INPUT_STRING type
-- without causing the exception STORAGE_ERROR to be raised.

-- STRING_SMALL_NATURAL

subtype STRING_SMALL_NATURAL is
  PHIGS_NATURAL range 0..MAX_STRING_SMALL_NATURAL;

-- This is a subtype declaration which allows for
-- unconstrained record objects for various string record
-- types without causing the exception STORAGE_ERROR to be
-- raised.


-- DATA_VALUE_INDEX_PAIR

type DATA_VALUE_INDEX_PAIR is
  record
    INDEX_A  : DATA_VALUE_INDEX;
    INDEX_B  : DATA_VALUE_INDEX;
  end record;

-- Provides for selecting from bi-valued sets of data mapping input values.

-- DATA_VALUE_RANGE
```

```
type DATA_VALUE_RANGE is
  record
    UPPER   : DATA_VALUE;
    LOWER   : DATA_VALUE;
  end record;

-- Provides for specifying a range of data mapping input values.

-- SOURCE_SELECTOR

type SOURCE_SELECTOR is ( COLOUR_ASPECT,
                          VERTEX_COLOUR,
                          VERTEX_DATA,
                          FACET_COLOUR,
                          FACET_DATA);

-- Provides for specifying sources for colour information.

-- SOURCE_SELECTOR_LIST

type SOURCE_SELECTOR_LIST is
  array (SMALL_NATURAL) of SOURCE_SELECTOR;

-- Provides for specification of ordered lists of sources for colour information.

-- DATA_MAPPING_METHOD

type DATA_MAPPING_METHOD is new PHIGS_INTEGER;

-- Provides for differentiating methods of performing data mapping.

-- DATA_MAPPING_METHODS

package DATA_MAPPING_METHODS is
        new PHIGS_LIST_UTILITIES
              (DATA_MAPPING_METHOD,
               MAX_DATA_MAPPING_METHODS_SUPPORTED);

-- Provides for lists of methods of performing data mapping.

DATA_MAPPING_COLOUR          : constant DATA_MAPPING_METHOD := 1;
SINGLE_VALUE_UNIFORM         : constant DATA_MAPPING_METHOD := 2;
SINGLE_VALUE_NON_UNIFORM     : constant DATA_MAPPING_METHOD := 3;
BI_VALUE_UNIFORM             : constant DATA_MAPPING_METHOD := 4;
BI_VALUE_NON_UNIFORM         : constant DATA_MAPPING_METHOD := 5;

-- DATA_MAPPING_INDICES
```

```
package DATA_MAPPING_INDICES is
        new PHIGS_LIST_UTILITIES
                (DATA_MAPPING_INDEX,
                 MAX_DATA_MAPPING_INDICES_SUPPORTED);

-- Provides for lists of data mapping indices.

-- RETURN_SPLINE_ORDER

subtype RETURN_SPLINE_ORDER is
            SPLINE_ORDER range 6..SPLINE_ORDER'last;

-- Provides for restricting spline orders to those acceptable for returning values of the
-- facilities.

-- COLOUR_AVAILABLE

type COLOUR_AVAILABLE is (  COLOUR,
                            MONOCHROME);

-- Indicates whether colour output is available on a
-- workstation.

-- COLOUR_INDICES

package COLOUR_INDICES is
  new PHIGS_LIST_UTILITIES ( COLOUR_INDEX,
                             MAX_COLOUR_INDICES_SUPPORTED);

-- Provides for a set of colour indices which are available
-- on a particular workstation.

-- COLOUR_MATRIX

type COLOUR_MATRIX is array ( COLOUR_MATRIX_SMALL_NATURAL range <>,
                              COLOUR_MATRIX_SMALL_NATURAL range <>)
  of COLOUR_INDEX;

-- Provides for matrices containing colour indices corresponding
-- to a cell array or pattern array.

-- COLOUR_MODELS

package COLOUR_MODELS is
  new PHIGS_LIST_UTILITIES ( COLOUR_MODEL,
                             MAX_COLOUR_MODELS_SUPPORTED);

-- Provides for lists of colour models.
```

```
-- The following constant is for use when specifying the rendering colour model:

WS_DEPENDENT_COLOUR_MODEL  : constant COLOUR_MODEL := 0;

-- COLOUR_COEFFICIENTS

package COLOUR_COEFFICIENTS is
  new PHIGS_LIST_UTILITIES ( COLOUR_COEFFICIENT,
                             MAX_COLOUR_COEFFICIENTS);

-- Provides for a lists of colour coefficients for the generalized
-- colour model.


-- COLOUR_REPRESENTATION

type COLOUR_REPRESENTATION (MODEL : COLOUR_MODEL := RGB) is
  record
    case MODEL is

      when RGB =>
        RED_RGB              : INTENSITY;
        GREEN_RGB            : INTENSITY;
        BLUE_RGB             : INTENSITY;

      when CIELUV =>
        L_STAR_CIE           : COLOUR_COEFFICIENT;
        U_STAR_CIE           : COLOUR_COEFFICIENT;
        V_STAR_CIE           : COLOUR_COEFFICIENT;

      when HSV =>
        HUE_HSV              : INTENSITY;
        SATURATION_HSV       : INTENSITY;
        VALUE_HSV            : INTENSITY;

      when HLS =>
        HUE_HLS              : INTENSITY;
        LIGHTNESS_HLS        : INTENSITY;
        SATURATION_HLS       : INTENSITY;

      when others =>
        GENERIC_COMPONENTS : COLOUR_COEFFICIENT_ARRAY;
    end case;
  end record;

-- Defines the representation of a colour as a combination of three
-- components whose meaning depends on the colour model.  Additional
-- variants may be included when additional registered or unregistered
-- colour models are supported by an implementation.
```

```
-- CHROMATICITY_COEFFICIENT

type CHROMATICITY_COEFFICIENT is
  record
    U_PRIME  : COLOUR_COEFFICIENT;
    V_PRIME  : COLOUR_COEFFICIENT;
    Y        : COLOUR_COEFFICIENT;
  end record;

--  Defines a CIE colour coefficient value.

-- CHROMATICITY_COEFFICIENT_SET

type CHROMATICITY_COEFFICIENT_SET is
  record
    R  : CHROMATICITY_COEFFICIENT;
    G  : CHROMATICITY_COEFFICIENT;
    B  : CHROMATICITY_COEFFICIENT;
  end record;

--  Defines a set of CIE primary colour chromaticity coefficients.

-- ACCESS_COLOUR_VALUE_ARRAY

type ACCESS_COLOUR_VALUE_ARRAY is access COLOUR_VALUE_ARRAY;

-- Provides for pointers to arrays of general colour values.



-- ACCESS_COLOUR_VALUE_LIST

type ACCESS_COLOUR_VALUE_LIST is access COLOUR_VALUE_LIST;

-- Provides for pointers to lists of general colour values.

-- ACCESS_LIST_OF_COLOUR_VALUE_LIST

type ACCESS_LIST_OF_COLOUR_VALUE_LIST is
        access LIST_OF_COLOUR_VALUE_LIST;

-- Provides for pointers to lists of lists of colour values values.

-- COLOUR CONTROL POINT ARRAY

type COLOUR_CONTROL_POINT_ARRAY
  (MODEL          : COLOUR_MODEL := RGB;
   RATIONALITY    : SPLINE_RATIONALITY := RATIONAL;
```

```
    LENGTH_U : COLOUR_VALUE_SET_DIMENSION := 1;
    LENGTH_V : COLOUR_VALUE_SET_DIMENSION := 1) is
     record
       case MODEL is

         when INDIRECT =>
           COLOURS_INDIRECT : INDIRECT_TYPES.CONTROL_POINT_ARRAY
                                (RATIONALITY, LENGTH_U, LENGTH_V);

         when RGB =>
           COLOURS_RGB : RGB_TYPES.CONTROL_POINT_ARRAY
                           (RATIONALITY, LENGTH_U, LENGTH_V);

         when CIELUV =>
           COLOURS_CIELUV : CIELUV_TYPES.CONTROL_POINT_ARRAY
                              (RATIONALITY, LENGTH_U, LENGTH_V);

         when HSV =>
           COLOURS_HSV  : HSV_TYPES.CONTROL_POINT_ARRAY
                            (RATIONALITY, LENGTH_U, LENGTH_V);

         when HLS =>
           COLOURS_HLS : HLS_TYPES.CONTROL_POINT_ARRAY
                           (RATIONALITY, LENGTH_U, LENGTH_V);

         when others =>
           COLOURS_GENERIC : GENERIC_COLOUR_TYPES.CONTROL_POINT_ARRAY
                               (RATIONALITY, LENGTH_U, LENGTH_V);

       end case;
     end record;

-- Provides for specification of two-dimensional arrays of colour control point values
-- all of which are the same colour model.


-- COLOUR CONTROL POINT LIST

type COLOUR_CONTROL_POINT_LIST
    (MODEL          : COLOUR_MODEL := RGB;
     RATIONALITY    : SPLINE_RATIONALITY := RATIONAL;
     LENGTH : COLOUR_VALUE_SET_DIMENSION := 0) is
      record
        case MODEL is

          when INDIRECT =>
            COLOURS_INDIRECT : INDIRECT_TYPES.CONTROL_POINT_LIST
                                 (RATIONALITY, LENGTH);
```

```
            when RGB =>
              COLOURS_RGB : RGB_TYPES.CONTROL_POINT_LIST
                                        (RATIONALITY, LENGTH);

            when CIELUV =>
              COLOURS_CIELUV  : CIELUV_TYPES.CONTROL_POINT_LIST
                                        (RATIONALITY, LENGTH);

            when HSV =>
              COLOURS_HSV : HSV_TYPES.CONTROL_POINT_LIST
                                        (RATIONALITY, LENGTH);

            when HLS =>
              COLOURS_HLS : HLS_TYPES.CONTROL_POINT_LIST
                                        (RATIONALITY, LENGTH);

            when others =>
              COLOURS_GENERIC : GENERIC_COLOUR_TYPES.CONTROL_POINT_LIST
                                        (RATIONALITY, LENGTH);

          end case;
        end record;

    -- Provides for specification of lists of colour control point values
    -- all of which are the same colour model.

    -- DATA_MAPPING_DATA_RECORD

    type DATA_MAPPING_DATA_RECORD
      (METHOD : DATA_MAPPING_METHOD := DATA_MAPPING_COLOUR) is
        record
          case METHOD is

            when DATA_MAPPING_COLOUR =>
              COLOUR_SELECTORS : SOURCE_SELECTOR_LIST;

            when SINGLE_VALUE_UNIFORM =>
              SINGLE_UNIFORM_SELECTORS          : SOURCE_SELECTOR_LIST;
              SINGLE_UNIFORM_DATA_VALUE_IND     : DATA_VALUE_INDEX;
              SINGLE_UNIFORM_RANGE              : DATA_VALUE_RANGE;
              SINGLE_UNIFORM_COLOURS            : ACCESS_COLOUR_VALUE_LIST;

            when SINGLE_VALUE_NON_UNIFORM =>
              SINGLE_NON_UNIFORM_SELECTORS      : SOURCE_SELECTOR_LIST;
              SINGLE_NON_UNIFORM_DATA_VALUE_IND : DATA_VALUE_INDEX;
              SINGLE_NON_UNIFORM_BOUNDARIES     : ACCESS_DATA_VALUE_SET;
              SINGLE_NON_UNIFORM_COLOURS        : ACCESS_COLOUR_VALUE_LIST;
```

```
            when BI_VALUE_UNIFORM =>
              BI_UNIFORM_SELECTORS : SOURCE_SELECTOR_LIST;
              BI_UNIFORM_DATA_VALUE_IND_PAIR  : DATA_VALUE_INDEX_PAIR;
              BI_UNIFORM_A_RANGE     : DATA_VALUE_RANGE;
              BI_UNIFORM_B_RANGE     : DATA_VALUE_RANGE;
              BI_UNIFORM_COLOURS     : ACCESS_LIST_OF_COLOUR_VALUE_LIST;

            when BI_VALUE_NON_UNIFORM =>
              BI_NON_UNIFORM_SELECTORS       : SOURCE_SELECTOR_LIST;
              BI_NON_UNIFORM_DATA_VALUE_IND_PAIR : DATA_VALUE_INDEX_PAIR;
              BI_NON_UNIFORM_A_BOUNDARIES : ACCESS_DATA_VALUE_SET;
              BI_NON_UNIFORM_B_BOUNDARIES : ACCESS_DATA_VALUE_SET;
              BI_NON_UNIFORM_COLOURS         : ACCESS_COLOUR_VALUE_ARRAY;

            when others =>
              null;

          end case;
        end record;

-- Provides for specifying values for different methods of data mapping.
-- Additional variants may be included when additional registered or unregistered data
-- mapping methods are supported by an implementation.

-- MC_TYPE

type MC_TYPE is digits PHIGS_PRECISION;

-- Defines the type of coordinate in the Modeling Coordinate
-- System.

-- MC

package MC is new PHIGS_COORDINATE_SYSTEM (MC_TYPE);

-- Defines the Modeling Coordinate System.

-- WC_TYPE

type WC_TYPE is digits PHIGS_PRECISION;

-- Defines the type of coordinate in the World Coordinate
-- System.

-- WC

package WC is new PHIGS_COORDINATE_SYSTEM (WC_TYPE);

-- Defines the World Coordinate System.
```

```
-- VRC_TYPE

type VRC_TYPE is digits PHIGS_PRECISION;

-- Defines the type of coordinate in the View Reference
-- Coordinate System.

-- VRC

package VRC is new PHIGS_COORDINATE_SYSTEM (VRC_TYPE);

-- Defines the viewing coordinate system.

-- NPC_TYPE

type NPC_TYPE is digits PHIGS_PRECISION;

-- Defines the type of a coordinate in the Normalized
-- Projection Coordinate System.

-- NPC

package NPC is new PHIGS_COORDINATE_SYSTEM (NPC_TYPE);

-- Defines the Normalized Projection Coordinate System.

-- DC_TYPE

type DC_TYPE is digits PHIGS_PRECISION;

-- The type of a coordinate component in the Device Coordinate
-- System.

-- DC

package DC is new PHIGS_COORDINATE_SYSTEM (DC_TYPE);

-- Defines the Device Coordinate System.


-- SPC_TYPE

type SPC_TYPE is digits PHIGS_PRECISION;

-- Defines the type of a coordinate in the Spline Parameter Coordinate System.


-- SPC
```

```
package SPC is new PHIGS_COORDINATE_SYSTEM(SPC_TYPE);

-- Defines the Spline Parameter Coordinate System.

-- KNOT_VECTOR

subtype KNOT_VECTOR is SPC.MAGNITUDE_LIST;

-- Provides for specifying spline knot vectors.


-- CURVE COLOURSPLINE

type CURVE_COLOURSPLINE( MODEL          : COLOUR_MODEL;
                         RATIONALITY    : SPLINE_RATIONALITY;
                         KNOT_COUNT     : SMALL_NATURAL;
                         LENGTH         : COLOUR_VALUE_SET_DIMENSION) is
  record
    ORDER           : SPLINE_ORDER;
    KNOTS           : KNOT_VECTOR(KNOT_COUNT);
    CONTROL_POINTS  : COLOUR_CONTROL_POINT_LIST
                        (MODEL, RATIONALITY, LENGTH);
  end record;

-- Provides for colour splines for use with Non-uniform B-Spline Curve primitives.


-- ACCESS CURVE COLOURSPLINE

type ACCESS_CURVE_COLOURSPLINE is
        access CURVE_COLOURSPLINE;

-- Provides for pointers to colourspline definitions.


-- SURFACE COLOURSPLINE

type SURFACE_COLOURSPLINE
        (MODEL          : COLOUR_MODEL;
         RATIONALITY    : SPLINE_RATIONALITY;
         KNOT_COUNT_U   : SMALL_NATURAL;
         KNOT_COUNT_V   : SMALL_NATURAL;
         LENGTH_U       : COLOUR_VALUE_SET_DIMENSION;
         LENGTH_V       : COLOUR_VALUE_SET_DIMENSION) is
  record
    U_ORDER         : SPLINE_ORDER;
    V_ORDER         : SPLINE_ORDER;
    U_KNOTS         : KNOT_VECTOR(KNOT_COUNT_U);
    V_KNOTS         : KNOT_VECTOR(KNOT_COUNT_V);
```

```
    CONTROL_POINTS : COLOUR_CONTROL_POINT_ARRAY
                        (MODEL, RATIONALITY, LENGTH_U, LENGTH_V);
  end record;

-- Provides for colour splines for use with Non-uniform B-Spline Surface primitives.


-- ACCESS SURFACE COLOURSPLINE

type ACCESS_SURFACE_COLOURSPLINE is
        access SURFACE_COLOURSPLINE;

-- Provides for pointers to colourspline definitions.


-- DATASPLINE

type DATASPLINE( MODEL          : COLOUR_MODEL := RGB;
                 RATIONALITY    : SPLINE_RATIONALITY := RATIONAL;
                 KNOT_COUNT_U   : SMALL_NATURAL := 4;
                 KNOT_COUNT_V   : SMALL_NATURAL := 4;
                 WIDTH          : COLOUR_VALUE_SET_DIMENSION := 4;
                 HEIGHT         : COLOUR_VALUE_SET_DIMENSION := 4) is
  record
    U_ORDER          : SPLINE_ORDER;
    V_ORDER          : SPLINE_ORDER;
    U_KNOTS          : KNOT_VECTOR(KNOT_COUNT_U);
    V_KNOTS          : KNOT_VECTOR(KNOT_COUNT_V);
    CONTROL_POINTS : SPC.CONTROL_POINT_ARRAY_3
                        (RATIONALITY, WIDTH, HEIGHT);
  end record;

-- Provides for data splines for use with Non-uniform B-Spline primitives.

-- DATASPLINE_LIST

type DATASPLINE_LIST is
        array (DATA_VALUE_INDEX range <>) of DATASPLINE;

-- Provides for lists of data mapping data splines.

-- ACCESS_DATASPLINE_LIST

type ACCESS_DATASPLINE_LIST is
        access DATASPLINE_LIST;

-- Provides for pointers to lists of dataspline definitions.

-- GENERAL_FLOAT_PARAMETER
```

```
type GENERAL_FLOAT_PARAMETER is digits PHIGS_PRECISION;

-- Defines a type for specifying implementation-specific real
-- data in GDP, GSE, and ESCAPE data records.

-- OFF_ON

type OFF_ON is ( OFF,
                 ON);

-- Generic type for off/on switches.

-- ANGLE

type ANGLE is digits PHIGS_PRECISION;

-- Values used in the modeling transformation utility functions
-- for specifying angles of rotation in radians.  Positive
-- indicates a counterclockwise direction.

-- ANNOTATION_STYLE

type ANNOTATION_STYLE is new PHIGS_INTEGER;

-- Defines the annotation styles for annotation primitives.

-- ANNOTATION_STYLES

package ANNOTATION_STYLES is
  new PHIGS_LIST_UTILITIES (ANNOTATION_STYLE,
                            MAX_ANNOTATION_STYLES_SUPPORTED);

-- Provides for lists of annotation styles.

-- APPLICATION_DATA_RANGE

subtype APPLICATION_DATA_RANGE is
  PHIGS_NATURAL range 0..MAX_APPLICATION_DATA;

-- Defines range of lengths for application data objects.

-- APPLICATION DATA RECORD

type APPLICATION_DATA_RECORD
  (LENGTH : APPLICATION_DATA_RANGE := 0) is
    record
      DATA : PHIGS_STRING (1..LENGTH);
    end record;
```

```
-- Defines type for defining application data objects.

-- ARCHIVE_ID

type ARCHIVE_ID is new PHIGS_NATURAL;

-- Provides for an application specified pointer to archive
-- files separate from the file identifier.

-- ARCHIVE_IDS

package ARCHIVE_IDS is
  new PHIGS_LIST_UTILITIES ( ARCHIVE_ID,
                             MAX_ARCHIVE_IDS_SUPPORTED);

-- Provides for lists of archive identifiers.

-- ARCHIVE_STATE

type ARCHIVE_STATE is ( ARCL,
                        AROP);

-- The type used to return the archive state.

-- ASF

type ASF is ( BUNDLED,
              INDIVIDUAL);

-- This type defines an aspect source flag whose
-- value indicates whether an aspect of a primitive
-- should be set from a bundle table or from an
-- individual attribute.

-- ASPECT

type ASPECT is ( TYPE_OF_LINE,
                 LINEWIDTH_SF,
                 LINE_COLOUR,

                 TYPE_OF_MARKER,
                 SIZE,
                 MARKER_COLOUR,
```

```
    FONT,
    PRECISION,
    EXPANSION,
    SPACING,
    TEXT_COLOUR,

    STYLE_OF_INTERIOR,
    STYLE_IND,
    INTERIOR_COLOUR,

    FLAG,
    TYPE_OF_EDGE,
    EDGEWIDTH_SF,
    EDGE_COLOUR,

    GENERAL_POLYLINE_COLOUR,
    GENERAL_POLYMARKER_COLOUR,
    GENERAL_TEXT_COLOUR,
    GENERAL_INTERIOR_COLOUR,
    GENERAL_EDGE_COLOUR,

    METHOD_OF_POLYLINE_SHADING,
    METHOD_OF_INTERIOR_SHADING,
    METHOD_OF_DATA_MAPPING,
    REFLECTANCE_PROPERTIES,
    MODEL_OF_REFLECTANCE,

    BACK_STYLE_OF_INTERIOR,
    BACK_STYLE_IND,
    BACK_INTERIOR_COLOUR_DIR,
    BACK_METHOD_OF_INTERIOR_SHADING,
    BACK_METHOD_OF_DATA_MAPPING,
    BACK_REFLECTANCE_PROPERTIES,
    BACK_REFLECTANCE_MODEL,

    CRITERIA_FOR_CURVE_APPROX,
    CRITERIA_FOR_SURFACE_APPROX,
    CHARACTERISTICS_OF_PARAMETRIC_SURFACE);
```

-- This type lists the aspects for which an aspect source flag exists in PHIGS PLUS.

-- ATTRIBUTES_USED_TYPE

```
type ATTRIBUTES_USED_TYPE is ( POLYLINE_ATTRIBUTES,
                               POLYMARKER_ATTRIBUTES,
                               TEXT_ATTRIBUTES,
                               INTERIOR_ATTRIBUTES,
                               EDGE_ATTRIBUTES,
                               REFLECTANCE_ATTRIBUTES,
                               PARAMETRIC_SURFACE_ATTRIBUTES);

-- The types of attributes which may be used in generating output for a GDP and in generating
-- prompt and echo information for certain prompt and echo types of certain classes of input
-- devices.



-- ATTENUATION_COEFFICIENT

type ATTENUATION_COEFFICIENTS is
  record
    C1 : COLOUR_COEFFICIENT;
    C2 : COLOUR_COEFFICIENT;
  end record;

-- Provides for specification of light source attenuation.


-- ATTRIBUTES_USED

package ATTRIBUTES_USED is
  new PHIGS_LIST_UTILITIES
    (ATTRIBUTES_USED_TYPE,
     ATTRIBUTES_USED_TYPE'POS(ATTRIBUTES_USED_TYPE'LAST)+1);

-- Provides for a list of the attributes used which is returned
-- by the INQ_GDP and LOCATOR_ATTRIBUTES_USED functions.

-- CHAR_EXPANSION

type CHAR_EXPANSION is new SCALE_FACTOR range
  SCALE_FACTOR'SAFE_SMALL..SCALE_FACTOR'LAST;

-- Defines a character expansion factor.

-- CHAR_SET

type CHAR_SET is new PHIGS_NATURAL;

-- Provides identifications for the available character
-- sets.  ISO 646 character set maps to value zero.
```

```
-- CHAR_SETS

package CHAR_SETS is new PHIGS_LIST_UTILITIES (        CHAR_SET,
MAX_CHAR_SETS_SUPPORTED);

-- Provides for lists of character set identifications.

-- CHAR_SPACING

type CHAR_SPACING is new SCALE_FACTOR;

-- Defines a character spacing factor.  The factors are
-- unitless.  A positive value indicates the amount of extra
-- space between character boxes in a text string, and a
-- negative value indicates the amount of overlap between
-- character boxes in a text string.

-- CHOICE_PROMPT_ECHO_TYPE

type CHOICE_PROMPT_ECHO_TYPE is new PHIGS_INTEGER;

-- Defines the choice prompt and echo type.

-- CHOICE_PROMPT_ECHO_TYPES

package CHOICE_PROMPT_ECHO_TYPES is
  new PHIGS_LIST_UTILITIES
    (CHOICE_PROMPT_ECHO_TYPE,
     MAX_CHOICE_PROMPT_ECHO_TYPES_SUPPORTED);

-- Provides for lists of choice prompt and echo types.

-- DEVICE_NUMBER

subtype DEVICE_NUMBER is PHIGS_POSITIVE;

--  Defines the base type for input device numbers.

-- CHOICE_DEVICE_NUMBER

type CHOICE_DEVICE_NUMBER is new DEVICE_NUMBER;

-- Provides for choice device identifiers.

-- CHOICE_NUMBER

type CHOICE_NUMBER is new PHIGS_POSITIVE;

-- Defines the choice numbers available on an implementation.
```

```
-- CHOICE_PROMPT

type CHOICE_PROMPT is new OFF_ON;

-- Indicates whether a specified choice prompt is to be displayed
-- or not.

-- CHOICE_PROMPTS

package CHOICE_PROMPTS is
  new PHIGS_LIST_UTILITIES ( CHOICE_PROMPT,
                             MAX_CHOICE_PROMPTS_SUPPORTED);

-- Provides for lists of choice prompts.

-- CHOICE_PROMPT_STRING

type CHOICE_PROMPT_STRING (LENGTH : STRING_SMALL_NATURAL := 0) is
  record
    CONTENTS : PHIGS_STRING (1..LENGTH);
  end record;

-- Provides for a variable length prompt. Objects of this type
-- should be declared unconstrained to allow for dynamic
-- modification of the length.

-- CHOICE_PROMPT_STRING_ARRAY

type CHOICE_PROMPT_STRING_ARRAY is array (PHIGS_POSITIVE range <>)
  of CHOICE_PROMPT_STRING;

-- Provides for an array of prompt strings.

-- CHOICE_PROMPT_STRING_LIST

type CHOICE_PROMPT_STRING_LIST
  (LENGTH : CHOICE_SMALL_NATURAL := 0) is
    record
      LIST : CHOICE_PROMPT_STRING_ARRAY (1..LENGTH);
    end record;

-- Provides for lists of prompt strings.

-- CHOICE_REQUEST_STATUS

type CHOICE_REQUEST_STATUS is (  OK,
                                 NOCHOICE,
                                 NONE);
```

```
-- Defines the status of a choice input operation for the
-- request function.

-- CHOICE_STATUS

subtype CHOICE_STATUS is CHOICE_REQUEST_STATUS range OK..NOCHOICE;

-- Indicates if a choice was made by the operator for the
-- sample, get, and inquiry functions.

-- CLIPPING_INDICATOR

type CLIPPING_INDICATOR is ( CLIP,
                             NOCLIP);

-- Indicates whether or not clipping is to be performed.


-- WEIGHT_LIST_ARRAY

type WEIGHT_LIST_ARRAY is
        array (COLOUR_COMPONENTS range <>) of COLOUR_COEFFICIENT;

-- Provides for a list of weights to be applied during colour mapping.


-- WEIGHT_LIST

type WEIGHT_LIST
        (LENGTH : COLOUR_COMPONENTS := MAX_COLOUR_COEFFICIENTS) is
  record
    WEIGHTS : WEIGHT_LIST_ARRAY(1..LENGTH);
  end record;

-- Provides for a variable length list of weights to be applied during colour mapping.

type ACCESS_WEIGHT_LIST is access WEIGHT_LIST;

-- Provides for pointers to lists of weights.

-- COLOUR_MAPPING_METHOD

type COLOUR_MAPPING_METHOD is new PHIGS_INTEGER;

-- Provides for specifying different methods for performing colour mapping.
```

```
TRUE_COLOUR_MAPPING          : constant COLOUR_MAPPING_METHOD := 1;
PSEUDO_COLOUR_MAPPING        : constant COLOUR_MAPPING_METHOD := 2;
PSEUDO_N_COLOUR_MAPPING      : constant COLOUR_MAPPING_METHOD := 3;

    -- COLOUR_MAPPING_METHODS

    package COLOUR_MAPPING_METHODS is
        new PHIGS_LIST_UTILITIES
            (COLOUR_MAPPING_METHOD,
             MAX_COLOUR_MAPPING_METHODS_SUPPORTED);

    -- Provides for lists of colour mapping methods.

    -- COLOUR_MAPPING_DATA_RECORD

    type COLOUR_MAPPING_DATA_RECORD
      (METHOD : COLOUR_MAPPING_METHOD) is
        record
          case METHOD is

            when TRUE_COLOUR_MAPPING =>
              null;

            when PSEUDO_COLOUR_MAPPING =>
              PSEUDO_COLOUR_MODEL  : COLOUR_MODEL;
              PSEUDO_WEIGHTS       : ACCESS_WEIGHT_LIST;
              PSEUDO_COLOURS       : ACCESS_COLOUR_VALUE_LIST;

            when PSEUDO_N_COLOUR_MAPPING =>
              PSEUDO_N_COLOUR_MODEL  : COLOUR_MODEL;
              PSEUDO_N_COLOURS       : ACCESS_LIST_OF_COLOUR_VALUE_LIST;

            when others =>
              null;

          end case;
        end record;

    -- Provides for specifying values for different methods of colour mapping.
    -- Additional variants may be included when additional registered or unregistered colour
    -- mapping methods are supported by an implementation.

    -- ACCESS_COLOUR_MAPPING_DATA_RECORD

    type ACCESS_COLOUR_MAPPING_DATA_RECORD is
        access COLOUR_MAPPING_DATA_RECORD;

    -- Provides for pointers to colour mapping data records.
```

```
-- COLOUR_MAPPING_INDEX

type COLOUR_MAPPING_INDEX is new PHIGS_NATURAL;

-- Provides for index values for colour mapping representations.

-- COLOUR_MAPPING_INDICES

package COLOUR_MAPPING_INDICES is
        new PHIGS_LIST_UTILITIES
                (COLOUR_MAPPING_INDEX,
                 MAX_COLOUR_MAPPING_INDICES_SUPPORTED);

-- Provides for lists of index values for colour mapping representations.

-- COLOUR_MAPPING_METHOD_FACILITIES

type COLOUR_MAPPING_METHOD_FACILITIES
  (METHOD : COLOUR_MAPPING_METHOD := TRUE_COLOUR_MAPPING) is
    record
      case METHOD is

        when TRUE_COLOUR_MAPPING =>
          NUMBER_TRUE_COLOURS_AVAIL : PHIGS_INTEGER;

        when PSEUDO_COLOUR_MAPPING =>
          NUMBER_PSEUDO_COLORS_AVAIL : PHIGS_INTEGER;

        when PSEUDO_N_COLOUR_MAPPING =>
          null;

        when others =>
          null;

      end case;
    end record;

-- Provides for colour mapping availability information.  Additional variants may be included
-- when additional registered or unregistered colour mapping methods are supported by an
-- implementation.

-- COLOUR_MAPPING_STATE

type COLOUR_MAPPING_STATE
  (METHOD : COLOUR_MAPPING_METHOD := TRUE_COLOUR_MAPPING) is
    record
      case METHOD is
```

```
            when TRUE_COLOUR_MAPPING =>
              null;

            when PSEUDO_COLOUR_MAPPING =>
              null;

            when PSEUDO_N_COLOUR_MAPPING =>
              null;

            when others =>
              null;

          end case;
        end record;

-- Provides for colour mapping state information.  Additional variants may be included
-- when additional registered or unregistered colour mapping methods are supported by an
-- implementation.

-- COMPOSITION_TYPE

type COMPOSITION_TYPE is (  PRECONCATENATE,
                            POSTCONCATENATE,
                            REPLACE);

-- Defines the type of matrix composition allowed between
-- modeling transformations.

-- CONFLICT_RESOLUTION

type CONFLICT_RESOLUTION is ( MAINTAIN,
                              ABANDON,
                              UPDATE);

-- Defines a type for specifying the conflict resolution mode.

-- CONNECTION_ID

type CONNECTION_ID is new PHIGS_STRING;

--  Provides for specifying connection identifiers.

-- CONTROL_FLAG

type CONTROL_FLAG is (  CONDITIONALLY,
                        ALWAYS);

-- The control flag is used to indicate the conditions under
-- which the display surface should be cleared.
```

```
        -- CURVE_APPROX_CRITERIA_TYPE

        type CURVE_APPROX_CRITERIA_TYPE is new PHIGS_INTEGER;

        -- Provides for differentiating types of curve approximation.

        -- CURVE_APPROX_CRITERIA_TYPES

        package CURVE_APPROX_CRITERIA_TYPES is
                new PHIGS_LIST_UTILITIES
                        (CURVE_APPROX_CRITERIA_TYPE,
                         MAX_CURVE_APPROX_CRITERIA_TYPES_SUPPORTED);

        -- Provides for lists of curve approximation criteria.

WS_DEPENDENT_CURVE              : constant CURVE_APPROX_CRITERIA_TYPE := 1;
CONSTANT_SUBDIVISION_CURVE      : constant CURVE_APPROX_CRITERIA_TYPE := 2;
WC_CHORDAL_SIZE_CURVE           : constant CURVE_APPROX_CRITERIA_TYPE := 3;
NPC_CHORDAL_SIZE_CURVE          : constant CURVE_APPROX_CRITERIA_TYPE := 4;
DC_CHORDAL_SIZE_CURVE           : constant CURVE_APPROX_CRITERIA_TYPE := 5;
WC_CHORDAL_DEVIATION_CURVE      : constant CURVE_APPROX_CRITERIA_TYPE := 6;
NPC_CHORDAL_DEVIATION_CURVE     : constant CURVE_APPROX_CRITERIA_TYPE := 7;
DC_CHORDAL_DEVIATION_CURVE      : constant CURVE_APPROX_CRITERIA_TYPE := 8;
WC_RELATIVE_CURVE               : constant CURVE_APPROX_CRITERIA_TYPE := 9;
NPC_RELATIVE_CURVE              : constant CURVE_APPROX_CRITERIA_TYPE := 10;
DC_RELATIVE_CURVE               : constant CURVE_APPROX_CRITERIA_TYPE := 11;

        -- CURVE_APPROXIMATION_DATA_RECORD

        type CURVE_APPROX_DATA_RECORD
          (CRITERIA_TYPE : CURVE_APPROX_CRITERIA_TYPE := WS_DEPENDENT_CURVE) is
            record
              case CRITERIA_TYPE is

                when WS_DEPENDENT_CURVE =>
                  null;

                when CONSTANT_SUBDIVISION_CURVE =>
                  COUNT : PHIGS_INTEGER;

                when WC_CHORDAL_SIZE_CURVE =>
                  WC_CHORDAL_SIZE_VALUE : WC.MAGNITUDE;

                when NPC_CHORDAL_SIZE_CURVE =>
                  NPC_CHORDAL_SIZE_VALUE : NPC.MAGNITUDE;
```

```
            when DC_CHORDAL_SIZE_CURVE =>
              DC_CHORDAL_SIZE_VALUE : DC.MAGNITUDE;

            when WC_CHORDAL_DEVIATION_CURVE =>
              WC_CHORDAL_DEVIATION_VALUE : WC.MAGNITUDE;

            when NPC_CHORDAL_DEVIATION_CURVE =>
              NPC_CHORDAL_DEVIATION_VALUE : NPC.MAGNITUDE;

            when DC_CHORDAL_DEVIATION_CURVE =>
              DC_CHORDAL_DEVIATION_VALUE : DC.MAGNITUDE;

            when WC_RELATIVE_CURVE =>
              WC_RELATIVE_VALUE : WC.RELATIVE_MAGNITUDE;

            when NPC_RELATIVE_CURVE =>
              NPC_RELATIVE_VALUE : NPC.RELATIVE_MAGNITUDE;

            when DC_RELATIVE_CURVE =>
              DC_RELATIVE_VALUE : DC.RELATIVE_MAGNITUDE;

            when others =>
              null;

          end case;
        end record;

-- Provides for specifying values for different methods of curve approximation.
-- Additional variants may be included when additional registered or unregistered curve
-- approximation criteria types are supported by an implementation.


-- CURVE_PLACEMENT_TYPE

type CURVE_PLACEMENT_TYPE is (UNIFORM, NON_UNIFORM);

-- Provides for different ways of placing curves.

-- DC_UNITS

type DC_UNITS is ( METRES,
                   OTHER);

-- Device coordinate units for a particular workstation
-- may be in metres, or some other unit (such as inches).

-- DEFERRAL_MODE

type DEFERRAL_MODE is (  ASAP,
```

```
                                BNIG,
                                BNIL,
                                ASTI,
                                WAIT);

-- Defines the five PHIGS deferral modes.

-- DEPTH_CUE_INDEX

type DEPTH_CUE_INDEX is new PHIGS_NATURAL;

-- Provides for index values for depth cue representations.


-- DEPTH_CUE_INDICES

package DEPTH_CUE_INDICES is
        new PHIGS_LIST_UTILITIES
                (DEPTH_CUE_INDEX,
                 MAX_DEPTH_CUE_INDICES_SUPPORTED);

-- Provides for lists of depth cue indices.

-- DEPTH_CUE_MODE

type DEPTH_CUE_MODE is (SUPPRESSED, ALLOWED);

-- Provides for activation and deactivation of depth cue operations.

--DEPTH_CUE_MODES

package DEPTH_CUE_MODES is
        new PHIGS_LIST_UTILITIES (DEPTH_CUE_MODE, 2);

-- Provides for lists of depth cue indices.


-- DEPTH_CUE_SCALE_FACTOR

subtype DEPTH_CUE_SCALE_FACTOR is SCALE_FACTOR range 0.0 .. 1.0;

-- Provides for specifying scale factors for depth cue operations.


-- DEPTH_CUE_SCALE_FACTORS
```

```
type DEPTH_CUE_SCALE_FACTORS is
  record
    FRONT : DEPTH_CUE_SCALE_FACTOR;
    BACK  : DEPTH_CUE_SCALE_FACTOR;
  end record;

-- Provides for specifying a pair of scale factors for apportioning colours during depth cue
-- operations.


-- RETURN_DEPTH_CUE_INDEX_COUNT

subtype RETURN_DEPTH_CUE_INDEX_COUNT is
            PHIGS_POSITIVE range 2..MAX_DEPTH_CUE_INDICES_SUPPORTED;

-- Provides for restricting the returned count of predefined depth cue indices.


-- DISPLAY_CLASS

type DISPLAY_CLASS is ( VECTOR_DISPLAY,
                        RASTER_DISPLAY,
                        OTHER_DISPLAY);

-- The classification of a workstation of category OUTPUT
-- or OUTIN.

-- DISPLAY_SURFACE_EMPTY

type DISPLAY_SURFACE_EMPTY is ( EMPTY,
                                NOTEMPTY);

-- Indicates whether there is graphical output on the display
-- surface.

-- DYNAMIC_MODIFICATION

type DYNAMIC_MODIFICATION is ( IRG,
                               IMM,
                               CBS);

-- Indicates whether an update to the state list is performed
-- immediately (IMM), requires implicit regeneration (IRG), or
-- can be simulated (CBS).

-- ECHO_SWITCH

type ECHO_SWITCH is ( NOECHO,
                      ECHO);
```

```
-- Indicates whether or not echoing of the measure is
-- performed.

-- EDIT_MODE

type EDIT_MODE is ( INSERT,
                    REPLACE);

-- Defines a type for specifying the mode in which editing
-- takes place.

-- ELEMENT_POSITION

type ELEMENT_POSITION is new PHIGS_INTEGER;

-- Base type for element pointer values and offsets.

-- RETURNED_ELEMENT_POSITION

subtype RETURNED_ELEMENT_POSITION is
  ELEMENT_POSITION range 0..ELEMENT_POSITION'LAST;

-- Defines a type for returning element pointer values.

-- ELEMENT_TYPE

type ELEMENT_TYPE is
        (ALL_ELEMENT_TYPES,
         NIL,

         POLYLINE_3,
         POLYLINE,

         POLYMARKER_3,
         POLYMARKER,

         TEXT_3,
         TEXT,

         ANNOTATION_TEXT_RELATIVE_3,
         ANNOTATION_TEXT_RELATIVE,

         FILL_AREA_3,
         FILL_AREA,
         FILL_AREA_SET_3,
         FILL_AREA_SET,
```

CELL_ARRAY_3,
CELL_ARRAY,

GDP_3,
GDP,

SET_POLYLINE_INDEX,
SET_POLYMARKER_INDEX,
SET_TEXT_INDEX,
SET_INTERIOR_INDEX,
SET_EDGE_INDEX,

SET_LINETYPE,
SET_LINEWIDTH_SCALE_FACTOR,
SET_POLYLINE_COLOUR_INDEX,

SET_MARKER_TYPE,
SET_MARKER_SIZE_SCALE_FACTOR,
SET_POLYMARKER_COLOUR_INDEX,

SET_TEXT_FONT,
SET_TEXT_PRECISION,
SET_CHAR_EXPANSION_FACTOR,
SET_CHAR_SPACING,
SET_TEXT_COLOUR_INDEX,
SET_CHAR_HEIGHT,
SET_CHAR_UP_VECTOR,
SET_TEXT_PATH,
SET_TEXT_ALIGNMENT,

SET_ANNOTATION_TEXT_CHAR_HEIGHT,
SET_ANNOTATION_TEXT_CHAR_UP_VECTOR,
SET_ANNOTATION_TEXT_PATH,
SET_ANNOTATION_TEXT_ALIGNMENT,
SET_ANNOTATION_STYLE,

SET_INTERIOR_STYLE,
SET_INTERIOR_STYLE_INDEX,
SET_INTERIOR_COLOUR_INDEX,

SET_EDGE_FLAG,
SET_EDGETYPE,
SET_EDGEWIDTH_SCALE_FACTOR,
SET_EDGE_COLOUR_INDEX,

SET_PATTERN_SIZE,
SET_PATTERN_REFERENCE_POINT_AND_VECTORS,
SET_PATTERN_REFERENCE_POINT,

ADD_NAMES_TO_SET,
REMOVE_NAMES_FROM_SET,

SET_INDIVIDUAL_ASF,
SET_HLHSR_IDENTIFIER,

SET_LOCAL_TRANSFORMATION_3,
SET_LOCAL_TRANSFORMATION,
SET_GLOBAL_TRANSFORMATION_3,
SET_GLOBAL_TRANSFORMATION,
SET_MODELLING_CLIPPING_VOLUME_3,
SET_MODELLING_CLIPPING_VOLUME,
SET_MODELLING_CLIPPING_INDICATOR,
RESTORE_MODELLING_CLIPPING_VOLUME,
SET_VIEW_INDEX,

EXECUTE_STRUCTURE,

LABEL,
APPLICATION_DATA,
GSE,
SET_PICK_IDENTIFIER,

POLYLINE_SET_3_WITH_COLOUR,
FILL_AREA_SET_3_WITH_DATA,
FILL_AREA_SET_WITH_DATA,
CELL_ARRAY_3_PLUS,
SET_OF_FILL_AREA_SETS_3_WITH_DATA,
SET_OF_FILL_AREA_SETS_WITH_DATA,
TRIANGLE_SET_3_WITH_DATA,
TRIANGLE_SET_WITH_DATA,
TRIANGLE_STRIP_3_WITH_DATA,
TRIANGLE_STRIP_WITH_DATA,
QUADRILATERAL_MESH_3_WITH_DATA,
QUADRILATERAL_MESH_WITH_DATA,
NON_UNIFORM_B_SPLINE_CURVE,
NON_UNIFORM_B_SPLINE_CURVE_WITH_COLOUR,
NON_UNIFORM_B_SPLINE_SURFACE,
NON_UNIFORM_B_SPLINE_SURFACE_WITH_DATA,

SET_DATA_MAPPING_INDEX,
SET_REFLECTANCE_INDEX,

SET_BACK_INTERIOR_INDEX,
SET_BACK_DATA_MAPPING_INDEX,
SET_BACK_REFLECTANCE_INDEX,

SET_PARAMETRIC_SURFACE_INDEX,

```
                SET_POLYLINE_COLOUR,
                SET_POLYLINE_SHADING_METHOD,

                SET_POLYMARKER_COLOUR,

                SET_TEXT_COLOUR,

                SET_FACET_DISTINGUISHING_MODE,
                SET_FACET_CULLING_MODE,

                SET_INTERIOR_COLOUR,
                SET_INTERIOR_SHADING_METHOD,

                SET_DATA_MAPPING_METHOD,
                SET_REFLECTANCE_PROPERTIES,
                SET_REFLECTANCE_MODEL,

                SET_BACK_INTERIOR_STYLE,
                SET_BACK_INTERIOR_STYLE_INDEX,
                SET_BACK_INTERIOR_COLOUR,
                SET_BACK_INTERIOR_SHADING_METHOD,
                SET_BACK_DATA_MAPPING_METHOD,
                SET_BACK_REFLECTANCE_PROPERTIES,
                SET_BACK_REFLECTANCE_MODEL,

                SET_LIGHT_SOURCE_STATE,

                SET_EDGE_COLOUR,

                SET_CURVE_APPROX_CRITERIA,
                SET_SURFACE_APPROX_CRITERIA,
                SET_PARAMETRIC_SURFACE_CHARACTERISTICS,

                SET_RENDERING_COLOUR_MODEL,
                SET_DEPTH_CUE_INDEX,
                SET_COLOUR_MAPPING_INDEX);
```

-- This type lists the element types which exist in PHIGS PLUS.

```
-- ELEMENT_TYPES

package ELEMENT_TYPES is
  new PHIGS_LIST_UTILITIES ( ELEMENT_TYPE,
                             ELEMENT_TYPE'POS(ELEMENT_TYPE'LAST)+1);

-- Provides for lists of element types.

-- ERROR_HANDLING_MODE
```

```
type ERROR_HANDLING_MODE is new OFF_ON;

-- Type for inquiring the error handling mode.

-- ERROR_NUMBER

type ERROR_NUMBER is new PHIGS_INTEGER;

-- Defines the type for error indicator values.

-- ESCAPE_ID

type ESCAPE_ID is new PHIGS_INTEGER;

-- Defines a type for identifying different escapes.

-- ESCAPE_IDS

package ESCAPE_IDS is
  new PHIGS_LIST_UTILITIES ( ESCAPE_ID,
                             MAX_ESCAPE_IDS_SUPPORTED);

-- Provides for lists of Escape ID's.

-- ESCAPE_DATA_RECORD

type ESCAPE_DATA_RECORD (ESCAPE : ESCAPE_ID := 0) is
  record
    case ESCAPE is

      -- For each ESCAPE which needs a record, an implementation
      -- dependent entry is defined.  For others, a null record
      -- is defined.

      when others =>
        null;
    end case;
  end record;

-- Provides for the definition of ESCAPE data records.

-- FACET_CULLING_MODE

type FACET_CULLING_MODE is (  NO_FACET_CULLING,
                              BACKFACING,
                              FRONTFACING);

-- Provides for indicating the type of culling to be applied to facets.
```

```
-- FACET_DISTINGUISHING_MODE

type FACET_DISTINGUISHING_MODE is new OFF_ON;

-- Provides for enabling and disabling facet distinguishing operations.


-- FILE_ID

subtype FILE_ID is PHIGS_STRING;

-- Provides for file identifiers.

-- VARIABLE_FILE_ID

type VARIABLE_FILE_ID (LENGTH : FILE_SMALL_NATURAL := 0) is
  record
    NAME : FILE_ID (1..LENGTH);
  end record;

-- Provides for variable length file identifiers.

-- VARIABLE_FILE_IDS

package VARIABLE_FILE_IDS is
  new PHIGS_LIST_UTILITIES ( VARIABLE_FILE_ID,
                             MAX_FILE_IDS_SUPPORTED);

-- This type is used to define lists of file identifiers.

-- GDP_3_ID

type GDP_3_ID is new PHIGS_INTEGER;

-- Defines a type for selecting a Generalized Drawing Primitive.

-- GDP_3_IDS

package GDP_3_IDS is
  new PHIGS_LIST_UTILITIES ( GDP_3_ID,
                             MAX_GDP_3_IDS_SUPPORTED);

-- Provides for lists of 3D Generalized Drawing Primitive ID's.

-- GDP_3_RECORD

type GDP_3_RECORD (GDP : GDP_3_ID := 0) is
  record
```

```
    case GDP is

      -- For each supported GDP_3 which needs a record, an implementation
      -- dependent entry is defined.  For others, a generic record
      -- is defined.

      when others =>
        GDP_INFO : APPLICATION_DATA_RECORD;
    end case;
  end record;

-- Provides for the definition of GDP_3 data records.

-- GDP_ID

type GDP_ID is new PHIGS_INTEGER;

-- Defines a type for selecting a Generalized Drawing Primitive.

-- GDP_IDS

package GDP_IDS is
  new PHIGS_LIST_UTILITIES ( GDP_ID,
                             MAX_GDP_IDS_SUPPORTED);

-- Provides for lists of Generalized Drawing Primitive ID's.

-- GDP_RECORD

type GDP_RECORD (GDP : GDP_ID := 0) is
  record
    case GDP is

      -- For each supported GDP which needs a record, an implementation
      -- dependent entry is defined.  For others, a generic record
      -- is defined.

      when others =>
        GDP_INFO : APPLICATION_DATA_RECORD;
    end case;
  end record;

-- Provides for the definition of GDP data records.

-- GSE_ID

type GSE_ID is new PHIGS_INTEGER;

--  Defines a type for selecting a Generalized Structure Element.
```

```
-- GSE_IDS

package GSE_IDS is new PHIGS_LIST_UTILITIES ( GSE_ID,
                                              MAX_GSE_IDS_SUPPORTED);

--  Provides for lists of Generalized Structure Element ID's.

-- GSE_RECORD

type GSE_RECORD (GSE : GSE_ID := 0) is
  record
    case GSE is

      -- For each supported GSE which needs a record, an implementation
      -- dependent entry is defined.  For others, a generic record
      -- is defined.

      when others =>
        GSE_INFO : APPLICATION_DATA_RECORD;
    end case;
  end record;

--  Provides for the definition of GSE data records.

-- STYLE_INDEX

type STYLE_INDEX is new PHIGS_INTEGER;

-- Defines an interior style index.

-- HATCH_STYLE

subtype HATCH_STYLE is STYLE_INDEX;

-- Defines the interior hatch styles type.

-- HATCH_STYLES

package HATCH_STYLES is
  new PHIGS_LIST_UTILITIES ( HATCH_STYLE,
                             MAX_HATCH_STYLES_SUPPORTED);

-- Provides for lists of hatch styles.

-- HLHSR_ID

type HLHSR_ID is new PHIGS_INTEGER;
```

```
-- A number used as an attribute for workstation independent
-- HLHSR factors.

-- HLHSR_IDS

package HLHSR_IDS is new PHIGS_LIST_UTILITIES (  HLHSR_ID,
                                                 MAX_HLHSR_IDS_SUPPORTED);

-- Provides for lists of HLHSR identifiers.

-- HLHSR_MODE

type HLHSR_MODE is new PHIGS_INTEGER;

-- A number used for controlling workstation dependent
-- HLHSR factors.

-- HLHSR_MODES

package HLHSR_MODES is
  new PHIGS_LIST_UTILITIES ( HLHSR_MODE,
                             MAX_HLHSR_MODES_SUPPORTED);

-- Provides for lists of HLHSR modes.

-- HORIZONTAL_ALIGNMENT

type HORIZONTAL_ALIGNMENT is ( NORMAL,
                               LEFT,
                               CENTRE,
                               RIGHT);

-- The alignment of the text extent rectangle with respect to
-- horizontal positioning of the text.

-- INPUT_CLASS

type INPUT_CLASS is (  NONE,
                       LOCATOR_INPUT,
                       STROKE_INPUT,
                       VALUATOR_INPUT,
                       CHOICE_INPUT,
                       PICK_INPUT,
                       STRING_INPUT);

-- Defines the input device classifications for workstations
-- of category INPUT or OUTIN.

-- INPUT_QUEUE_CLASS
```

```
subtype INPUT_QUEUE_CLASS is
  INPUT_CLASS range LOCATOR_INPUT .. STRING_INPUT;

-- Defines the input device classifications for workstations
-- of category INPUT and OUTIN that do not return a NONE
-- classification.

-- INPUT_STATUS

type INPUT_STATUS is ( OK,
                       NONE);

-- Defines the status of a locator, stroke, valuator, or
-- string operation.

-- INPUT_STRING

type INPUT_STRING (LENGTH : INPUT_STRING_SMALL_NATURAL := 0) is
  record
    CONTENTS : PHIGS_STRING (1..LENGTH);
  end record;

-- Provides a variable length input string.  Objects of this type
-- should be declared unconstrained to allow for dynamic modification
-- of the length.

-- INTERIOR_INDEX

type INTERIOR_INDEX is new PHIGS_POSITIVE;

-- Defines interior bundle table indices.

-- INTERIOR_INDICES

package INTERIOR_INDICES is
  new PHIGS_LIST_UTILITIES ( INTERIOR_INDEX,
                             MAX_INTERIOR_INDICES_SUPPORTED);

-- Provides for lists of interior bundle table indices.

-- INTERIOR_SHADING_METHOD

type INTERIOR_SHADING_METHOD is new PHIGS_INTEGER;

-- Provides for differentiating methods of shading interiors of enclosed regions.


-- INTERIOR_SHADING_METHODS
```

```
        package INTERIOR_SHADING_METHODS is
                new PHIGS_LIST_UTILITIES
                        (INTERIOR_SHADING_METHOD,
                         MAX_INTERIOR_SHADING_METHODS_SUPPORTED);

        -- Provides for lists of methods of performing interior shading.

NO_INTERIOR_SHADING                 : constant INTERIOR_SHADING_METHOD := 1;
INTERIOR_COLOUR_SHADING             : constant INTERIOR_SHADING_METHOD := 2;
INTERIOR_DATA_SHADING               : constant INTERIOR_SHADING_METHOD := 3;
INTERIOR_DATA_AND_DOT_SHADING       : constant INTERIOR_SHADING_METHOD := 4;
INTERIOR_DATA_AND_NORMAL_SHADING    : constant INTERIOR_SHADING_METHOD := 5;

        -- INTERIOR_STYLE

        type INTERIOR_STYLE is ( HOLLOW,
                                 SOLID,
                                 PATTERN,
                                 HATCH,
                                 EMPTY);

        -- Defines the interior styles for filled areas.

        -- INTERIOR_STYLES

        package INTERIOR_STYLES is
          new PHIGS_LIST_UTILITIES ( INTERIOR_STYLE,
                                     INTERIOR_STYLE'POS(INTERIOR_STYLE'LAST)+1);

        -- Provides for lists of interior styles.

        -- INTERIOR_DATA

        type INTERIOR_DATA is
          record
            STYLE_OF_INTERIOR_ASF  : ASF;
            STYLE_IND_ASF          : ASF;
            INTERIOR_COLOUR_ASF    : ASF;
            INTERIOR_IND           : INTERIOR_INDEX;
            STYLE_OF_INTERIOR      : INTERIOR_STYLE;
            STYLE_IND              : STYLE_INDEX;
            INTERIOR_COLOUR        : COLOUR_INDEX;
          end record;

        -- A record containing information needed to specify the
        -- appearance of a filled area interior.

        -- INVALID_VALUES_INDICATOR
```

```
type INVALID_VALUES_INDICATOR is (   ABSENT,
                                     PRESENT);

-- Indicates whether invalid values are contained in a colour
-- array or matrix.

-- KNOT

subtype KNOT is SPC.MAGNITUDE;

-- Provides for specifying spline knots.


-- LABEL_ID

type LABEL_ID is new PHIGS_INTEGER;

-- Defines a type for specifying structure element labels.

-- LIGHTING_EXPONENT

type LIGHTING_EXPONENT is digits PHIGS_PRECISION;

-- Provides for specifying values for specular lighting exponents.


-- SPREAD_ANGLE

subtype SPREAD_ANGLE is ANGLE range 0.0 .. PHIGS_PI;

-- Provides for specifying the amount of spread of spot light sources.


-- LIGHT_SOURCE_TYPE

type LIGHT_SOURCE_TYPE is new PHIGS_INTEGER;

-- Provides for specifying kinds of light sources.


-- LIGHT_SOURCE_TYPES

package LIGHT_SOURCE_TYPES is
        new PHIGS_LIST_UTILITIES ( LIGHT_SOURCE_TYPE,
                                   MAX_LIGHT_SOURCE_TYPES_SUPPORTED);

-- Provides for lists of light source types.
```

```
AMBIENT_LIGHT                : constant LIGHT_SOURCE_TYPE := 1;
DIRECTIONAL_LIGHT            : constant LIGHT_SOURCE_TYPE := 2;
POSITIONAL_LIGHT             : constant LIGHT_SOURCE_TYPE := 3;
SPOT_LIGHT                   : constant LIGHT_SOURCE_TYPE := 4;

    -- LIGHT_SOURCE_DATA_RECORD

    type LIGHT_SOURCE_DATA_RECORD
      (TYPE_OF_LIGHT : LIGHT_SOURCE_TYPE := AMBIENT_LIGHT) is
        record
          case TYPE_OF_LIGHT is

            when AMBIENT_LIGHT =>
              AMBIENT_COLOUR          : GENERAL_COLOUR;

            when DIRECTIONAL_LIGHT =>
              DIRECTIONAL_COLOUR      : GENERAL_COLOUR;
              DIRECTIONAL_DIRECTION   : WC.VECTOR_3;

            when POSITIONAL_LIGHT =>
              POSITIONAL_COLOUR       : GENERAL_COLOUR;
              POSITIONAL_POSITION     : WC.POINT_3;
              POSITIONAL_ATTENUATION  : ATTENUATION_COEFFICIENTS;

            when SPOT_LIGHT =>
              SPOT_COLOUR             : GENERAL_COLOUR;
              SPOT_POSITION           : WC.POINT_3;
              SPOT_DIRECTION          : WC.VECTOR_3;
              SPOT_CONCENTRATION      : LIGHTING_EXPONENT;
              SPOT_ATTENUATION        : ATTENUATION_COEFFICIENTS;
              SPOT_SPREAD             : SPREAD_ANGLE;

            when others =>
              null;

          end case;
        end record;

    -- Provides for descriptions of light sources.  Additional variants may be included when
    -- additional registered or unregistered light source types are supported by an
    -- implementation.


    -- LIGHT_SOURCE_INDEX

    type LIGHT_SOURCE_INDEX is new PHIGS_POSITIVE;

    -- Provides for identifying light sources.
```

```
-- LIGHT_SOURCE_INDICES

package LIGHT_SOURCE_INDICES is
        new PHIGS_LIST_UTILITIES ( LIGHT_SOURCE_INDEX,
                                   MAX_LIGHT_SOURCE_INDICES_SUPPORTED);

-- Provides for lists of light sources.

-- RETURN_LIGHT_SOURCE_COUNT

subtype RETURN_LIGHT_SOURCE_COUNT is
              PHIGS_POSITIVE range 2..PHIGS_POSITIVE'last;

-- Provides for restricting spline orders to those acceptable for returning values of the
-- facilities.

-- POLYLINE_INDEX

type POLYLINE_INDEX is new PHIGS_POSITIVE;

-- Defines the range of polyline indices.

-- POLYLINE_INDICES

package POLYLINE_INDICES is
  new PHIGS_LIST_UTILITIES ( POLYLINE_INDEX,
                             MAX_POLYLINE_INDICES_SUPPORTED);

-- Provides for lists of polyline indices.

-- LINETYPE

type LINETYPE is new PHIGS_INTEGER;

-- Defines the types of line styles provided by PHIGS.

-- LINETYPES

package LINETYPES is new PHIGS_LIST_UTILITIES (   LINETYPE,
                                                  MAX_LINETYPES_SUPPORTED);

-- Provides for lists of linetypes.

-- LINEWIDTH

type LINEWIDTH is new SCALE_FACTOR range 0.0..SCALE_FACTOR'LAST;

-- The width of a line is indicated by a scale factor.
```

```
-- LINE_DATA

type LINE_DATA is
  record
    TYPE_OF_LINE_ASF   : ASF;
    WIDTH_ASF          : ASF;
    LINE_COLOUR_ASF    : ASF;
    POLYLINE_IND       : POLYLINE_INDEX;
    TYPE_OF_LINE       : LINETYPE;
    WIDTH              : LINEWIDTH;
    LINE_COLOUR        : COLOUR_INDEX;
  end record;

-- A record containing information needed to specify the
-- appearance of a line for input data records.

-- EDGE_FLAG

type EDGE_FLAG is new OFF_ON;

-- Defines a type for indicating whether or not to depict edges
-- on fill area set and related primitives.

-- CURVE VISIBILITY FLAG

subtype CURVE_VISIBILITY_FLAG is EDGE_FLAG;

-- Provides control over display of curves on spline surfaces


-- EDGE_FLAG_LIST

type EDGE_FLAG_LIST is array (POSITIVE range <>) of EDGE_FLAG;

-- Provides for arrays of edge flags.  These will be used with various "with Data" primitives.


-- ACCESS_EDGE_FLAG_LIST

type ACCESS_EDGE_FLAG_LIST is access EDGE_FLAG_LIST;

-- Provides for pointers to lists of edge flags.


-- LIST_OF_EDGE_FLAG_LIST

type LIST_OF_EDGE_FLAG_LIST is
  array (POSITIVE range <>) of ACCESS_EDGE_FLAG_LIST;
```

```
-- Provides for pointers to lists of lists of edge flags.


-- ACCESS_LIST_OF_EDGE_FLAG_LIST

type ACCESS_LIST_OF_EDGE_FLAG_LIST is access LIST_OF_EDGE_FLAG_LIST;

-- Provides for pointers to lists of lists of edge flags.


-- LIST_OF_LIST_OF_EDGE_FLAG_LIST

type LIST_OF_LIST_OF_EDGE_FLAG_LIST is
  array (POSITIVE range <>) of ACCESS_LIST_OF_EDGE_FLAG_LIST;

-- Provides for pointers to lists of lists of edge flags.


-- ACCESS_LIST_OF_LIST_OF_EDGE_FLAG_LIST

type ACCESS_LIST_OF_LIST_OF_EDGE_FLAG_LIST is
        access LIST_OF_LIST_OF_EDGE_FLAG_LIST;

-- Provides for pointers to lists of lists of edge flag lists.


-- EDGE_FLAG_PAIR

type EDGE_FLAG_PAIR is array (1..2) of EDGE_FLAG;

-- Provides for pairs of edge flags.  These will be used with "Quadrilateral Mesh with Data"
-- primitives.


-- ARRAY_OF_EDGE_FLAG_PAIR

type ARRAY_OF_EDGE_FLAG_PAIR is
        array ( POSITIVE range <>,
                POSITIVE range <>) of EDGE_FLAG_PAIR;

-- Provides for two-dimensional arrays of edge flag pairs.  These will be used with
-- "Quadrilateral Mesh with Data" primitives.


-- ACCESS_ARRAY_OF_EDGE_FLAG_PAIR
```

```
type ACCESS_ARRAY_OF_EDGE_FLAG_PAIR is access ARRAY_OF_EDGE_FLAG_PAIR;

-- Provides for access variable to two-dimensional arrays of edge flag pairs.  These will be
-- used when inquiring structure elements which are "Quadrilateral Mesh with Data"
-- primitives.


-- EDGE_FLAG_TRIPLET

type EDGE_FLAG_TRIPLET is array (1..3) of EDGE_FLAG;

-- Provides for triplets of edge flags.  These will be used with "Triangle Set with Data"
-- primitives.


-- LIST_OF_EDGE_FLAG_TRIPLET

type LIST_OF_EDGE_FLAG_TRIPLET is
  array (POSITIVE range <>) of EDGE_FLAG_TRIPLET;

-- Provides for lists of triplets of edge flags.


-- ACCESS_LIST_OF_EDGE_FLAG_TRIPLET

type ACCESS_LIST_OF_EDGE_FLAG_TRIPLET is
  access LIST_OF_EDGE_FLAG_TRIPLET;

-- Provides for pointers to lists of triplets of edge flags.


-- EDGE_INDEX

type EDGE_INDEX is new PHIGS_POSITIVE;

-- Defines the edge bundle table indices.

-- EDGE_INDICES

package EDGE_INDICES is
  new PHIGS_LIST_UTILITIES ( EDGE_INDEX,
                             MAX_EDGE_INDICES_SUPPORTED);

-- Provides for a list of edge indices.

-- EDGETYPE

type EDGETYPE is new PHIGS_INTEGER;
```

```
-- Defines a type for describing the form of edges.

-- EDGETYPES

package EDGETYPES is
  new PHIGS_LIST_UTILITIES ( EDGETYPE,
                             MAX_EDGETYPES_SUPPORTED);

-- Provides for lists of edgetypes.

-- EDGEWIDTH

type EDGEWIDTH is new SCALE_FACTOR range 0.0..SCALE_FACTOR'LAST;

-- Defines a type for describing the width scale factor of edges.

-- EDGE_DATA

type EDGE_DATA is
  record
    FLAG_ASF              : ASF;
    TYPE_OF_EDGE_ASF      : ASF;
    EDGEWIDTH_SF_ASF      : ASF;
    EDGE_COLOUR_ASF       : ASF;
    FLAG                  : EDGE_FLAG;
    TYPE_OF_EDGE          : EDGETYPE;
    EDGEWIDTH_SF          : EDGEWIDTH;
    EDGE_COLOUR           : COLOUR_INDEX;
  end record;

-- A record containing information needed to specify the
-- appearance of a fill area set edge.

-- LOCATOR_DEVICE_NUMBER

type LOCATOR_DEVICE_NUMBER is new DEVICE_NUMBER;

-- Provides for locator device identifiers.

-- LOCATOR_PROMPT_ECHO_TYPE

type LOCATOR_PROMPT_ECHO_TYPE is new PHIGS_INTEGER;

-- Defines the locator prompt and echo types supported by
-- the implementation.

-- LOCATOR_PROMPT_ECHO_TYPES

package LOCATOR_PROMPT_ECHO_TYPES is
```

```
  new PHIGS_LIST_UTILITIES
    (LOCATOR_PROMPT_ECHO_TYPE,
     MAX_LOCATOR_PROMPT_ECHO_TYPES_SUPPORTED);

-- Provides for lists of locator prompt and echo types.

-- POLYMARKER_INDEX

type POLYMARKER_INDEX is new PHIGS_POSITIVE;

-- Defines the range of polymarker bundle table indices.

-- POLYMARKER_INDICES

package POLYMARKER_INDICES is
  new PHIGS_LIST_UTILITIES ( POLYMARKER_INDEX,
                             MAX_POLYMARKER_INDICES_SUPPORTED);

-- Provides for lists of polymarker indices.

-- MARKER_SIZE

type MARKER_SIZE is new SCALE_FACTOR;

-- The size of a marker is indicated by a scale factor.

-- MARKER_TYPE

type MARKER_TYPE is new PHIGS_INTEGER;

-- Defines the type for markers provided by PHIGS.

-- MARKER_TYPES

package MARKER_TYPES is
  new PHIGS_LIST_UTILITIES ( MARKER_TYPE,
                             MAX_MARKER_TYPES_SUPPORTED);

-- Provides for lists of marker types.

-- MARKER_DATA

type MARKER_DATA is
  record
    TYPE_OF_MARKER_ASF  : ASF;
    SIZE_ASF            : ASF;
    MARKER_COLOUR_ASF   : ASF;
    POLYMARKER_IND      : POLYMARKER_INDEX;
    TYPE_OF_MARKER      : MARKER_TYPE;
```

```
    SIZE                    : MARKER_SIZE;
    MARKER_COLOUR           : COLOUR_INDEX;
  end record;

-- A record containing information needed to specify the
-- appearance of a marker.

-- MEMORY_UNITS

type MEMORY_UNITS is range 0..MAX_MEMORY_UNITS;

-- Defines the type for units of memory that may be allocated
-- by PHIGS.

-- METAFILE_ITEM_TYPE

type METAFILE_ITEM_TYPE is
  new PHIGS_NATURAL
    range MIN_METAFILE_ITEM_TYPE..MAX_METAFILE_ITEM_TYPE;

--  The type of an item contained in a metafile.

-- METAFILE_ITEM_LENGTH

type METAFILE_ITEM_LENGTH is
  new PHIGS_NATURAL range 0..MAX_METAFILE_ITEM_LENGTH;

--  The length of an item contained in a metafile.

-- MODELLING_CLIP_OPERATION_TYPE

type MODELLING_CLIP_OPERATION_TYPE is new PHIGS_INTEGER;

--  This type provides for specifying modelling clip operation
--  types.

-- MODELLING_CLIP_OPERATION_TYPES

package MODELLING_CLIP_OPERATION_TYPES is
  new PHIGS_LIST_UTILITIES
    (MODELLING_CLIP_OPERATION_TYPE,
     MAX_MODELLING_CLIP_OPERATION_TYPES_SUPPORTED);

--  This type provides for lists of modelling clip operation types.

-- MODIFICATION_MODE

type MODIFICATION_MODE is ( NIVE,
                            UWOR,
```

```
                                        UQUM);

-- Defines type for specifying modification modes.

-- MORE_EVENTS

type MORE_EVENTS is ( NOMORE,
                      MORE);

-- Indicates whether more simultaneous events are contained in
-- the input queue.

-- NAME_SET

subtype NAME_SET is PHIGS_NAME_SET_FACILITY.NAME_SET;

-- Used to define name sets. The internal structure of
-- a name set is implementation-dependent. "Build" functions
-- contained in the package PHIGS_NAME_SET_FACILITIES shall be
-- used to create and manipulate name sets (See 5.14.4).

-- NAME_SET_ACCEPTANCE

subtype NAME_SET_ACCEPTANCE is
  PHIGS_NAME_SET_FACILITY.NAME_SET_ACCEPTANCE;

-- Used to indicate the results of applying a name set
-- against a filter pair. See 5.14.4.

-- NAME

subtype NAME is PHIGS_NAME_SET_FACILITY.NAME;

-- Provides for names for each of the name set classes.
-- See 5.14.4.

-- NAMES

package NAMES renames PHIGS_NAME_SET_FACILITY.NAMES;

-- Provides for lists of name set classes. See 5.14.4.

-- NAME_SET_FILTER

subtype NAME_SET_FILTER is PHIGS_NAME_SET_FACILITY.NAME_SET_FILTER;

-- Used to define name set pairs used as filters.
-- See 5.14.4.
```

```
-- NAME_SET_FILTERS

package NAME_SET_FILTERS is
  new PHIGS_LIST_UTILITIES
    (NAME_SET_FILTER,
     MAX_NAME_SET_FILTER_LIST_LENGTH_SUPPORTED);

-- Used to provide lists of name set filters.
-- See 5.14.4.

-- NAME_SET_MEMBERSHIP

subtype NAME_SET_MEMBERSHIP is
  PHIGS_NAME_SET_FACILITY.NAME_SET_MEMBERSHIP;

-- Provides for indicating whether a class is contained in a
-- name set. See 5.14.4.

-- OPEN_STRUCTURE_STATUS

type OPEN_STRUCTURE_STATUS is ( NONE,
                                OPEN);

-- Returns the status of the open structure.

-- OPERATING_MODE

type OPERATING_MODE is ( REQUEST_MODE,
                         SAMPLE_MODE,
                         EVENT_MODE);

-- Defines the operating modes of an input device.

-- PARAMETRIC_SURFACE_CHARACTERISTIC

type PARAMETRIC_SURFACE_CHARACTERISTIC is new PHIGS_INTEGER;

-- Provides for differentiating representations of parametric surfaces.


-- PARAMETRIC_SURFACE_CHARACTERISTICS

package PARAMETRIC_SURFACE_CHARACTERISTICS is
        new PHIGS_LIST_UTILITIES
              (PARAMETRIC_SURFACE_CHARACTERISTIC,
               MAX_PARAMETRIC_SURFACE_CHARACTERISTICS_SUPPORTED);

-- Provides for lists of parametric surface characteristics types.
```

```
NO_PARAMETRIC_SURFACE
                              : constant PARAMETRIC_SURFACE_CHARACTERISTIC := 1;
WS_DEPENDENT_PARAMETRIC_SURFACE
                              : constant PARAMETRIC_SURFACE_CHARACTERISTIC := 2;
ISOPARAMETRIC_CURVES    : constant PARAMETRIC_SURFACE_CHARACTERISTIC := 3;
MC_LEVEL_CURVES         : constant PARAMETRIC_SURFACE_CHARACTERISTIC := 4;
WC_LEVEL_CURVES         : constant PARAMETRIC_SURFACE_CHARACTERISTIC := 5;

    -- PARAMETRIC_SURFACE_DATA_RECORD

    type PARAMETRIC_SURFACE_DATA_RECORD
      (CHARACTERISTIC : PARAMETRIC_SURFACE_CHARACTERISTIC
                              := NO_PARAMETRIC_SURFACE) is
        record
          case CHARACTERISTIC is

            when NO_PARAMETRIC_SURFACE =>
              null;

            when WS_DEPENDENT_PARAMETRIC_SURFACE =>
              null;

            when ISOPARAMETRIC_CURVES =>
              CURVE_PLACEMENT : CURVE_PLACEMENT_TYPE;
              UCOUNT          : PHIGS_NATURAL;
              VCOUNT          : PHIGS_NATURAL;

            when MC_LEVEL_CURVES =>
              MC_LEVEL_ORIGIN      : MC.POINT_3;
              MC_LEVEL_DIRECTION   : MC.VECTOR_3;
              MC_LEVEL_PARAMETERS  : MC.MAGNITUDE_LIST;

            when WC_LEVEL_CURVES =>
              WC_LEVEL_ORIGIN      : WC.POINT_3;
              WC_LEVEL_DIRECTION   : WC.VECTOR_3;
              WC_LEVEL_PARAMETERS  : WC.MAGNITUDE_LIST;

            when others =>
              null;

          end case;
        end record;

    -- Provides for specifying values for different parametric surface characteristics.  Additional
    -- variants may be included when additional registered or unregistered parametric surface
    -- characteristics are supported by an implementation.


    -- PARAMETRIC_SURFACE_INDEX
```

```
type PARAMETRIC_SURFACE_INDEX is new PHIGS_NATURAL;

-- Provides for index values for parametric surface representations.


-- PARAMETRIC_SURFACE_INDICES

package PARAMETRIC_SURFACE_INDICES is
        new PHIGS_LIST_UTILITIES
                (PARAMETRIC_SURFACE_INDEX,
                 MAX_PARAMETRIC_SURFACE_INDICES_SUPPORTED);

-- Provides for lists of parametric surface indices.


-- PATH_DEPTH

subtype PATH_DEPTH is
  PHIGS_NATURAL range 0..MAX_PATH_DEPTH_SUPPORTED;

-- Defines a type for specifying the depth of a traversal path.

-- PATH_ORDER

type PATH_ORDER is (  TOPFIRST,
                      BOTTOMFIRST);

-- Provides for specifying the order of pick paths.

-- PATTERN_INDEX

subtype PATTERN_INDEX is STYLE_INDEX range 1..STYLE_INDEX'LAST;

-- Defines the range of pattern table indices.

-- PATTERN_INDICES

package PATTERN_INDICES is
  new PHIGS_LIST_UTILITIES ( PATTERN_INDEX,
                             MAX_PATTERN_INDICES_SUPPORTED);

-- Provides for lists of pattern table indices.

-- PICK_DEVICE_NUMBER

type PICK_DEVICE_NUMBER is new DEVICE_NUMBER;

-- Provides for pick device identifiers.
```

```
-- PICK_ID

type PICK_ID is new PHIGS_INTEGER;

-- Defines the range of pick identifiers available on an
-- implementation.

-- PICK_IDS

package PICK_IDS is
  new PHIGS_LIST_UTILITIES ( PICK_ID,
                             MAX_PICK_IDS_SUPPORTED);

-- Provides for a list of pick identifiers.

-- PICK_PROMPT_ECHO_TYPE

type PICK_PROMPT_ECHO_TYPE is new PHIGS_INTEGER;

-- Defines the pick prompt and echo type.

-- PICK_PROMPT_ECHO_TYPES

package PICK_PROMPT_ECHO_TYPES is
  new PHIGS_LIST_UTILITIES ( PICK_PROMPT_ECHO_TYPE,
                             MAX_PICK_PROMPT_ECHO_TYPES_SUPPORTED);

-- Provides for lists of pick prompt and echo types.

-- PICK_REQUEST_STATUS

type PICK_REQUEST_STATUS is ( OK,
                              NOPICK,
                              NONE);

-- Defines the status of a pick input operation for the
-- request function.

-- PICK_STATUS

subtype PICK_STATUS is PICK_REQUEST_STATUS range OK..NOPICK;

-- Defines the status of pick input operation for the sample
-- get, and inquiry functions.

-- REFLECTANCE_MODEL
```

```
        type REFLECTANCE_MODEL is new PHIGS_INTEGER;

        -- Provides for selecting the type of reflectance calculations.


        -- REFLECTANCE_MODELS

        package REFLECTANCE_MODELS is
                new PHIGS_LIST_UTILITIES
                        (REFLECTANCE_MODEL,
                         MAX_REFLECTANCE_MODELS_SUPPORTED);

        -- Provides for lists of reflectance models.

NO_REFLECTANCE                          : constant REFLECTANCE_MODEL := 1;
AMBIENT_REFLECTANCE                     : constant REFLECTANCE_MODEL := 2;
AMBIENT_DIFFUSE_REFLECTANCE             : constant REFLECTANCE_MODEL := 3;
AMBIENT_DIFFUSE_SPECULAR_REFLECTANCE    : constant REFLECTANCE_MODEL := 4;

        -- REFLECTANCE_COEFFICIENT

        type REFLECTANCE_COEFFICIENT is digits PHIGS_PRECISION range 0.0 .. 1.0;

        -- Provides for specifying values for lighting reflectance coefficients.


        -- REFLECTANCE_INDEX

        type REFLECTANCE_INDEX is new PHIGS_NATURAL;

        -- Provides for index values for reflectance representations.


        -- REFLECTANCE_INDICES

        package REFLECTANCE_INDICES is
                new PHIGS_LIST_UTILITIES ( REFLECTANCE_INDEX,
                                           MAX_REFLECTANCE_INDICES_SUPPORTED);

        -- Provides for lists of reflectance indices.


        -- REFLECTANCE_PROPERTIES_TYPE
```

```
type REFLECTANCE_PROPERTIES_TYPE is new PHIGS_INTEGER;

-- Provides for selecting the type of reflectance properties.

-- REFLECTANCE_PROPERTIES_TYPES

package REFLECTANCE_PROPERTIES_TYPES is
        new PHIGS_LIST_UTILITIES
                (REFLECTANCE_PROPERTIES_TYPE,
                 MAX_REFLECTANCE_PROPERTIES_TYPES_SUPPORTED);

-- Provides for lists of reflectance properties types.

SIMPLE_REFLECTANCE  : constant REFLECTANCE_PROPERTIES_TYPE := 1;

-- REFLECTANCE_PROPERTIES_DATA_RECORD

type REFLECTANCE_PROPERTIES_DATA_RECORD
      (TYPE_OF_REFLECTANCE_PROPERTIES
              : REFLECTANCE_PROPERTIES_TYPE := SIMPLE_REFLECTANCE) is
    record
      case TYPE_OF_REFLECTANCE_PROPERTIES is

        when SIMPLE_REFLECTANCE =>
          AMBIENT_REFLECTANCE   : REFLECTANCE_COEFFICIENT;
          DIFFUSE_REFLECTANCE   : REFLECTANCE_COEFFICIENT;
          SPECULAR_REFLECTANCE  : REFLECTANCE_COEFFICIENT;
          SPECULAR_COLOUR       : GENERAL_COLOUR;
          SPECULAR_EXPONENT     : LIGHTING_EXPONENT;

        when others =>
          null;

      end case;
    end record;

-- Provides for specifying values for different types of reflectance properties.  Additional
-- variants may be included when additional registered or unregistered reflectance
-- properties types are supported by an implementation.


-- STRUCTURE_ID

type STRUCTURE_ID is new PHIGS_NATURAL;

-- Defines a type for structure identifiers.

-- STRUCTURE_IDS
```

```
package STRUCTURE_IDS is
  new PHIGS_LIST_UTILITIES ( STRUCTURE_ID,
                             MAX_STRUCTURE_IDS_SUPPORTED);

-- Provides for lists of structure identifiers.

-- PICK_PATH_ENTRY

type PICK_PATH_ENTRY is
  record
    STRUCTURE_IDENTIFIER : STRUCTURE_ID;
    PICK_IDENTIFIER      : PICK_ID;
    ELEMENT_NUMBER       : ELEMENT_POSITION;
  end record;

-- Defines the entries of a pick path.

-- PICK_PATH_ARRAY

type PICK_PATH_ARRAY is array (PATH_DEPTH range <>) of
  PICK_PATH_ENTRY;

-- Provides for the specification of an unconstrained array of pick
-- path entries.

-- PICK_PATH

type PICK_PATH (DEPTH : PATH_DEPTH := 0) is
  record
    PATH : PICK_PATH_ARRAY (0..DEPTH);
  end record;

-- Provides for the specification of a pick path.

-- POLYLINE_FILL_AREA_CONTROL_FLAG

type POLYLINE_FILL_AREA_CONTROL_FLAG is ( POLYLINE,
                                          FILL_AREA,
                                          FILL_AREA_SET);

-- Provides for indicating the type of primitive used
-- to produce locator prompt and echo type 5.

-- POLYLINE_SHADING_METHOD

type POLYLINE_SHADING_METHOD is new PHIGS_NATURAL;

-- Provides for values for distinguishing between various methods of polyline shading.
```

```
        -- POLYLINE_SHADING_METHODS

        package POLYLINE_SHADING_METHODS is
                new PHIGS_LIST_UTILITIES
                        (POLYLINE_SHADING_METHOD,
                         MAX_POLYLINE_SHADING_METHODS_SUPPORTED);

        -- Provides for lists of polyline shading methods.

NO_POLYLINE_SHADING             : constant POLYLINE_SHADING_METHOD := 1;
POLYLINE_COLOUR_SHADING         : constant POLYLINE_SHADING_METHOD := 2;

        -- PROJECTION_TYPE

        type PROJECTION_TYPE is (  PARALLEL,
                                   PERSPECTIVE);

        -- Defines the type of projections supported by PHIGS.

        -- RASTER_UNITS

        type RASTER_UNITS is new PHIGS_POSITIVE;

        -- Defines the range of raster units.

        -- RASTER_UNIT_SIZE_3

        type RASTER_UNIT_SIZE_3 is
          record
            X  : RASTER_UNITS;
            Y  : RASTER_UNITS;
            Z  : RASTER_UNITS;
          end record;

        -- Defines the size of an object in raster units on a raster
        -- device.

        -- RASTER_UNIT_SIZE_2

        type RASTER_UNIT_SIZE_2 is
          record
            X  : RASTER_UNITS;
            Y  : RASTER_UNITS;
          end record;

        -- Defines the size of an object in raster units on a raster
        -- device.
```

```
-- REFERENCE_HANDLING_FLAG

type REFERENCE_HANDLING_FLAG is (  DELETE,
                                   KEEP);

-- Provides for specification of the handling of deleted
-- structure networks.

-- REFERENCE_PATH_ENTRY

type REFERENCE_PATH_ENTRY is
  record
    STRUCTURE_IDENTIFIER : STRUCTURE_ID;
    ELEMENT_NUMBER       : ELEMENT_POSITION;
  end record;

--  Defines the entries of a reference path.

-- REFERENCE_PATH_ARRAY

type REFERENCE_PATH_ARRAY is array (PATH_DEPTH range <>) of
  REFERENCE_PATH_ENTRY;

--  Provides for the specification of a basic reference path.

-- REFERENCE_PATH

type REFERENCE_PATH (DEPTH : PATH_DEPTH := 0) is
  record
    PATH : REFERENCE_PATH_ARRAY (0..DEPTH);
  end record;

--  Provides for the specification of a variable size
--  reference path.

-- REFERENCE_PATHS

package REFERENCE_PATHS is
  new PHIGS_LIST_UTILITIES ( REFERENCE_PATH,
                             MAX_REFERENCE_PATHS_SUPPORTED);

--  Provides for the specification of lists of reference
--  paths.

-- RELATIVE_PRIORITY

type RELATIVE_PRIORITY is (  HIGHER,
                             LOWER);
```

```
-- Indicates the relative input priority between two views.

-- RETURN_VALUE_TYPE

type RETURN_VALUE_TYPE is (   SET,
                              REALIZED);

-- Indicates whether the returned values should be as they
-- were set by the program or as they were actually realized
-- on the device.

-- SEARCH_DIRECTION

type SEARCH_DIRECTION is (  BACKWARD,
                            FORWARD);

--  This type provides for indicating search direction.

-- SEARCH_STATUS_INDICATOR

type SEARCH_STATUS_INDICATOR is (   FAILURE,
                                    SUCCESS);

--  This type provides for indicating the result of a search.

-- STRING_DEVICE_NUMBER

type STRING_DEVICE_NUMBER is new DEVICE_NUMBER;

--  Provides for string device identifiers.

-- STRING_PROMPT_ECHO_TYPE

type STRING_PROMPT_ECHO_TYPE is new PHIGS_INTEGER;

-- Defines the string prompt and echo types.

-- STRING_PROMPT_ECHO_TYPES

package STRING_PROMPT_ECHO_TYPES is
  new PHIGS_LIST_UTILITIES ( STRING_PROMPT_ECHO_TYPE,
                             MAX_STRING_PROMPT_ECHO_TYPES_SUPPORTED);

-- Provides for lists of string prompt and echo types.

-- STROKE DEVICE NUMBER

type STROKE_DEVICE_NUMBER is new DEVICE_NUMBER;
```

```
-- Provides for stroke device identifiers.

-- STROKE_PROMPT_ECHO_TYPE

type STROKE_PROMPT_ECHO_TYPE is new PHIGS_INTEGER;

-- Defines the stroke prompt and echo types.

-- STROKE_PROMPT_ECHO_TYPES

package STROKE_PROMPT_ECHO_TYPES is
  new PHIGS_LIST_UTILITIES ( STROKE_PROMPT_ECHO_TYPE,
                             MAX_STROKE_PROMPT_ECHO_TYPES_SUPPORTED);

-- Provides for lists of stroke prompt and echo types

-- STRUCTURE_ELEMENT_TYPE

subtype STRUCTURE_ELEMENT_TYPE is
  ELEMENT_TYPE range NIL..ELEMENT_TYPE'LAST;

--  Provides for identifying the types of individual
--  structure elements.

-- STRUCTURE NETWORK SOURCE

type STRUCTURE_NETWORK_SOURCE is ( CSS,
                                   ARCHIVE);

--  Provides for indicating source for check of conflicting
--  structure identifiers.

-- SURFACE_APPROX_CRITERIA_TYPE

type SURFACE_APPROX_CRITERIA_TYPE is new PHIGS_INTEGER;

-- Provides for differentiating methods of surface approximation.


-- SURFACE_APPROX_CRITERIA_TYPES

package SURFACE_APPROX_CRITERIA_TYPES is
        new PHIGS_LIST_UTILITIES
                (SURFACE_APPROX_CRITERIA_TYPE,
                 MAX_SURFACE_APPROX_CRITERIA_TYPES_SUPPORTED);

-- Provides for lists of surface approximation criteria types.
```

```
WS_DEPENDENT_SURFACE              : constant SURFACE_APPROX_CRITERIA_TYPE := 1;
CONSTANT_SUBDIVISION_SURFACE_BETWEEN_KNOTS
                                  : constant SURFACE_APPROX_CRITERIA_TYPE := 2;
WC_CHORDAL_SIZE_SURFACE           : constant SURFACE_APPROX_CRITERIA_TYPE := 3;
NPC_CHORDAL_SIZE_SURFACE          : constant SURFACE_APPROX_CRITERIA_TYPE := 4;
DC_CHORDAL_SIZE_SURFACE           : constant SURFACE_APPROX_CRITERIA_TYPE := 5;
WC_PLANAR_DEVIATION_SURFACE       : constant SURFACE_APPROX_CRITERIA_TYPE := 6;
NPC_PLANAR_DEVIATION_SURFACE      : constant SURFACE_APPROX_CRITERIA_TYPE := 7;
DC_PLANAR_DEVIATION_SURFACE       : constant SURFACE_APPROX_CRITERIA_TYPE := 8;
WC_RELATIVE_SURFACE               : constant SURFACE_APPROX_CRITERIA_TYPE := 9;
NPC_RELATIVE_SURFACE              : constant SURFACE_APPROX_CRITERIA_TYPE := 10;
DC_RELATIVE_SURFACE               : constant SURFACE_APPROX_CRITERIA_TYPE := 11;

    -- SURFACE_APPROX_DATA_RECORD

    type SURFACE_APPROX_DATA_RECORD
      (CRITERIA_TYPE : SURFACE_APPROX_CRITERIA_TYPE
                     := WS_DEPENDENT_SURFACE) is
        record
          case CRITERIA_TYPE is

            when WS_DEPENDENT_SURFACE =>
              null;

            when CONSTANT_SUBDIVISION_SURFACE_BETWEEN_KNOTS =>
              CONSTANT_SUBDIVISION_UCOUNT : PHIGS_INTEGER;
              CONSTANT_SUBDIVISION_VCOUNT : PHIGS_INTEGER;

            when WC_CHORDAL_SIZE_SURFACE =>
              WC_CHORDAL_SIZE_UVALUE : WC.MAGNITUDE;
              WC_CHORDAL_SIZE_VVALUE : WC.MAGNITUDE;

            when NPC_CHORDAL_SIZE_SURFACE =>
              NPC_CHORDAL_SIZE_UVALUE : NPC.MAGNITUDE;
              NPC_CHORDAL_SIZE_VVALUE : NPC.MAGNITUDE;

            when DC_CHORDAL_SIZE_SURFACE =>
              DC_CHORDAL_SIZE_UVALUE : DC.MAGNITUDE;
              DC_CHORDAL_SIZE_VVALUE : DC.MAGNITUDE;

            when WC_PLANAR_DEVIATION_SURFACE =>
              WC_PLANAR_DEVIATION_VALUE : WC.MAGNITUDE;

            when NPC_PLANAR_DEVIATION_SURFACE =>
              NPC_PLANAR_DEVIATION_VALUE : NPC.MAGNITUDE;

            when DC_PLANAR_DEVIATION_SURFACE =>
              DC_PLANAR_DEVIATION_VALUE : DC.MAGNITUDE;
```

```
            when WC_RELATIVE_SURFACE =>
              WC_RELATIVE_VALUE : WC.RELATIVE_MAGNITUDE;

            when NPC_RELATIVE_SURFACE =>
              NPC_RELATIVE_VALUE : NPC.RELATIVE_MAGNITUDE;

            when DC_RELATIVE_SURFACE =>
              DC_RELATIVE_VALUE : DC.RELATIVE_MAGNITUDE;

            when others =>
              null;

          end case;
        end record;

-- Provides for specifying values for different methods of surface approximation.
-- Additional variants may be included when additional registered or unregistered surface
-- approximation criteria types are supported by an implementation.


-- DISPLAY_PRIORITY

type DISPLAY_PRIORITY is digits PHIGS_PRECISION range 0.0..1.0;

-- Defines type for specifying structure display priority.

-- STRUCTURE_STATE

type STRUCTURE_STATE is ( STCL,
                          STOP);

-- The type used to return the structure state value.

-- STRUCTURE_STATUS

type STRUCTURE_STATUS is ( NON_EXISTENT,
                           EMPTY,
                           NOTEMPTY);

--  The type used to return the structure status.

-- POSTED_STRUCTURE

type POSTED_STRUCTURE is
  record
    STRUCTURE_IDENTIFIER : STRUCTURE_ID;
    PRIORITY             : DISPLAY_PRIORITY;
  end record;
```

```
-- Defines a data type for describing posted structure networks.

-- POSTED_STRUCTURES

package POSTED_STRUCTURES is
  new PHIGS_LIST_UTILITIES ( POSTED_STRUCTURE,
                             MAX_POSTED_STRUCTURES_SUPPORTED);

-- Provides for the specification of lists of posted structure
-- networks.

-- SUBPROGRAM_NAME

subtype SUBPROGRAM_NAME is PHIGS_STRING;

-- Defines the name of a PHIGS function detecting an error.

-- SYSTEM_STATE

type SYSTEM_STATE is ( PHCL,
                       PHOP);

-- The type used to return the system state value.

-- VERTICAL_ALIGNMENT

type VERTICAL_ALIGNMENT is ( NORMAL,
                             TOP,
                             CAP,
                             HALF,
                             BASE,
                             BOTTOM);

-- The alignment of the text extent parallelogram with respect
-- to the vertical positioning of the text.

-- TEXT_ALIGNMENT

type TEXT_ALIGNMENT is
  record
    HORIZONTAL  : HORIZONTAL_ALIGNMENT;
    VERTICAL    : VERTICAL_ALIGNMENT;
  end record;

-- The type of the attribute controlling the positioning of
-- the text extent parallelogram in relation to the text
-- position, having horizontal and vertical components as
-- defined above.
```

```
-- TEXT_FONT

type TEXT_FONT is new PHIGS_INTEGER;

-- Defines the types of fonts provided by the implementation.

-- TEXT_PRECISION

type TEXT_PRECISION is ( STRING_PRECISION,
                         CHAR_PRECISION,
                         STROKE_PRECISION);

-- The precision with which text appears.

-- TEXT_FONT_PRECISION

type TEXT_FONT_PRECISION is
  record
    FONT      : TEXT_FONT;
    PRECISION : TEXT_PRECISION;
  end record;

-- This type defines a record describing the text font and
-- precision supported.

-- TEXT_FONT_PRECISIONS

package TEXT_FONT_PRECISIONS is
  new PHIGS_LIST_UTILITIES ( TEXT_FONT_PRECISION,
                             MAX_TEXT_FONT_PRECISION_PAIRS_SUPPORTED);

-- Provides for lists of text font and precision pairs.

-- TEXT_INDEX

type TEXT_INDEX is new PHIGS_POSITIVE;

-- Defines the range of text bundle table indices.

-- TEXT_INDICES

package TEXT_INDICES is
  new PHIGS_LIST_UTILITIES ( TEXT_INDEX,
                             MAX_TEXT_INDICES_SUPPORTED);

-- Provides for lists of text indices.

-- TEXT_PATH
```

```
type TEXT_PATH is ( RIGHT,
                    LEFT,
                    UP,
                    DOWN);

-- The direction taken by a text string.

-- TRANSFORMATION_MATRIX_2

type TRANSFORMATION_MATRIX_2 is array (1..3, 1..3) of MC_TYPE;

-- For 2D modeling and view transformation matrices.

-- TRANSFORMATION_MATRIX_3

type TRANSFORMATION_MATRIX_3 is array (1..4, 1..4) of MC_TYPE;

-- For 3D modeling and view transformation matrices.

-- TRIMCURVE_ORDER

subtype TRIMCURVE_ORDER is SPLINE_ORDER range 2..SPLINE_ORDER'last;

-- Provides for restricting spline orders to those acceptable for trimming curves.


-- TRIMCURVE

type TRIMCURVE( CRITERIA_TYPE : CURVE_APPROX_CRITERIA_TYPE
                                := CONSTANT_SUBDIVISION_CURVE;
                RATIONALITY   : SPLINE_RATIONALITY := RATIONAL;
                KNOT_COUNT    : SMALL_NATURAL := 4;
                LENGTH        : VERTEX_SET_DIMENSION := 2) is
  record
    CRITERIA_FOR_CURVE_APPROX
                  : CURVE_APPROX_DATA_RECORD (CRITERIA_TYPE);
    FLAG          : CURVE_VISIBILITY_FLAG;
    ORDER         : TRIMCURVE_ORDER;
    KNOTS         : KNOT_VECTOR(KNOT_COUNT);
    CONTROL_POINTS : SPC.CONTROL_POINT_LIST_2(RATIONALITY, LENGTH);
    LIMITS        : SPC.RANGE_OF_MAGNITUDES;
  end record;

 -- Provides for specifying minimum and maximum values of spline parameters.


 -- TRIMCURVE_LIST
```

```
type TRIMCURVE_LIST is array (POSITIVE range <>) of TRIMCURVE;

-- Provides for arrays of trimming curves.  These will be used with NURB surface primitives.


-- ACCESS_TRIMCURVE_LIST

type ACCESS_TRIMCURVE_LIST is access TRIMCURVE_LIST;

-- Provides for pointers to lists of trimming curves.


-- LIST_OF_TRIMCURVE_LIST

type LIST_OF_TRIMCURVE_LIST is
  array (POSITIVE range <>) of ACCESS_TRIMCURVE_LIST;

-- Provides for pointers to lists of lists of trimming curves.


-- ACCESS_LIST_OF_TRIMCURVE_LIST

type ACCESS_LIST_OF_TRIMCURVE_LIST is access LIST_OF_TRIMCURVE_LIST;

-- Provides for pointers to lists of lists of edge flags.


-- RETURN_TRIMCURVE_ORDER

subtype RETURN_TRIMCURVE_ORDER is
          TRIMCURVE_ORDER range 4..TRIMCURVE_ORDER'last;

-- Provides for restricting trimcurve orders to those acceptable for returning values of the
-- facilities.


-- TRUNCATION_METHOD

type TRUNCATION_METHOD is ( HEAD,
                            TAIL);

-- Provides for specifying if the head or tail of a reference path should be truncated.

-- UPDATE_REGENERATION_FLAG

type UPDATE_REGENERATION_FLAG is ( PERFORM,
                                   POSTPONE);

-- Flag indicating regeneration action on display.
```

```
-- UPDATE_STATE

type UPDATE_STATE is ( NOTPENDING,
                       PENDING);

-- Indicates whether or not a workstation transformation
-- change has been requested and not yet provided.

-- VALUATOR_DEVICE_NUMBER

type VALUATOR_DEVICE_NUMBER is new DEVICE_NUMBER;

-- Provides for valuator device identifiers.

-- VALUATOR_PROMPT_ECHO_TYPE

type VALUATOR_PROMPT_ECHO_TYPE is new PHIGS_INTEGER;

-- Defines the possible range of valuator prompt and
-- echo types.

-- VALUATOR_PROMPT_ECHO_TYPES

package VALUATOR_PROMPT_ECHO_TYPES is
  new PHIGS_LIST_UTILITIES
    (VALUATOR_PROMPT_ECHO_TYPE,
     MAX_VALUATOR_PROMPT_ECHO_TYPES_SUPPORTED);

-- Provides for lists of valuator prompt and echo types.

-- VALUATOR_VALUE

type VALUATOR_VALUE is digits PHIGS_PRECISION;

-- Defines the range of accuracy of valuator input values on
-- an implementation.

-- EVENT_DEVICE_NUMBER

type EVENT_DEVICE_NUMBER (CLASS : INPUT_CLASS := NONE) is
  record
    case CLASS is

      when NONE =>
        null;

      when LOCATOR_INPUT =>
        LOCATOR_EVENT_DEVICE : LOCATOR_DEVICE_NUMBER;
```

```
        when STROKE_INPUT =>
          STROKE_EVENT_DEVICE : STROKE_DEVICE_NUMBER;

        when VALUATOR_INPUT =>
          VALUATOR_EVENT_DEVICE : VALUATOR_DEVICE_NUMBER;

        when CHOICE_INPUT =>
          CHOICE_EVENT_DEVICE : CHOICE_DEVICE_NUMBER;

        when PICK_INPUT =>
          PICK_EVENT_DEVICE : PICK_DEVICE_NUMBER;

        when STRING_INPUT =>
          STRING_EVENT_DEVICE : STRING_DEVICE_NUMBER;

      end case;
    end record;

-- Provides for returning any class of device number from the
-- event queue.

-- EVENT_QUEUE_DEVICE_NUMBER

type EVENT_QUEUE_DEVICE_NUMBER
  (CLASS : INPUT_QUEUE_CLASS := LOCATOR_INPUT) is
    record
      case CLASS is

        when LOCATOR_INPUT =>
          LOCATOR_EVENT_DEVICE : LOCATOR_DEVICE_NUMBER;

        when STROKE_INPUT =>
          STROKE_EVENT_DEVICE : STROKE_DEVICE_NUMBER;

        when VALUATOR_INPUT =>
          VALUATOR_EVENT_DEVICE : VALUATOR_DEVICE_NUMBER;

        when CHOICE_INPUT =>
          CHOICE_EVENT_DEVICE : CHOICE_DEVICE_NUMBER;

        when PICK_INPUT =>
          PICK_EVENT_DEVICE : PICK_DEVICE_NUMBER;

        when STRING_INPUT =>
          STRING_EVENT_DEVICE : STRING_DEVICE_NUMBER;

      end case;
    end record;
```

```
-- Allows EVENT_DEVICE_NUMBERs which cannot have value NONE.  This is
-- used when returning only real input classes as causes of event
-- queue overflow and for flushing the queue.

-- ACCESS_COLOUR_MATRIX

type ACCESS_COLOUR_MATRIX is access COLOUR_MATRIX;

-- Provides for returning variable sized matrices containing colour
-- indices corresponding to a cell array or pattern array.

-- VARIABLE_CONNECTION_ID

type VARIABLE_CONNECTION_ID (LENGTH : STRING_SMALL_NATURAL := 0) is
  record
    CONNECT : CONNECTION_ID (1..LENGTH);
  end record;

-- Defines a variable length connection identifier for

-- ACCESS_STRING

type ACCESS_STRING is access PHIGS_STRING;

--  Defines a variable length string needed to store strings in
--  structure element records.

-- VERTEX_INDEX

type VERTEX_INDEX is new PHIGS_NATURAL range 1..MAX_VERTICES_SUPPORTED;

-- Provides for indices into arrays of vertices.  These will be used with various "with Data"
-- primitives.

-- VERTEX_INDEX_LIST

type VERTEX_INDEX_LIST is array (POSITIVE range <>) of VERTEX_INDEX;

-- Provides for arrays of indices.  These will be used with various "with Data" primitives.


-- ACCESS_VERTEX_INDEX_LIST

type ACCESS_VERTEX_INDEX_LIST is access VERTEX_INDEX_LIST;

-- Provides for pointers to lists of vertex indices.

-- LIST_OF_VERTEX_INDEX_LIST
```

```
type LIST_OF_VERTEX_INDEX_LIST is
  array (POSITIVE range <>) of ACCESS_VERTEX_INDEX_LIST;

-- Provides for lists of lists of vertex indices.

-- ACCESS_LIST_OF_VERTEX_INDEX_LIST

type ACCESS_LIST_OF_VERTEX_INDEX_LIST is
  access LIST_OF_VERTEX_INDEX_LIST;

-- Provides for pointers to lists of lists of vertex indices.

-- LIST_OF_LIST_OF_VERTEX_INDEX_LIST

type LIST_OF_LIST_OF_VERTEX_INDEX_LIST is
  array (POSITIVE range <>) of ACCESS_LIST_OF_VERTEX_INDEX_LIST;

-- Provides for pointers to lists of lists of vertex indices.

-- ACCESS_LIST_OF_LIST_OF_VERTEX_INDEX_LIST

type ACCESS_LIST_OF_LIST_OF_VERTEX_INDEX_LIST is
        access LIST_OF_LIST_OF_VERTEX_INDEX_LIST;

-- Provides for pointers to lists of lists of lists of indices.

-- VERTEX_INDEX_TRIPLET

subtype VERTEX_INDEX_TRIPLET is VERTEX_INDEX_LIST(1..3);

-- Provides for triplets of indices.  These will be used with "Triangle Set with Data"
-- primitives.

-- LIST_OF_VERTEX_INDEX_TRIPLET

type LIST_OF_VERTEX_INDEX_TRIPLET is
  array (POSITIVE range <>) of VERTEX_INDEX_TRIPLET;

-- Provides for lists of triplets of vertex indices.


-- ACCESS_LIST_OF_VERTEX_INDEX_TRIPLET

type ACCESS_LIST_OF_VERTEX_INDEX_TRIPLET is
  access LIST_OF_VERTEX_INDEX_TRIPLET;

-- Provides for pointers to lists of triplets of vertex indices.

-- VIEW_INDEX
```

```
type VIEW_INDEX is new PHIGS_NATURAL;

-- Defines a type for specifying view indices.

-- VIEW_INDICES

package VIEW_INDICES is
  new PHIGS_LIST_UTILITIES ( VIEW_INDEX,
                             MAX_VIEW_INDICES_SUPPORTED);

-- Provides for lists of view indices.

-- VISUAL REPRESENTATION STATE

type VISUAL_REPRESENTATION_STATE is ( CORRECT,
                                      DEFERRED,
                                      SIMULATED);

-- Specifies state of image on display surface.

-- WS_CATEGORY

type WS_CATEGORY is ( OUTPUT,
                      INPUT,
                      OUTIN,
                      MO,
                      MI);

-- Type for PHIGS workstation categories.

-- WS_DEPENDENCY

type WS_DEPENDENCY is ( WS_INDEPENDENT,
                        WS_DEPENDENT);

-- Type for indications of workstation dependency.

-- WS_DEPENDENCIES

package WS_DEPENDENCIES is
  new PHIGS_LIST_UTILITIES ( WS_DEPENDENCY,
                             MAX_GSE_IDS_SUPPORTED);

-- Provides for lists of workstation dependencies.

-- WS_ID

type WS_ID is new PHIGS_POSITIVE;
```

```
-- Defines the range of workstation identifiers.

-- WS_IDS

package WS_IDS is new PHIGS_LIST_UTILITIES ( WS_ID,
                                             MAX_OPEN_WS_SUPPORTED);

-- Provides for lists of workstation identifiers.

-- WS_STATE

type WS_STATE is ( WSCL,
                   WSOP);

-- The type used to return the workstation state value.

-- WS_TYPE

type WS_TYPE is new PHIGS_POSITIVE;

-- Range of values corresponding to valid workstation types.
-- Constants specifying names for the various types of
-- workstations should be provided by an implementation.

-- WS_TYPES

package WS_TYPES is new PHIGS_LIST_UTILITIES ( WS_TYPE,
                                               MAX_WS_TYPES_SUPPORTED);

-- Provides for lists of workstation types.

-- STRUCTURE_ELEMENT_RECORD

type STRUCTURE_ELEMENT_RECORD
  (ELEMENT_TYPE : STRUCTURE_ELEMENT_TYPE := NIL) is
   record
      case ELEMENT_TYPE is

         -- The empty element

         when NIL =>
           null;

         -- PHIGS Primitive Elements

         when POLYLINE_3 =>
           POLYLINE_3_POINTS : MC.ACCESS_POINT_LIST_3;

         when POLYLINE =>
           POLYLINE_POINTS : MC.ACCESS_POINT_LIST_2;
```

```
when POLYMARKER_3 =>
  POLYMARKER_3_POINTS : MC.ACCESS_POINT_LIST_3;

when POLYMARKER =>
  POLYMARKER_POINTS : MC.ACCESS_POINT_LIST_2;

when TEXT_3 =>
  TEXT_3_POINT : MC.POINT_3;
  TEXT_DIRECTION_VECTORS : MC.VECTOR_PAIR_3;
  TEXT_3_CHAR_STRING : ACCESS_STRING;

when TEXT =>
  TEXT_POINT : MC.POINT_2;
  TEXT_CHAR_STRING : ACCESS_STRING;

when ANNOTATION_TEXT_RELATIVE_3 =>
  ANNOTATION_TEXT_RELATIVE_3_REF_POINT : MC.POINT_3;
  ANNOTATION_TEXT_RELATIVE_3_OFFSET : NPC.POINT_3;
  ANNOTATION_TEXT_3_CHAR_STRING : ACCESS_STRING;

when ANNOTATION_TEXT_RELATIVE =>
  ANNOTATION_TEXT_RELATIVE_REF_POINT : MC.POINT_2;
  ANNOTATION_TEXT_RELATIVE_OFFSET : NPC.POINT_2;
  ANNOTATION_TEXT_CHAR_STRING : ACCESS_STRING;

when FILL_AREA_3 =>
  FILL_AREA_3_POINTS : MC.ACCESS_POINT_LIST_3;

when FILL_AREA =>
  FILL_AREA_POINTS : MC.ACCESS_POINT_LIST_2;

when FILL_AREA_SET_3 =>
  FILL_AREA_SET_3_POINTS : MC.ACCESS_LIST_OF_POINT_LIST_3;

when FILL_AREA_SET =>
  FILL_AREA_SET_POINTS : MC.ACCESS_LIST_OF_POINT_LIST_2;

when CELL_ARRAY_3 =>
  CORNER_P_3 : MC.POINT_3;
  CORNER_Q_3 : MC.POINT_3;
  CORNER_R_3 : MC.POINT_3;
  CELL_ARRAY_3_CELLS : ACCESS_COLOUR_MATRIX;

when CELL_ARRAY =>
  CORNER_P : MC.POINT_2;
  CORNER_Q : MC.POINT_2;
  CELL_ARRAY_CELLS : ACCESS_COLOUR_MATRIX;
```

```
when GDP_3 =>
  GDP_3_POINTS : MC.ACCESS_POINT_LIST_3;
  GDP_3_DATA : GDP_3_RECORD;

when GDP =>
  GDP_POINTS : MC.ACCESS_POINT_LIST_2;
  GDP_DATA : GDP_RECORD;

-- PHIGS Bundle Index Elements

when SET_POLYLINE_INDEX =>
  POLYLINE_IND : POLYLINE_INDEX;

when SET_POLYMARKER_INDEX =>
  POLYMARKER_IND : POLYMARKER_INDEX;

when SET_TEXT_INDEX =>
  TEXT_IND : TEXT_INDEX;

when SET_INTERIOR_INDEX =>
  INTERIOR_IND : INTERIOR_INDEX;

when SET_EDGE_INDEX =>
  EDGE_IND : EDGE_INDEX;

-- PHIGS Individual Aspect Elements

when SET_LINETYPE =>
  TYPE_OF_LINE : LINETYPE;

when SET_LINEWIDTH_SCALE_FACTOR =>
  LINEWIDTH_SF : LINEWIDTH;

when SET_POLYLINE_COLOUR_INDEX =>
  LINE_COLOUR : COLOUR_INDEX;

when SET_MARKER_TYPE =>
  TYPE_OF_MARKER : MARKER_TYPE;

when SET_MARKER_SIZE_SCALE_FACTOR =>
  SIZE : MARKER_SIZE;

when SET_POLYMARKER_COLOUR_INDEX =>
  MARKER_COLOUR : COLOUR_INDEX;

when SET_TEXT_FONT =>
  FONT : TEXT_FONT;

when SET_TEXT_PRECISION =>
  PRECISION : TEXT_PRECISION;
```

```
when SET_CHAR_EXPANSION_FACTOR =>
  EXPANSION : CHAR_EXPANSION;

when SET_CHAR_SPACING =>
  SPACING : CHAR_SPACING;

when SET_TEXT_COLOUR_INDEX =>
  TEXT_COLOUR : COLOUR_INDEX;

when SET_CHAR_HEIGHT =>
  HEIGHT : MC.MAGNITUDE;

when SET_CHAR_UP_VECTOR =>
  CHAR_UP_VECTOR : MC.VECTOR_2;

when SET_TEXT_PATH =>
  PATH : TEXT_PATH;

when SET_TEXT_ALIGNMENT =>
  ALIGNMENT : TEXT_ALIGNMENT;

when SET_ANNOTATION_TEXT_CHAR_HEIGHT =>
  ANNOTATION_HEIGHT : NPC.MAGNITUDE;

when SET_ANNOTATION_TEXT_CHAR_UP_VECTOR =>
  ANNOTATION_CHAR_UP_VECTOR : NPC.VECTOR_2;

when SET_ANNOTATION_TEXT_PATH =>
  ANNOTATION_PATH : TEXT_PATH;

when SET_ANNOTATION_TEXT_ALIGNMENT =>
  ANNOTATION_ALIGNMENT : TEXT_ALIGNMENT;

when SET_ANNOTATION_STYLE =>
  STYLE_OF_ANNOTATION : ANNOTATION_STYLE;

when SET_INTERIOR_STYLE =>
  STYLE_OF_INTERIOR : INTERIOR_STYLE;

when SET_INTERIOR_STYLE_INDEX =>
  STYLE_IND : STYLE_INDEX;

when SET_INTERIOR_COLOUR_INDEX =>
  INTERIOR_COLOUR : COLOUR_INDEX;

when SET_EDGE_FLAG =>
  FLAG : EDGE_FLAG;

when SET_EDGETYPE =>
  TYPE_OF_EDGE : EDGETYPE;
```

```
when SET_EDGEWIDTH_SCALE_FACTOR =>
  EDGEWIDTH_SF : EDGEWIDTH;

when SET_EDGE_COLOUR_INDEX =>
  EDGE_COLOUR : COLOUR_INDEX;

-- PHIGS Pattern Attribute Elements

when SET_PATTERN_SIZE =>
  PATTERN_SIZE : MC.SIZE_2;

when SET_PATTERN_REFERENCE_POINT_AND_VECTORS =>
  PATTERN_REFERENCE_POINT_3 : MC.POINT_3;
  PATTERN_REFERENCE_VECTORS : MC.VECTOR_PAIR_3;

when SET_PATTERN_REFERENCE_POINT =>
  PATTERN_REFERENCE_POINT : MC.POINT_2;

-- PHIGS Name Set Elements

when ADD_NAMES_TO_SET =>
  NAMES_TO_ADD : NAME_SET;

when REMOVE_NAMES_FROM_SET =>
  NAMES_TO_REMOVE : NAME_SET;

-- PHIGS ASF Elements

when SET_INDIVIDUAL_ASF =>
  ATTRIBUTE_ID : ASPECT;
  SOURCE_FLAG : ASF;

-- PHIGS HLHSR Elements

when SET_HLHSR_IDENTIFIER =>
  HLHSR_IDENTIFIER : HLHSR_ID;

-- PHIGS Transformation Elements

when SET_LOCAL_TRANSFORMATION_3 =>
  LOCAL_MATRIX_3 : TRANSFORMATION_MATRIX_3;
  HOW_APPLIED_3 : COMPOSITION_TYPE;

when SET_LOCAL_TRANSFORMATION =>
  LOCAL_MATRIX : TRANSFORMATION_MATRIX_2;
  HOW_APPLIED : COMPOSITION_TYPE;

when SET_GLOBAL_TRANSFORMATION_3 =>
  GLOBAL_MATRIX_3 : TRANSFORMATION_MATRIX_3;
```

```
when SET_GLOBAL_TRANSFORMATION =>
  GLOBAL_MATRIX : TRANSFORMATION_MATRIX_2;

when SET_MODELLING_CLIPPING_VOLUME_3 =>
  MODELLING_CLIPPING_OPERATOR_3
                        : MODELLING_CLIP_OPERATION_TYPE;
  MODELLING_CLIPPING_LIMITS_3
                        : MC.ACCESS_HALF_SPACE_LIST_3;

when SET_MODELLING_CLIPPING_VOLUME =>
  MODELLING_CLIPPING_OPERATOR
                        : MODELLING_CLIP_OPERATION_TYPE;
  MODELLING_CLIPPING_LIMITS : MC.ACCESS_HALF_SPACE_LIST_2;

when SET_MODELLING_CLIPPING_INDICATOR =>
  MODELLING_CLIPPING_INDICATOR : CLIPPING_INDICATOR;

when RESTORE_MODELLING_CLIPPING_VOLUME =>
  null;

when SET_VIEW_INDEX =>
  VIEW_IND : VIEW_INDEX;

-- PHIGS Invocation Elements

when EXECUTE_STRUCTURE =>
  STRUCTURE_IDENTIFIER : STRUCTURE_ID;

-- PHIGS Structure Content Identification Elements

when LABEL =>
  LABEL_IDENTIFIER : LABEL_ID;

when APPLICATION_DATA =>
  DATA : APPLICATION_DATA_RECORD;

when GSE =>
  GSE_DATA : GSE_RECORD;

when SET_PICK_IDENTIFIER =>
  PICK_IDENTIFIER : PICK_ID;

-- PHIGS PLUS Primitive Elements

when POLYLINE_SET_3_WITH_COLOUR =>
  POLYLINE_SET_3_VERTICES
                        : MC.ACCESS_LIST_OF_VERTEX_COLOUR_LIST_SET_3;
```

```
when FILL_AREA_SET_3_WITH_DATA =>
  FILL_AREA_SET_3_FACET : MC.ACCESS_FACET_DATA_SET;
  FILL_AREA_SET_3_EDGES : ACCESS_LIST_OF_EDGE_FLAG_LIST;
  FILL_AREA_SET_3_VERTICES
                        : MC.ACCESS_LIST_OF_VERTEX_DATA_LIST_SET_3;

when FILL_AREA_SET_WITH_DATA =>
  FILL_AREA_SET_FACET : MC.ACCESS_FACET_DATA_SET;
  FILL_AREA_SET_EDGES : ACCESS_LIST_OF_EDGE_FLAG_LIST;
  FILL_AREA_SET_VERTICES
                        : MC.ACCESS_LIST_OF_VERTEX_DATA_LIST_SET_2;

when CELL_ARRAY_3_PLUS =>
  CORNER_P_3_PLUS : MC.POINT_3;
  CORNER_Q_3_PLUS : MC.POINT_3;
  CORNER_R_3_PLUS : MC.POINT_3;
  CELL_ARRAY_3_PLUS_CELLS : ACCESS_COLOUR_VALUE_ARRAY;

when SET_OF_FILL_AREA_SETS_3_WITH_DATA =>
  SET_OF_FILL_AREA_SETS_3_FACETS
                        : MC.ACCESS_FACET_DATA_LIST_SET;
  SET_OF_FILL_AREA_SETS_3_EDGES
                        : ACCESS_LIST_OF_LIST_OF_EDGE_FLAG_LIST;
  SET_OF_FILL_AREA_SETS_3_VERTICES
                        : MC.ACCESS_VERTEX_DATA_LIST_SET_3;
  SET_OF_FILL_AREA_SETS_3_INDICES
                        : ACCESS_LIST_OF_LIST_OF_VERTEX_INDEX_LIST;

when SET_OF_FILL_AREA_SETS_WITH_DATA =>
  SET_OF_FILL_AREA_SETS_FACETS : MC.ACCESS_FACET_DATA_LIST_SET;
  SET_OF_FILL_AREA_SETS_EDGES
                        : ACCESS_LIST_OF_LIST_OF_EDGE_FLAG_LIST;
  SET_OF_FILL_AREA_SETS_VERTICES
                        : MC.ACCESS_VERTEX_DATA_LIST_SET_2;
  SET_OF_FILL_AREA_SETS_INDICES
                        : ACCESS_LIST_OF_LIST_OF_VERTEX_INDEX_LIST;

when TRIANGLE_SET_3_WITH_DATA =>
  TRIANGLE_SET_3_FACETS : MC.ACCESS_FACET_DATA_LIST_SET;
  TRIANGLE_SET_3_EDGES : ACCESS_LIST_OF_EDGE_FLAG_TRIPLET;
  TRIANGLE_SET_3_VERTICES : MC.ACCESS_VERTEX_DATA_LIST_SET_3;
  TRIANGLE_SET_3_INDICES
                        : ACCESS_LIST_OF_VERTEX_INDEX_TRIPLET;

when TRIANGLE_SET_WITH_DATA =>
  TRIANGLE_SET_FACETS : MC.ACCESS_FACET_DATA_LIST_SET;
  TRIANGLE_SET_EDGES : ACCESS_LIST_OF_EDGE_FLAG_TRIPLET;
  TRIANGLE_SET_VERTICES : MC.ACCESS_VERTEX_DATA_LIST_SET_2;
```

```
    TRIANGLE_SET_INDICES
                        : ACCESS_LIST_OF_VERTEX_INDEX_TRIPLET;

  when TRIANGLE_STRIP_3_WITH_DATA =>
    TRIANGLE_STRIP_3_FACETS : MC.ACCESS_FACET_DATA_LIST_SET;
    TRIANGLE_STRIP_3_EDGES : ACCESS_EDGE_FLAG_LIST;
    TRIANGLE_STRIP_3_VERTICES
                        : MC.ACCESS_VERTEX_DATA_LIST_SET_3;

  when TRIANGLE_STRIP_WITH_DATA =>
    TRIANGLE_STRIP_FACETS : MC.ACCESS_FACET_DATA_LIST_SET;
    TRIANGLE_STRIP_EDGES : ACCESS_EDGE_FLAG_LIST;
    TRIANGLE_STRIP_VERTICES : MC.ACCESS_VERTEX_DATA_LIST_SET_2;

  when QUADRILATERAL_MESH_3_WITH_DATA =>
    QUADRILATERAL_MESH_3_FACETS
                        : MC.ACCESS_FACET_DATA_ARRAY_SET;
    QUADRILATERAL_MESH_3_EDGES
                        : ACCESS_ARRAY_OF_EDGE_FLAG_PAIR;
    QUADRILATERAL_MESH_3_VERTICES
                        : MC.ACCESS_VERTEX_DATA_ARRAY_SET_3;

  when QUADRILATERAL_MESH_WITH_DATA =>
    QUADRILATERAL_MESH_FACETS
                        : MC.ACCESS_FACET_DATA_ARRAY_SET;
    QUADRILATERAL_MESH_EDGES
                        : ACCESS_ARRAY_OF_EDGE_FLAG_PAIR;
    QUADRILATERAL_MESH_VERTICES
                        : MC.ACCESS_VERTEX_DATA_ARRAY_SET_2;

  when NON_UNIFORM_B_SPLINE_CURVE =>
    NURB_CURVE_SPLINE : MC.ACCESS_CURVE_GEOMETRYSPLINE_3;
    NURB_CURVE_LIMITS : SPC.RANGE_OF_MAGNITUDES;

  when NON_UNIFORM_B_SPLINE_CURVE_WITH_COLOUR =>
    NURB_CURVE_COLOUR_GEOMETRYSPLINE
                        : MC.ACCESS_CURVE_GEOMETRYSPLINE_3;
    NURB_CURVE_COLOUR_LIMITS : SPC.RANGE_OF_MAGNITUDES;
    NURB_CURVE_COLOUR_COLOURSPLINE
                        : ACCESS_CURVE_COLOURSPLINE;

  when NON_UNIFORM_B_SPLINE_SURFACE =>
    NURB_SURFACE_GEOMETRYSPLINE
                        : MC.ACCESS_SURFACE_GEOMETRYSPLINE;
    NURB_SURFACE_TRIM_CURVES
                        : ACCESS_LIST_OF_TRIMCURVE_LIST;
```

```
when NON_UNIFORM_B_SPLINE_SURFACE_WITH_DATA =>
  NURB_SURFACE_DATA_GEOMETRYSPLINE
                          : MC.ACCESS_SURFACE_GEOMETRYSPLINE;
  NURB_SURFACE_DATA_TRIM_CURVES
                          : ACCESS_LIST_OF_TRIMCURVE_LIST;
  NURB_SURFACE_DATA_COLOURSPLINE
                          : ACCESS_SURFACE_COLOURSPLINE;
  NURB_SURFACE_DATA_DATASPLINES
                                : ACCESS_DATASPLINE_LIST;

-- PHIGS PLUS Bundle Index Elements

when SET_DATA_MAPPING_INDEX =>
  DATA_MAPPING_IND : DATA_MAPPING_INDEX;

when SET_REFLECTANCE_INDEX =>
  REFLECTANCE_IND : REFLECTANCE_INDEX;

when SET_BACK_INTERIOR_INDEX =>
  BACK_INTERIOR_IND : INTERIOR_INDEX;

when SET_BACK_DATA_MAPPING_INDEX =>
  BACK_DATA_MAPPING_IND : DATA_MAPPING_INDEX;

when SET_BACK_REFLECTANCE_INDEX =>
  BACK_REFLECTANCE_IND : REFLECTANCE_INDEX;

when SET_PARAMETRIC_SURFACE_INDEX =>
  PARAMETRIC_SURFACE_IND : PARAMETRIC_SURFACE_INDEX;

-- PHIGS PLUS Individual Aspect Elements

when SET_POLYLINE_COLOUR =>
  GENERAL_POLYLINE_COLOUR : GENERAL_COLOUR;

when SET_POLYLINE_SHADING_METHOD =>
  POLYLINE_SHADING : POLYLINE_SHADING_METHOD;

when SET_POLYMARKER_COLOUR =>
  GENERAL_POLYMARKER_COLOUR : GENERAL_COLOUR;

when SET_TEXT_COLOUR =>
  GENERAL_TEXT_COLOUR : GENERAL_COLOUR;

when SET_FACET_DISTINGUISHING_MODE =>
  FACET_DISTINGUISHING : FACET_DISTINGUISHING_MODE;

when SET_FACET_CULLING_MODE =>
  FACET_CULLING : FACET_CULLING_MODE;
```

```
when SET_INTERIOR_COLOUR =>
  GENERAL_INTERIOR_COLOUR : GENERAL_COLOUR;

when SET_INTERIOR_SHADING_METHOD =>
  INTERIOR_SHADING : INTERIOR_SHADING_METHOD;

when SET_DATA_MAPPING_METHOD =>
  METHOD_OF_DATA_MAPPING : DATA_MAPPING_DATA_RECORD;

when SET_REFLECTANCE_PROPERTIES =>
  REFLECTANCE_PROPERTIES : REFLECTANCE_PROPERTIES_DATA_RECORD;

when SET_REFLECTANCE_MODEL =>
  MODEL_OF_REFLECTANCE : REFLECTANCE_MODEL;

when SET_BACK_INTERIOR_STYLE =>
  BACK_STYLE_OF_INTERIOR : INTERIOR_STYLE;

when SET_BACK_INTERIOR_STYLE_INDEX =>
  BACK_STYLE_IND : STYLE_INDEX;

when SET_BACK_INTERIOR_COLOUR =>
  BACK_GENERAL_INTERIOR_COLOUR : GENERAL_COLOUR;

when SET_BACK_INTERIOR_SHADING_METHOD =>
  BACK_INTERIOR_SHADING : INTERIOR_SHADING_METHOD;

when SET_BACK_DATA_MAPPING_METHOD =>
  BACK_METHOD_OF_DATA_MAPPING : DATA_MAPPING_METHOD;

when SET_BACK_REFLECTANCE_PROPERTIES =>
  BACK_REFLECTANCE_PROPERTIES
                    : REFLECTANCE_PROPERTIES_DATA_RECORD;

when SET_BACK_REFLECTANCE_MODEL =>
  BACK_REFLECTANCE : REFLECTANCE_MODEL;

when SET_LIGHT_SOURCE_STATE =>
  ACTIVATION_LIST : LIGHT_SOURCE_INDICES.LIST_OF;
  DEACTIVATION_LIST : LIGHT_SOURCE_INDICES.LIST_OF;

when SET_EDGE_COLOUR =>
  GENERAL_EDGE_COLOUR : GENERAL_COLOUR;

when SET_CURVE_APPROX_CRITERIA =>
  CRITERIA_FOR_CURVE_APPROX : CURVE_APPROX_DATA_RECORD;

when SET_SURFACE_APPROX_CRITERIA =>
  CRITERIA_FOR_SURFACE_APPROX
                    : SURFACE_APPROX_DATA_RECORD;
```

```
            when SET_PARAMETRIC_SURFACE_CHARACTERISTICS =>
              PARAMETRIC_SURFACE_CHARACTERISTICS
                                        : PARAMETRIC_SURFACE_DATA_RECORD;

            when SET_RENDERING_COLOUR_MODEL =>
              RENDERING_COLOUR_MODEL : COLOUR_MODEL;

            when SET_DEPTH_CUE_INDEX =>
              DEPTH_CUE_IND : DEPTH_CUE_INDEX;

            when SET_COLOUR_MAPPING_INDEX =>
              COLOUR_MAPPING_IND : COLOUR_MAPPING_INDEX;

          end case;
        end record;

    -- This type defines the format of structure contents for PHIGS PLUS structures and is used to
    -- return structure elements during inquiry.

    PHIGS_ERROR : exception;

    -- Exception to notify the application program of an error in a
    -- PHIGS procedure should the version of ERROR_HANDLING which raises
    -- an exception be used.

    -- This section contains the constants that provide the PHIGS standard
    -- values defined for some PHIGS/Ada types.  The constants for colour
    -- models are defined earlier.

    -- The following constants define the PHIGS standard line types:

SOLID_LINE                  : constant LINETYPE := 1;
DASHED_LINE                 : constant LINETYPE := 2;
DOTTED_LINE                 : constant LINETYPE := 3;
DASHED_DOTTED_LINE          : constant LINETYPE := 4;

-- The following constants define the PHIGS standard marker types:

DOT_MARKER                  : constant MARKER_TYPE := 1;
PLUS_MARKER                 : constant MARKER_TYPE := 2;
STAR_MARKER                 : constant MARKER_TYPE := 3;
ZERO_MARKER                 : constant MARKER_TYPE := 4;
X_MARKER                    : constant MARKER_TYPE := 5;

-- The following constants define the PHIGS standard edgetypes:
```

```
   SOLID_EDGE                    : constant EDGETYPE := 1;
   DASHED_EDGE                   : constant EDGETYPE := 2;
   DOTTED_EDGE                   : constant EDGETYPE := 3;
   DASHED_DOTTED_EDGE            : constant EDGETYPE := 4;

-- The following constants define the PHIGS standard annotation styles:

   UNCONNECTED_ANNOTATION        : constant ANNOTATION_STYLE := 1;
   LEAD_LINE_ANNOTATION          : constant ANNOTATION_STYLE := 2;

-- The following constants are used for defining the modelling clipping
-- operators specified by PHIGS:

   REPLACE_VOLUME                : constant MODELLING_CLIP_OPERATION_TYPE := 1;
   INTERSECT_VOLUME              : constant MODELLING_CLIP_OPERATION_TYPE := 2;

-- The following constants define the prompt and echo types supported
-- by PHIGS:

   DEFAULT_LOCATOR               : constant LOCATOR_PROMPT_ECHO_TYPE := 1;
   CROSS_HAIR_LOCATOR            : constant LOCATOR_PROMPT_ECHO_TYPE := 2;
   TRACKING_CROSS_LOCATOR        : constant LOCATOR_PROMPT_ECHO_TYPE := 3;
   RUBBER_BAND_LINE_LOCATOR      : constant LOCATOR_PROMPT_ECHO_TYPE := 4;
   RECTANGLE_LOCATOR             : constant LOCATOR_PROMPT_ECHO_TYPE := 5;
   DIGITAL_LOCATOR               : constant LOCATOR_PROMPT_ECHO_TYPE := 6;

   DEFAULT_STROKE                : constant STROKE_PROMPT_ECHO_TYPE := 1;
   DIGITAL_STROKE                : constant STROKE_PROMPT_ECHO_TYPE := 2;
   MARKER_STROKE                 : constant STROKE_PROMPT_ECHO_TYPE := 3;
   LINE_STROKE                   : constant STROKE_PROMPT_ECHO_TYPE := 4;

   DEFAULT_VALUATOR              : constant VALUATOR_PROMPT_ECHO_TYPE := 1;
   GRAPHICAL_VALUATOR            : constant VALUATOR_PROMPT_ECHO_TYPE := 2;
   DIGITAL_VALUATOR              : constant VALUATOR_PROMPT_ECHO_TYPE := 3;

   DEFAULT_CHOICE                : constant CHOICE_PROMPT_ECHO_TYPE := 1;
   PROMPT_ECHO_CHOICE            : constant CHOICE_PROMPT_ECHO_TYPE := 2;
   STRING_PROMPT_CHOICE          : constant CHOICE_PROMPT_ECHO_TYPE := 3;
   STRING_INPUT_CHOICE           : constant CHOICE_PROMPT_ECHO_TYPE := 4;
   STRUCTURE_CHOICE              : constant CHOICE_PROMPT_ECHO_TYPE := 5;

   DEFAULT_PICK                  : constant PICK_PROMPT_ECHO_TYPE := 1;
   GROUP_HIGHLIGHT_PICK          : constant PICK_PROMPT_ECHO_TYPE := 2;
   STRUCTURE_HIGHLIGHT_PICK      : constant PICK_PROMPT_ECHO_TYPE := 3;

   end PHIGS_TYPES;
```

```
-- The PHIGS Package

with PHIGS_CONFIGURATION, PHIGS_TYPES, PHIGS_NAME_SET_FACILITY;
use PHIGS_CONFIGURATION, PHIGS_TYPES;

package PHIGS is

-- The package PHIGS contains all of the procedures that are required
-- to implement PHIGS.

   -- Private Type Declarations

   -- The following data types are the PHIGS private types and are
   -- included in package PHIGS for ease of manipulation.

   -- METAFILE_DATA_RECORD

   type METAFILE_DATA_RECORD (TYPE_OF_ITEM : METAFILE_ITEM_TYPE := 0) is
     private;

   --  A data record for metafiles.

   -- CHOICE_DATA_RECORD

   type CHOICE_DATA_RECORD
     (PROMPT_ECHO_TYPE : CHOICE_PROMPT_ECHO_TYPE := 1) is private;

   -- Defines a choice input data record.

   -- LOCATOR_DATA_RECORD

   type LOCATOR_DATA_RECORD
     (PROMPT_ECHO_TYPE : LOCATOR_PROMPT_ECHO_TYPE := 1) is private;

   -- Defines a locator input data record.

   -- PICK_DATA_RECORD

   type PICK_DATA_RECORD
     (PROMPT_ECHO_TYPE : PICK_PROMPT_ECHO_TYPE := 1) is private;

   -- Defines a pick input data record.

   -- STRING_DATA_RECORD

   type STRING_DATA_RECORD
     (PROMPT_ECHO_TYPE : STRING_PROMPT_ECHO_TYPE := 1) is private;
```

```
        -- Defines a string input data record.

        -- STROKE_DATA_RECORD

        type STROKE_DATA_RECORD
          (PROMPT_ECHO_TYPE : STROKE_PROMPT_ECHO_TYPE := 1) is private;

        -- Defines a stroke input data record.

        -- VALUATOR_DATA_RECORD

        type VALUATOR_DATA_RECORD
          (PROMPT_ECHO_TYPE : VALUATOR_PROMPT_ECHO_TYPE := 1) is private;

        -- Defines a valuator data record.

    -- PHIGS Procedures

    -- CONTROL FUNCTIONS


        procedure OPEN_PHIGS
          (ERROR_FILE           : in FILE_ID := PHIGS_STRING(DEFAULT_ERROR_FILE);
           AMOUNT_OF_MEMORY : in PHIGS_NATURAL := DEFAULT_MEMORY_UNITS);

        procedure CLOSE_PHIGS;

        procedure OPEN_WS
          (WS          : in WS_ID;
           CONNECTION  : in CONNECTION_ID;
           TYPE_OF_WS  : in WS_TYPE);

        procedure CLOSE_WS
          (WS : in WS_ID);

        procedure REDRAW_ALL_STRUCTURES
          (WS    : in WS_ID;
           FLAG  : in CONTROL_FLAG);

        procedure UPDATE_WS
          (WS                 : in WS_ID;
           REGENERATION   : in UPDATE_REGENERATION_FLAG);

        procedure SET_DISPLAY_UPDATE_STATE
          (WS                 : in WS_ID;
           DEFERRAL       : in DEFERRAL_MODE;
           MODIFICATION   : in MODIFICATION_MODE);

        procedure MESSAGE
```

```
      (WS              : in WS_ID;
       CONTENTS        : in PHIGS_STRING);

  -- OUTPUT FUNCTIONS

    procedure POLYLINE
      (POINTS : in MC.POINT_LIST_3);

    procedure POLYLINE
      (POINTS : in MC.POINT_LIST_2);

    procedure POLYMARKER
      (POINTS : in MC.POINT_LIST_3);

    procedure POLYMARKER
      (POINTS : in MC.POINT_LIST_2);

    procedure TEXT
      (POSITION                : in MC.POINT_3;
       DIRECTION_VECTORS       : in MC.VECTOR_PAIR_3;
       CHAR_STRING             : in PHIGS_STRING);

    procedure TEXT
      (POSITION        : in MC.POINT_2;
       CHAR_STRING     : in PHIGS_STRING);

    procedure ANNOTATION_TEXT_RELATIVE
      (REFERENCE_POINT         : in MC.POINT_3;
       ANNOTATION_OFFSET       : in NPC.POINT_3;
       CHAR_STRING             : in PHIGS_STRING);

    procedure ANNOTATION_TEXT_RELATIVE
      (REFERENCE_POINT         : in MC.POINT_2;
       ANNOTATION_OFFSET       : in NPC.POINT_2;
       CHAR_STRING             : in PHIGS_STRING);

    procedure FILL_AREA
      (POINTS : in MC.POINT_LIST_3);

    procedure FILL_AREA
      (POINTS : in MC.POINT_LIST_2);

    procedure FILL_AREA_SET
      (POINT_LISTS : in MC.LIST_OF_POINT_LIST_3);

    procedure FILL_AREA_SET
      (POINT_LISTS : in MC.LIST_OF_POINT_LIST_2);

    procedure CELL_ARRAY
```

```
  (CORNER_P   : in MC.POINT_3;
   CORNER_Q   : in MC.POINT_3;
   CORNER_R   : in MC.POINT_3;
   CELLS      : in COLOUR_MATRIX);

procedure CELL_ARRAY
  (CORNER_P   : in MC.POINT_2;
   CORNER_Q   : in MC.POINT_2;
   CELLS      : in COLOUR_MATRIX);


procedure POLYLINE_SET
  (VERTICES : in MC.LIST_OF_VERTEX_COLOUR_LIST_SET_3);

procedure FILL_AREA_SET_WITH_DATA
  (FACET      : in MC.FACET_DATA_SET;
   VERTICES   : in MC.LIST_OF_VERTEX_DATA_LIST_SET_3);

procedure FILL_AREA_SET_WITH_DATA
  (FACET      : in MC.FACET_DATA_SET;
   EDGES      : in LIST_OF_EDGE_FLAG_LIST;
   VERTICES   : in MC.LIST_OF_VERTEX_DATA_LIST_SET_3);

procedure FILL_AREA_SET_WITH_DATA
  (FACET      : in MC.FACET_DATA_SET;
   VERTICES   : in MC.LIST_OF_VERTEX_DATA_LIST_SET_2);

procedure FILL_AREA_SET_WITH_DATA
  (FACET      : in MC.FACET_DATA_SET;
   EDGES      : in LIST_OF_EDGE_FLAG_LIST;
   VERTICES   : in MC.LIST_OF_VERTEX_DATA_LIST_SET_2);

procedure CELL_ARRAY
  (CORNER_P   : in MC.POINT_3;
   CORNER_Q   : in MC.POINT_3;
   CORNER_R   : in MC.POINT_3;
```

```
    CELLS       : in COLOUR_VALUE_ARRAY);

procedure SET_OF_FILL_AREA_SET_WITH_DATA
  (FACETS       : in MC.FACET_DATA_LIST_SET;
   VERTICES     : in MC.VERTEX_DATA_LIST_SET_3;
   INDICES      : in LIST_OF_LIST_OF_VERTEX_INDEX_LIST);

procedure SET_OF_FILL_AREA_SET_WITH_DATA
  (FACETS       : in MC.FACET_DATA_LIST_SET;
   EDGES        : in LIST_OF_LIST_OF_EDGE_FLAG_LIST;
   VERTICES     : in MC.VERTEX_DATA_LIST_SET_3;
   INDICES      : in LIST_OF_LIST_OF_VERTEX_INDEX_LIST);

procedure SET_OF_FILL_AREA_SET_WITH_DATA
  (FACETS       : in MC.FACET_DATA_LIST_SET;
   VERTICES     : in MC.VERTEX_DATA_LIST_SET_2;
   INDICES      : in LIST_OF_LIST_OF_VERTEX_INDEX_LIST);

procedure SET_OF_FILL_AREA_SET_WITH_DATA
  (FACETS       : in MC.FACET_DATA_LIST_SET;
   EDGES        : in LIST_OF_LIST_OF_EDGE_FLAG_LIST;
   VERTICES     : in MC.VERTEX_DATA_LIST_SET_2;
   INDICES      : in LIST_OF_LIST_OF_VERTEX_INDEX_LIST);

procedure TRIANGLE_SET_WITH_DATA
  (FACETS       : in MC.FACET_DATA_LIST_SET;
   VERTICES     : in MC.VERTEX_DATA_LIST_SET_3;
   INDICES      : in LIST_OF_VERTEX_INDEX_TRIPLET);

procedure TRIANGLE_SET_WITH_DATA
  (FACETS       : in MC.FACET_DATA_LIST_SET;
   EDGES        : in LIST_OF_EDGE_FLAG_TRIPLET;
   VERTICES     : in MC.VERTEX_DATA_LIST_SET_3;
   INDICES      : in LIST_OF_VERTEX_INDEX_TRIPLET);

procedure TRIANGLE_SET_WITH_DATA
  (FACETS       : in MC.FACET_DATA_LIST_SET;
   VERTICES     : in MC.VERTEX_DATA_LIST_SET_2;
   INDICES      : in LIST_OF_VERTEX_INDEX_TRIPLET);

procedure TRIANGLE_SET_WITH_DATA
  (FACETS       : in MC.FACET_DATA_LIST_SET;
   EDGES        : in LIST_OF_EDGE_FLAG_TRIPLET;
   VERTICES     : in MC.VERTEX_DATA_LIST_SET_2;
   INDICES      : in LIST_OF_VERTEX_INDEX_TRIPLET);

procedure TRIANGLE_STRIP_WITH_DATA
  (FACETS       : in MC.FACET_DATA_LIST_SET;
   VERTICES     : in MC.VERTEX_DATA_LIST_SET_3);
```

```
procedure TRIANGLE_STRIP_WITH_DATA
  (FACETS      : in MC.FACET_DATA_LIST_SET;
   EDGES       : in EDGE_FLAG_LIST;
   VERTICES    : in MC.VERTEX_DATA_LIST_SET_3);

procedure TRIANGLE_STRIP_WITH_DATA
  (FACETS      : in MC.FACET_DATA_LIST_SET;
   VERTICES    : in MC.VERTEX_DATA_LIST_SET_2);

procedure TRIANGLE_STRIP_WITH_DATA
  (FACETS      : in MC.FACET_DATA_LIST_SET;
   EDGES       : in EDGE_FLAG_LIST;
   VERTICES    : in MC.VERTEX_DATA_LIST_SET_2);

procedure QUADRILATERAL_MESH_WITH_DATA
  (FACETS      : in MC.FACET_DATA_ARRAY_SET;
   VERTICES    : in MC.VERTEX_DATA_ARRAY_SET_3);

procedure QUADRILATERAL_MESH_WITH_DATA
  (FACETS      : in MC.FACET_DATA_ARRAY_SET;
   EDGES       : in ARRAY_OF_EDGE_FLAG_PAIR;
   VERTICES    : in MC.VERTEX_DATA_ARRAY_SET_3);

procedure QUADRILATERAL_MESH_WITH_DATA
  (FACETS      : in MC.FACET_DATA_ARRAY_SET;
   VERTICES    : in MC.VERTEX_DATA_ARRAY_SET_2);

procedure QUADRILATERAL_MESH_WITH_DATA
  (FACETS      : in MC.FACET_DATA_ARRAY_SET;
   EDGES       : in ARRAY_OF_EDGE_FLAG_PAIR;
   VERTICES    : in MC.VERTEX_DATA_ARRAY_SET_2);

procedure NON_UNIFORM_B_SPLINE_CURVE
  (SPLINE    : in MC.CURVE_GEOMETRYSPLINE_3;
   LIMITS    : in SPC.RANGE_OF_MAGNITUDES);

procedure NON_UNIFORM_B_SPLINE_CURVE
  (CURVE_SPLINE  : in MC.CURVE_GEOMETRYSPLINE_3;
   LIMITS        : in SPC.RANGE_OF_MAGNITUDES;
   COLOUR        : in CURVE_COLOURSPLINE);

procedure NON_UNIFORM_B_SPLINE_SURFACE
  (SURFACE_SPLINE : in MC.SURFACE_GEOMETRYSPLINE;
   TRIM_CURVES    : in LIST_OF_TRIMCURVE_LIST);

procedure NON_UNIFORM_B_SPLINE_SURFACE_WITH_DATA
  (SURFACE_SPLINE : in MC.SURFACE_GEOMETRYSPLINE;
   TRIM_CURVES    : in LIST_OF_TRIMCURVE_LIST;
   COLOUR_SPLINE  : in SURFACE_COLOURSPLINE;
   DATA_SPLINES   : in DATASPLINE_LIST);
```

```
-- OUTPUT ATTRIBUTE FUNCTIONS

    procedure SET_POLYLINE_INDEX
      (POLYLINE_IND : in POLYLINE_INDEX);

    procedure SET_POLYMARKER_INDEX
      (POLYMARKER_IND : in POLYMARKER_INDEX);

    procedure SET_TEXT_INDEX
      (TEXT_IND : in TEXT_INDEX);

    procedure SET_INTERIOR_INDEX
      (INTERIOR_IND : in INTERIOR_INDEX);

    procedure SET_EDGE_INDEX
      (EDGE_IND : in EDGE_INDEX);

    procedure SET_DATA_MAPPING_INDEX
      (DATA_MAPPING_IND : in DATA_MAPPING_INDEX);

    procedure SET_REFLECTANCE_INDEX
      (REFLECTANCE_IND : in REFLECTANCE_INDEX);

    procedure SET_BACK_INTERIOR_INDEX
      (BACK_INTERIOR_IND : in INTERIOR_INDEX);

    procedure SET_BACK_DATA_MAPPING_INDEX
      (BACK_DATA_MAPPING_IND : in DATA_MAPPING_INDEX);

    procedure SET_BACK_REFLECTANCE_INDEX
      (BACK_REFLECTANCE_IND : in REFLECTANCE_INDEX);

    procedure SET_PARAMETRIC_SURFACE_INDEX
      (PARAMETRIC_SURFACE_IND : in PARAMETRIC_SURFACE_INDEX);

    procedure SET_LINETYPE
      (TYPE_OF_LINE : in LINETYPE);

    procedure SET_LINEWIDTH_SCALE_FACTOR
      (LINEWIDTH_SF : in LINEWIDTH);

    procedure SET_POLYLINE_COLOUR_INDEX
```

```
    (LINE_COLOUR : in COLOUR_INDEX);

  procedure SET_POLYLINE_COLOUR
    (GENERAL_POLYLINE_COLOUR : in GENERAL_COLOUR);

  procedure SET_POLYLINE_SHADING_METHOD
    (POLYLINE_SHADING : in POLYLINE_SHADING_METHOD);

  procedure SET_MARKER_TYPE
    (TYPE_OF_MARKER : in MARKER_TYPE);

  procedure SET_MARKER_SIZE_SCALE_FACTOR
    (SIZE : in MARKER_SIZE);

  procedure SET_POLYMARKER_COLOUR_INDEX
    (MARKER_COLOUR : in COLOUR_INDEX);

  procedure SET_POLYMARKER_COLOUR
    (GENERAL_POLYMARKER_COLOUR : in GENERAL_COLOUR);

  procedure SET_TEXT_FONT
    (FONT : in TEXT_FONT);

  procedure SET_TEXT_PRECISION
    (PRECISION : in TEXT_PRECISION);

  procedure SET_CHAR_EXPANSION_FACTOR
    (EXPANSION : in CHAR_EXPANSION);

  procedure SET_CHAR_SPACING
    (SPACING : in CHAR_SPACING);

  procedure SET_TEXT_COLOUR_INDEX
    (TEXT_COLOUR : in COLOUR_INDEX);

  procedure SET_TEXT_COLOUR
    (GENERAL_TEXT_COLOUR : in GENERAL_COLOUR);

  procedure SET_CHAR_HEIGHT
    (HEIGHT : in MC.MAGNITUDE);

  procedure SET_CHAR_UP_VECTOR
    (CHAR_UP_VECTOR : in MC.VECTOR_2);

  procedure SET_TEXT_PATH
    (PATH : in TEXT_PATH);

  procedure SET_TEXT_ALIGNMENT
```

```
    (ALIGNMENT : in TEXT_ALIGNMENT);

  procedure SET_ANNOTATION_TEXT_CHAR_HEIGHT
    (ANNOTATION_HEIGHT : in NPC.MAGNITUDE);

  procedure SET_ANNOTATION_TEXT_CHAR_UP_VECTOR
    (ANNOTATION_CHAR_UP_VECTOR : in NPC.VECTOR_2);

  procedure SET_ANNOTATION_TEXT_PATH
    (ANNOTATION_PATH : in TEXT_PATH);

  procedure SET_ANNOTATION_TEXT_ALIGNMENT
    (ANNOTATION_ALIGNMENT : in TEXT_ALIGNMENT);

  procedure SET_ANNOTATION_STYLE
    (STYLE_OF_ANNOTATION : in ANNOTATION_STYLE);

  procedure SET_INTERIOR_STYLE
    (STYLE_OF_INTERIOR : in INTERIOR_STYLE);

  procedure SET_INTERIOR_STYLE_INDEX
    (STYLE_IND : in STYLE_INDEX);

  procedure SET_FACET_DISTINGUISHING_MODE
    (FACET_DISTINGUISHING : in FACET_DISTINGUISHING_MODE);

  procedure SET_FACET_CULLING_MODE
    (FACET_CULLING : in FACET_CULLING_MODE);

  procedure SET_INTERIOR_COLOUR_INDEX
```

```
    (INTERIOR_COLOUR : in COLOUR_INDEX);

  procedure SET_INTERIOR_COLOUR
    (GENERAL_INTERIOR_COLOUR : in GENERAL_COLOUR);

  procedure SET_INTERIOR_SHADING_METHOD
    (INTERIOR_SHADING : in INTERIOR_SHADING_METHOD);

  procedure SET_DATA_MAPPING_METHOD
    (METHOD_OF_DATA_MAPPING : in DATA_MAPPING_DATA_RECORD);

  procedure SET_REFLECTANCE_PROPERTIES
     (REFLECTANCE_DATA : in REFLECTANCE_PROPERTIES_DATA_RECORD);

  procedure SET_REFLECTANCE_MODEL
    (REFLECTANCE : in REFLECTANCE_MODEL);

  procedure SET_BACK_INTERIOR_STYLE
    (BACK_STYLE_OF_INTERIOR : in INTERIOR_STYLE);

  procedure SET_BACK_INTERIOR_STYLE_INDEX
    (BACK_STYLE_IND : in STYLE_INDEX);

  procedure SET_BACK_INTERIOR_COLOUR
    (BACK_GENERAL_INTERIOR_COLOUR : in GENERAL_COLOUR);

  procedure SET_BACK_INTERIOR_SHADING_METHOD
    (BACK_INTERIOR_SHADING : in INTERIOR_SHADING_METHOD);

  procedure SET_BACK_DATA_MAPPING_METHOD
    (BACK_METHOD_OF_DATA_MAPPING : in DATA_MAPPING_DATA_RECORD);

  procedure SET_BACK_REFLECTANCE_PROPERTIES
     (BACK_REFLECTANCE_DATA : in REFLECTANCE_PROPERTIES_DATA_RECORD);

  procedure SET_BACK_REFLECTANCE_MODEL
    (BACK_REFLECTANCE : in REFLECTANCE_MODEL);

  procedure SET_LIGHT_SOURCE_STATE
    (ACTIVATION_LIST      : in LIGHT_SOURCE_INDICES.LIST_OF;
     DEACTIVATION_LIST  : in LIGHT_SOURCE_INDICES.LIST_OF);

  procedure SET_EDGE_FLAG
    (FLAG : in EDGE_FLAG);

  procedure SET_EDGETYPE
    (TYPE_OF_EDGE : in EDGETYPE);

  procedure SET_EDGEWIDTH_SCALE_FACTOR
```

```
    (EDGEWIDTH_SF : in EDGEWIDTH);

  procedure SET_EDGE_COLOUR_INDEX
    (EDGE_COLOUR : in COLOUR_INDEX);

  procedure SET_EDGE_COLOUR
    (GENERAL_EDGE_COLOUR : GENERAL_COLOUR);

  procedure SET_PATTERN_SIZE
    (SIZE : in MC.SIZE_2);

  procedure SET_PATTERN_REFERENCE_POINT_AND_VECTORS
    (REFERENCE_POINT      : in MC.POINT_3;
     REFERENCE_VECTORS    : in MC.VECTOR_PAIR_3);

  procedure SET_PATTERN_REFERENCE_POINT
    (REFERENCE_POINT : MC.POINT_2);

  procedure SET_CURVE_APPROXIMATION_CRITERIA
    (CRITERIA_FOR_CURVE_APPROX : in CURVE_APPROX_DATA_RECORD);

  procedure SET_SURFACE_APPROXIMATION_CRITERIA
    (CRITERIA_FOR_SURFACE_APPROX : in SURFACE_APPROX_DATA_RECORD);

  procedure SET_PARAMETRIC_SURFACE_CHARACTERISTICS
    (PARAMETRIC_SURFACE_CHARACTERISTICS
                                    : in PARAMETRIC_SURFACE_DATA_RECORD);

  procedure SET_RENDERING_COLOUR_MODEL
    (RENDERING_COLOUR_MODEL : in COLOUR_MODEL);

  procedure SET_DEPTH_CUE_INDEX
    (DEPTH_CUE_IND : in DEPTH_CUE_INDEX);

  procedure SET_COLOUR_MAPPING_INDEX
    (COLOUR_MAPPING_IND : in COLOUR_MAPPING_INDEX);

  procedure ADD_NAMES_TO_SET
    (NAMES_TO_ADD : in NAME_SET);

  procedure REMOVE_NAMES_FROM_SET
    (NAMES_TO_REMOVE : in NAME_SET);

  procedure SET_INDIVIDUAL_ASF
    (ASPECT_ID        : in ASPECT;
     SOURCE_FLAG      : in ASF);

  procedure SET_POLYLINE_REPRESENTATION
    (WS               : in WS_ID;
     POLYLINE_IND     : in POLYLINE_INDEX;
```

```
    TYPE_OF_LINE    : in LINETYPE;
    LINEWIDTH_SF    : in LINEWIDTH;
    LINE_COLOUR     : in COLOUR_INDEX);

procedure SET_POLYLINE_REPRESENTATION
  (WS                              : in WS_ID;
   POLYLINE_IND                    : in POLYLINE_INDEX;
   TYPE_OF_LINE                    : in LINETYPE;
   LINEWIDTH_SF                    : in LINEWIDTH;
   GENERAL_LINE_COLOUR             : in GENERAL_COLOUR;
   LINE_SHADING                    : in POLYLINE_SHADING_METHOD;
   CRITERIA_FOR_CURVE_APPROX       : in CURVE_APPROX_DATA_RECORD);

procedure SET_POLYMARKER_REPRESENTATION
  (WS                  : in WS_ID;
   POLYMARKER_IND      : in POLYMARKER_INDEX;
   TYPE_OF_MARKER      : in MARKER_TYPE;
   SIZE                : in MARKER_SIZE;
   MARKER_COLOUR       : in COLOUR_INDEX);

procedure SET_POLYMARKER_REPRESENTATION
  (WS                              : in WS_ID;
   POLYMARKER_IND                  : in POLYMARKER_INDEX;
   TYPE_OF_MARKER                  : in MARKER_TYPE;
   SIZE                            : in MARKER_SIZE;
   GENERAL_MARKER_COLOUR           : in GENERAL_COLOUR);

procedure SET_TEXT_REPRESENTATION
  (WS                : in WS_ID;
   TEXT_IND          : in TEXT_INDEX;
   FONT              : in TEXT_FONT;
   PRECISION         : in TEXT_PRECISION;
   EXPANSION         : in CHAR_EXPANSION;
   SPACING           : in CHAR_SPACING;
   TEXT_COLOUR       : in COLOUR_INDEX);

procedure SET_TEXT_REPRESENTATION
  (WS                              : in WS_ID;
   TEXT_IND                        : in TEXT_INDEX;
   FONT                            : in TEXT_FONT;
   PRECISION                       : in TEXT_PRECISION;
   EXPANSION                       : in CHAR_EXPANSION;
   SPACING                         : in CHAR_SPACING;
   GENERAL_TEXT_COLOUR             : in GENERAL_COLOUR);

procedure SET_INTERIOR_REPRESENTATION
  (WS                    : in WS_ID;
   INTERIOR_IND          : in INTERIOR_INDEX;
   STYLE_OF_INTERIOR     : in INTERIOR_STYLE;
```

```
    STYLE_IND              : in STYLE_INDEX;
    INTERIOR_COLOUR        : in COLOUR_INDEX);

procedure SET_INTERIOR_REPRESENTATION
  (WS                              : in WS_ID;
   INTERIOR_IND                    : in INTERIOR_INDEX;
   STYLE_OF_INTERIOR               : in INTERIOR_STYLE;
   STYLE_IND                       : in STYLE_INDEX;
   GENERAL_INTERIOR_COLOUR         : in GENERAL_COLOUR;
   INTERIOR_SHADING                : in INTERIOR_SHADING_METHOD);

procedure SET_EDGE_REPRESENTATION
  (WS                : in WS_ID;
   EDGE_IND          : in EDGE_INDEX;
   FLAG              : in EDGE_FLAG;
   TYPE_OF_EDGE      : in EDGETYPE;
   EDGEWIDTH_SF      : in EDGEWIDTH;
   EDGE_COLOUR       : in COLOUR_INDEX);

procedure SET_EDGE_REPRESENTATION
  (WS                              : in WS_ID;
   EDGE_IND                        : in EDGE_INDEX;
   FLAG                            : in EDGE_FLAG;
   TYPE_OF_EDGE                    : in EDGETYPE;
   EDGEWIDTH_SF                    : in EDGEWIDTH;
   GENERAL_EDGE_COLOUR             : in GENERAL_COLOUR);

procedure SET_DATA_MAPPING_REPRESENTATION
  (WS                : in WS_ID;
   DATA_MAPPING_IND  : in DATA_MAPPING_INDEX;
   DATA_MAPPING      : in DATA_MAPPING_DATA_RECORD);

procedure SET_REFLECTANCE_REPRESENTATION
  (WS                              : in WS_ID;
   REFLECTANCE_IND                 : in REFLECTANCE_INDEX;
   MODEL_OF_REFLECTANCE            : in REFLECTANCE_MODEL;
   REFLECTANCE_PROPERTIES          : in REFLECTANCE_PROPERTIES_DATA_RECORD);

procedure SET_PARAMETRIC_SURFACE_REPRESENTATION
  (WS                                    : in WS_ID;
   PARAMETRIC_SURFACE_IND                : in PARAMETRIC_SURFACE_INDEX;
   CRITERIA_FOR_SURFACE_APPROX           : in SURFACE_APPROX_DATA_RECORD;
   PARAMETRIC_SURFACE_CHARACTERISTICS
                                         : in PARAMETRIC_SURFACE_DATA_RECORD);

procedure SET_PATTERN_REPRESENTATION
  (WS                : in WS_ID;
   PATTERN_IND       : in PATTERN_INDEX;
   PATTERN           : in COLOUR_MATRIX);
```

```
    procedure SET_PATTERN_REPRESENTATION
      (WS                  : in WS_ID;
       PATTERN_IND         : in PATTERN_INDEX;
       GENERAL_PATTERN     : in COLOUR_VALUE_ARRAY);

    procedure SET_LIGHT_SOURCE_REPRESENTATION
      (WS                  : in WS_ID;
       LIGHT_SOURCE_IND    : in LIGHT_SOURCE_INDEX;
       LIGHT_SOURCE        : in LIGHT_SOURCE_DATA_RECORD);

    procedure SET_DEPTH_CUE_REPRESENTATION
      (WS                  : in WS_ID;
       DEPTH_CUE_IND       : in DEPTH_CUE_INDEX;
       MODE                : in DEPTH_CUE_MODE;
       REFERENCE_PLANES    : in NPC.RANGE_OF_MAGNITUDES;
       DEPTH_CUE_SF        : in DEPTH_CUE_SCALE_FACTORS;
       DEPTH_CUE_COLOUR    : in GENERAL_COLOUR);

    procedure SET_COLOUR_MAPPING_REPRESENTATION
      (WS                  : in WS_ID;
       COLOUR_MAPPING_IND  : in COLOUR_MAPPING_INDEX;
       COLOUR_MAPPING      : in COLOUR_MAPPING_DATA_RECORD);

    procedure SET_COLOUR_REPRESENTATION
      (WS                  : in WS_ID;
       COLOUR_IND          : in COLOUR_INDEX;
       COLOUR              : in COLOUR_REPRESENTATION);

    procedure SET_HIGHLIGHTING_FILTER
      (WS                  : in WS_ID;
       HIGHLIGHTING        : in NAME_SET_FILTER);


    procedure SET_INVISIBILITY_FILTER
      (WS                  : in WS_ID;
       INVISIBILITY        : in NAME_SET_FILTER);

    procedure SET_COLOUR_MODEL
      (WS                  : in WS_ID;
       MODEL               : in COLOUR_MODEL);

    procedure SET_HLHSR_IDENTIFIER
      (HLHSR_IDENTIFIER : in HLHSR_ID);

    procedure SET_HLHSR_MODE
      (WS                  : in WS_ID;
       MODE                : in HLHSR_MODE);

  -- TRANSFORMATION FUNCTIONS
```

```
proccdure SET_LOCAL_TRANSFORMATION
  (MATRIX          : in TRANSFORMATION_MATRIX_3;
   HOW_APPLIED     : in COMPOSITION_TYPE);

procedure SET_LOCAL_TRANSFORMATION
  (MATRIX          : in TRANSFORMATION_MATRIX_2;
   HOW_APPLIED     : in COMPOSITION_TYPE);

procedure SET_GLOBAL_TRANSFORMATION
  (MATRIX : in TRANSFORMATION_MATRIX_3);

procedure SET_GLOBAL_TRANSFORMATION
  (MATRIX : in TRANSFORMATION_MATRIX_2);

procedure SET_MODELLING_CLIPPING_VOLUME
  (OPERATOR        : in MODELLING_CLIP_OPERATION_TYPE;
   HALF_SPACES     : in MC.HALF_SPACE_LIST_3);

procedure SET_MODELLING_CLIPPING_VOLUME
  (OPERATOR        : in MODELLING_CLIP_OPERATION_TYPE;
   HALF_SPACES     : in MC.HALF_SPACE_LIST_2);

procedure SET_MODELLING_CLIPPING_INDICATOR
  (MODELLING_CLIPPING : in CLIPPING_INDICATOR);

procedure RESTORE_MODELLING_CLIPPING_VOLUME;

procedure SET_VIEW_INDEX
  (VIEW_IND : in VIEW_INDEX);

procedure SET_VIEW_REPRESENTATION
  (WS                      : in WS_ID;
   VIEW_IND                : in VIEW_INDEX;
   ORIENTATION_MATRIX      : in TRANSFORMATION_MATRIX_3;
   MAPPING_MATRIX          : in TRANSFORMATION_MATRIX_3;
   CLIPPING_LIMITS         : in NPC.RECTANGULAR_REGION_3;
   XY_CLIPPING             : in CLIPPING_INDICATOR;
   BACK_CLIPPING           : in CLIPPING_INDICATOR;
   FRONT_CLIPPING          : in CLIPPING_INDICATOR);

procedure SET_VIEW_REPRESENTATION
  (WS                      : in WS_ID;
   VIEW_IND                : in VIEW_INDEX;
   ORIENTATION_MATRIX      : in TRANSFORMATION_MATRIX_2;
   MAPPING_MATRIX          : in TRANSFORMATION_MATRIX_2;
   CLIPPING_LIMITS         : in NPC.RECTANGULAR_REGION_2;
   XY_CLIPPING             : in CLIPPING_INDICATOR);

procedure SET_VIEW_TRANSFORMATION_INPUT_PRIORITY
```

```
      (WS                    : in WS_ID;
       VIEW_IND              : in VIEW_INDEX;
       REFERENCE_INDEX       : in VIEW_INDEX;
       PRIORITY              : in RELATIVE_PRIORITY);

    procedure SET_WS_WINDOW
      (WS                    : in WS_ID;
       WINDOW_LIMITS         : in NPC.RECTANGULAR_REGION_3);

    procedure SET_WS_WINDOW
      (WS                    : in WS_ID;
       WINDOW_LIMITS         : in NPC.RECTANGULAR_REGION_2);

    procedure SET_WS_VIEWPORT
      (WS                    : in WS_ID;
       VIEWPORT_LIMITS       : in DC.RECTANGULAR_REGION_3);

    procedure SET_WS_VIEWPORT
      (WS                    : in WS_ID;
       VIEWPORT_LIMITS       : in DC.RECTANGULAR_REGION_2);

  -- TRANSFORMATION UTILITY FUNCTIONS

    procedure TRANSLATE
      (VECTOR                : in  MC.VECTOR_3;
       ERROR_INDICATOR       : out ERROR_NUMBER;
       MATRIX                : out TRANSFORMATION_MATRIX_3);

    procedure TRANSLATE
      (VECTOR                : in  MC.VECTOR_2;
       ERROR_INDICATOR       : out ERROR_NUMBER;
       MATRIX                : out TRANSFORMATION_MATRIX_2);

    procedure SCALE
      (FACTOR                : in  MC.VECTOR_3;
       ERROR_INDICATOR       : out ERROR_NUMBER;
       MATRIX                : out TRANSFORMATION_MATRIX_3);

    procedure SCALE
      (FACTOR                : in  MC.VECTOR_2;
       ERROR_INDICATOR       : out ERROR_NUMBER;
       MATRIX                : out TRANSFORMATION_MATRIX_2);

    procedure ROTATE_X
      (ANGLE_X               : in  ANGLE;
       ERROR_INDICATOR       : out ERROR_NUMBER;
       MATRIX                : out TRANSFORMATION_MATRIX_3);

    procedure ROTATE_Y
```

```
    (ANGLE_Y           : in  ANGLE;
     ERROR_INDICATOR   : out ERROR_NUMBER;
     MATRIX            : out TRANSFORMATION_MATRIX_3);

  procedure ROTATE_Z
    (ANGLE_Z           : in  ANGLE;
     ERROR_INDICATOR   : out ERROR_NUMBER;
     MATRIX            : out TRANSFORMATION_MATRIX_3);

  procedure ROTATE
    (ANGLE_Z           : in  ANGLE;
     ERROR_INDICATOR   : out ERROR_NUMBER;
     MATRIX            : out TRANSFORMATION_MATRIX_2);

  procedure COMPOSE_MATRIX
    (MATRIX_A          : in  TRANSFORMATION_MATRIX_3;
     MATRIX_B          : in  TRANSFORMATION_MATRIX_3;
     ERROR_INDICATOR   : out ERROR_NUMBER;
     COMPOSED_MATRIX   : out TRANSFORMATION_MATRIX_3);

  procedure COMPOSE_MATRIX
    (MATRIX_A          : in  TRANSFORMATION_MATRIX_2;
     MATRIX_B          : in  TRANSFORMATION_MATRIX_2;
     ERROR_INDICATOR   : out ERROR_NUMBER;
     COMPOSED_MATRIX   : out TRANSFORMATION_MATRIX_2);

  procedure TRANSFORM_POINT
    (POINT             : in  MC.POINT_3;
     MATRIX            : in  TRANSFORMATION_MATRIX_3;
     ERROR_INDICATOR   : out ERROR_NUMBER;
     TRANSFORMED_POINT : out MC.POINT_3);

  procedure TRANSFORM_POINT
    (POINT             : in  MC.POINT_2;
     MATRIX            : in  TRANSFORMATION_MATRIX_2;
     ERROR_INDICATOR   : out ERROR_NUMBER;
     TRANSFORMED_POINT : out MC.POINT_2);

  procedure BUILD_TRANSFORMATION_MATRIX
    (FIXED_POINT       : in  MC.POINT_3;
     SHIFT_VECTOR      : in  MC.VECTOR_3;
     ANGLE_X           : in  ANGLE;
     ANGLE_Y           : in  ANGLE;
     ANGLE_Z           : in  ANGLE;
     SCALE_FACTORS     : in  MC.VECTOR_3;
     ERROR_INDICATOR   : out ERROR_NUMBER;
     MATRIX            : out TRANSFORMATION_MATRIX_3);

  procedure BUILD_TRANSFORMATION_MATRIX
```

```
  (FIXED_POINT          : in  MC.POINT_2;
   SHIFT_VECTOR         : in  MC.VECTOR_2;
   ANGLE_Z              : in  ANGLE;
   SCALE_FACTORS        : in  MC.VECTOR_2;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   MATRIX               : out TRANSFORMATION_MATRIX_2);

procedure COMPOSE_TRANSFORMATION_MATRIX
  (MATRIX               : in  TRANSFORMATION_MATRIX_3;
   FIXED_POINT          : in  MC.POINT_3;
   SHIFT_VECTOR         : in  MC.VECTOR_3;
   ANGLE_X              : in  ANGLE;
   ANGLE_Y              : in  ANGLE;
   ANGLE_Z              : in  ANGLE;
   SCALE_FACTORS        : in  MC.VECTOR_3;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   COMPOSED_MATRIX      : out TRANSFORMATION_MATRIX_3);

procedure COMPOSE_TRANSFORMATION_MATRIX
  (MATRIX               : in  TRANSFORMATION_MATRIX_2;
   FIXED_POINT          : in  MC.POINT_2;
   SHIFT_VECTOR         : in  MC.VECTOR_2;
   ANGLE_Z              : in  ANGLE;
   SCALE_FACTORS        : in  MC.VECTOR_2;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   COMPOSED_MATRIX      : out TRANSFORMATION_MATRIX_2);

procedure EVALUATE_VIEW_ORIENTATION_MATRIX
  (REFERENCE_POINT      : in  WC.POINT_3;
   NORMAL_VECTOR        : in  WC.VECTOR_3;
   UP_VECTOR            : in  WC.VECTOR_3;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   ORIENTATION_MATRIX   : out TRANSFORMATION_MATRIX_3);

procedure EVALUATE_VIEW_ORIENTATION_MATRIX
  (REFERENCE_POINT      : in  WC.POINT_2;
   UP_VECTOR            : in  WC.VECTOR_2;
   ERROR_INDICATOR      : out ERROR_NUMBER;
   ORIENTATION_MATRIX   : out TRANSFORMATION_MATRIX_2);

procedure EVALUATE_VIEW_MAPPING_MATRIX
  (WINDOW_LIMITS        : in  VRC.RECTANGULAR_REGION_2;
   VIEWPORT_LIMITS      : in  NPC.RECTANGULAR_REGION_3;
   TYPE_OF_PROJECTION   : in  PROJECTION_TYPE;
   REFERENCE_POINT      : in  VRC.POINT_3;
   VIEW_DISTANCE        : in  VRC_TYPE;
   BACK_DISTANCE        : in  VRC_TYPE;
   FRONT_DISTANCE       : in  VRC_TYPE;
   ERROR_INDICATOR      : out ERROR_NUMBER;
```

```
      MAPPING_MATRIX       : out  TRANSFORMATION_MATRIX_3);

    procedure EVALUATE_VIEW_MAPPING_MATRIX
      (WINDOW_LIMITS     : in   VRC.RECTANGULAR_REGION_2;
       VIEWPORT_LIMITS   : in   NPC.RECTANGULAR_REGION_2;
       ERROR_INDICATOR   : out  ERROR_NUMBER;
       MAPPING_MATRIX    : out  TRANSFORMATION_MATRIX_2);

-- STRUCTURE MANIPULATION FUNCTIONS

    procedure OPEN_STRUCTURE
      (STRUCTURE_IDENTIFIER : in STRUCTURE_ID);

    procedure CLOSE_STRUCTURE;

    procedure EXECUTE_STRUCTURE
      (STRUCTURE_IDENTIFIER : in STRUCTURE_ID);

    procedure LABEL
      (LABEL_IDENTIFIER : in LABEL_ID);

    procedure APPLICATION_DATA
      (DATA : in APPLICATION_DATA_RECORD);

    procedure SET_EDIT_MODE
      (MODE : in EDIT_MODE);

    procedure COPY_ALL_ELEMENTS_FROM_STRUCTURE
      (STRUCTURE_IDENTIFIER : in STRUCTURE_ID);

    procedure SET_ELEMENT_POINTER
      (POSITION : in ELEMENT_POSITION);

    procedure OFFSET_ELEMENT_POINTER
      (OFFSET : in ELEMENT_POSITION);

    procedure SET_ELEMENT_POINTER_AT_LABEL
      (LABEL_IDENTIFIER : in LABEL_ID);

    procedure DELETE_ELEMENT;

    procedure DELETE_ELEMENT_RANGE
      (POSITION_1  : in ELEMENT_POSITION;
       POSITION_2  : in ELEMENT_POSITION);

    procedure DELETE_ELEMENTS_BETWEEN_LABELS
      (LABEL_IDENTIFIER_1   : in LABEL_ID;
       LABEL_IDENTIFIER_2   : in LABEL_ID);

    procedure EMPTY_STRUCTURE
```

```
    (STRUCTURE_IDENTIFIER : in STRUCTURE_ID);

  procedure DELETE_STRUCTURE
    (STRUCTURE_IDENTIFIER : in STRUCTURE_ID);

  procedure DELETE_STRUCTURE_NETWORK
    (STRUCTURE_IDENTIFIER   : in STRUCTURE_ID;
     HANDLING_FLAG          : in REFERENCE_HANDLING_FLAG);

  procedure DELETE_ALL_STRUCTURES;

  procedure CHANGE_STRUCTURE_IDENTIFIER
    (ORIGINAL_IDENTIFIER    : in STRUCTURE_ID;
     RESULTING_IDENTIFIER   : in STRUCTURE_ID);

  procedure CHANGE_STRUCTURE_REFERENCES
    (ORIGINAL_IDENTIFIER    : in STRUCTURE_ID;
     RESULTING_IDENTIFIER   : in STRUCTURE_ID);

  procedure CHANGE_STRUCTURE_IDENTIFIER_AND_REFERENCES
    (ORIGINAL_IDENTIFIER    : in STRUCTURE_ID;
     RESULTING_IDENTIFIER   : in STRUCTURE_ID);

  procedure POST_STRUCTURE
    (WS                     : in WS_ID;
     STRUCTURE_IDENTIFIER   : in STRUCTURE_ID;
     PRIORITY               : in DISPLAY_PRIORITY);

  procedure UNPOST_STRUCTURE
    (WS                     : in WS_ID;
     STRUCTURE_IDENTIFIER   : in STRUCTURE_ID);

  procedure UNPOST_ALL_STRUCTURES
    (WS : in WS_ID);

-- STRUCTURE ARCHIVING FUNCTIONS

  procedure OPEN_ARCHIVE_FILE
    (ARCHIVE_IDENTIFIER   : in ARCHIVE_ID;
     ARCHIVE_FILE         : in FILE_ID);

  procedure CLOSE_ARCHIVE_FILE
    (ARCHIVE_IDENTIFIER : in ARCHIVE_ID);

  procedure ARCHIVE_STRUCTURES
    (ARCHIVE_IDENTIFIER     : in ARCHIVE_ID;
     STRUCTURE_IDENTIFIERS  : in STRUCTURE_IDS.LIST_OF);

  procedure ARCHIVE_STRUCTURE_NETWORKS
```

```
  (ARCHIVE_IDENTIFIER        : in ARCHIVE_ID;
   STRUCTURE_IDENTIFIERS     : in STRUCTURE_IDS.LIST_OF);

procedure ARCHIVE_ALL_STRUCTURES
  (ARCHIVE_IDENTIFIER : in ARCHIVE_ID);

procedure SET_CONFLICT_RESOLUTION
  (ARCHIVAL_CONFLICT     : in CONFLICT_RESOLUTION;
   RETRIEVAL_CONFLICT    : in CONFLICT_RESOLUTION);

procedure RETRIEVE_STRUCTURE_IDENTIFIERS
  (ARCHIVE_IDENTIFIER        : in  ARCHIVE_ID;
   STRUCTURE_IDENTIFIERS     : out STRUCTURE_IDS.LIST_OF);

procedure RETRIEVE_STRUCTURES
  (ARCHIVE_IDENTIFIER        : in ARCHIVE_ID;
   STRUCTURE_IDENTIFIERS     : in STRUCTURE_IDS.LIST_OF);

procedure RETRIEVE_STRUCTURE_NETWORKS
  (ARCHIVE_IDENTIFIER        : in ARCHIVE_ID;
   STRUCTURE_IDENTIFIERS     : in STRUCTURE_IDS.LIST_OF);

procedure RETRIEVE_ALL_STRUCTURES
  (ARCHIVE_IDENTIFIER : in ARCHIVE_ID);

procedure RETRIEVE_PATHS_TO_ANCESTORS
  (ARCHIVE_IDENTIFIER        : in  ARCHIVE_ID;
   STRUCTURE_IDENTIFIER      : in  STRUCTURE_ID;
   TRUNCATION                : in  TRUNCATION_METHOD;
   DEPTH                     : in  PATH_DEPTH;
   LIST_OF_PATHS             : out REFERENCE_PATHS.LIST_OF);

procedure RETRIEVE_PATHS_TO_DESCENDANTS
  (ARCHIVE_IDENTIFIER        : in  ARCHIVE_ID;
   STRUCTURE_IDENTIFIER      : in  STRUCTURE_ID;
   TRUNCATION                : in  TRUNCATION_METHOD;
   DEPTH                     : in  PATH_DEPTH;
   LIST_OF_PATHS             : out REFERENCE_PATHS.LIST_OF);

procedure DELETE_STRUCTURES_FROM_ARCHIVE
  (ARCHIVE_IDENTIFIER        : in ARCHIVE_ID;
   STRUCTURE_IDENTIFIERS     : in STRUCTURE_IDS.LIST_OF);

procedure DELETE_STRUCTURE_NETWORKS_FROM_ARCHIVE
  (ARCHIVE_IDENTIFIER        : in ARCHIVE_ID;
   STRUCTURE_IDENTIFIERS     : in STRUCTURE_IDS.LIST_OF);

procedure DELETE_ALL_STRUCTURES_FROM_ARCHIVE
  (ARCHIVE_IDENTIFIER : in ARCHIVE_ID);
```

```
-- INPUT FUNCTIONS

    procedure SET_PICK_IDENTIFIER
      (PICK_IDENTIFIER : in PICK_ID);

    procedure SET_PICK_FILTER
      (WS              : in WS_ID;
       PICK_DEVICE     : in PICK_DEVICE_NUMBER;
       PICKABILITY     : in NAME_SET_FILTER);

    procedure INITIALIZE_LOCATOR
      (WS              : in WS_ID;
       DEVICE          : in LOCATOR_DEVICE_NUMBER;
       INITIAL_VIEW_IND : in VIEW_INDEX;
       INITIAL_POSITION : in WC.POINT_3;
       ECHO_VOLUME     : in DC.RECTANGULAR_REGION_3;
       DATA_RECORD     : in LOCATOR_DATA_RECORD);

    procedure INITIALIZE_LOCATOR
      (WS              : in WS_ID;
       DEVICE          : in LOCATOR_DEVICE_NUMBER;
       INITIAL_VIEW_IND : in VIEW_INDEX;
       INITIAL_POSITION : in WC.POINT_2;
       ECHO_AREA       : in DC.RECTANGULAR_REGION_2;
       DATA_RECORD     : in LOCATOR_DATA_RECORD);

    procedure INITIALIZE_STROKE
      (WS              : in WS_ID;
       DEVICE          : in STROKE_DEVICE_NUMBER;
       INITIAL_VIEW_IND : in VIEW_INDEX;
       INITIAL_STROKE  : in WC.POINT_LIST_3;
       ECHO_VOLUME     : in DC.RECTANGULAR_REGION_3;
       DATA_RECORD     : in STROKE_DATA_RECORD);

    procedure INITIALIZE_STROKE
      (WS              : in WS_ID;
       DEVICE          : in STROKE_DEVICE_NUMBER;
       INITIAL_VIEW_IND : in VIEW_INDEX;
       INITIAL_STROKE  : in WC.POINT_LIST_2;
       ECHO_AREA       : in DC.RECTANGULAR_REGION_2;
       DATA_RECORD     : in STROKE_DATA_RECORD);

    procedure INITIALIZE_VALUATOR
      (WS              : in WS_ID;
       DEVICE          : in VALUATOR_DEVICE_NUMBER;
       INITIAL_VALUE   : in VALUATOR_VALUE;
       ECHO_VOLUME     : in DC.RECTANGULAR_REGION_3;
       DATA_RECORD     : in VALUATOR_DATA_RECORD);
```

```
procedure INITIALIZE_VALUATOR
  (WS             : in WS_ID;
   DEVICE         : in VALUATOR_DEVICE_NUMBER;
   INITIAL_VALUE  : in VALUATOR_VALUE;
   ECHO_AREA      : in DC.RECTANGULAR_REGION_2;
   DATA_RECORD    : in VALUATOR_DATA_RECORD);

procedure INITIALIZE_CHOICE
  (WS             : in WS_ID;
   DEVICE         : in CHOICE_DEVICE_NUMBER;
   INITIAL_STATUS : in CHOICE_STATUS;
   INITIAL_CHOICE : in CHOICE_NUMBER;
   ECHO_VOLUME    : in DC.RECTANGULAR_REGION_3;
   DATA_RECORD    : in CHOICE_DATA_RECORD);

procedure INITIALIZE_CHOICE
  (WS             : in WS_ID;
   DEVICE         : in CHOICE_DEVICE_NUMBER;
   INITIAL_STATUS : in CHOICE_STATUS;
   INITIAL_CHOICE : in CHOICE_NUMBER;
   ECHO_AREA      : in DC.RECTANGULAR_REGION_2;
   DATA_RECORD    : in CHOICE_DATA_RECORD);

procedure INITIALIZE_PICK
  (WS             : in WS_ID;
   DEVICE         : in PICK_DEVICE_NUMBER;
   INITIAL_STATUS : in PICK_STATUS;
   INITIAL_PATH   : in PICK_PATH;
   ECHO_VOLUME    : in DC.RECTANGULAR_REGION_3;
   DATA_RECORD    : in PICK_DATA_RECORD;
   ORDER          : in PATH_ORDER);

procedure INITIALIZE_PICK
  (WS             : in WS_ID;
   DEVICE         : in PICK_DEVICE_NUMBER;
   INITIAL_STATUS : in PICK_STATUS;
   INITIAL_PATH   : in PICK_PATH;
   ECHO_AREA      : in DC.RECTANGULAR_REGION_2;
   DATA_RECORD    : in PICK_DATA_RECORD;
   ORDER          : in PATH_ORDER);

procedure INITIALIZE_STRING
  (WS             : in WS_ID;
   DEVICE         : in STRING_DEVICE_NUMBER;
   INITIAL_STRING : in PHIGS_STRING;
   ECHO_VOLUME    : in DC.RECTANGULAR_REGION_3;
   DATA_RECORD    : in STRING_DATA_RECORD);

procedure INITIALIZE_STRING
```

```
    (WS                 : in WS_ID;
     DEVICE             : in STRING_DEVICE_NUMBER;
     INITIAL_STRING     : in PHIGS_STRING;
     ECHO_AREA          : in DC.RECTANGULAR_REGION_2;
     DATA_RECORD        : in STRING_DATA_RECORD);

  procedure SET_LOCATOR_MODE
    (WS         : in WS_ID;
     DEVICE     : in LOCATOR_DEVICE_NUMBER;
     MODE       : in OPERATING_MODE;
     SWITCH     : in ECHO_SWITCH);

  procedure SET_STROKE_MODE
    (WS         : in WS_ID;
     DEVICE     : in STROKE_DEVICE_NUMBER;
     MODE       : in OPERATING_MODE;
     SWITCH     : in ECHO_SWITCH);

  procedure SET_VALUATOR_MODE
    (WS         : in WS_ID;
     DEVICE     : in VALUATOR_DEVICE_NUMBER;
     MODE       : in OPERATING_MODE;
     SWITCH     : in ECHO_SWITCH);

  procedure SET_CHOICE_MODE
    (WS         : in WS_ID;
     DEVICE     : in CHOICE_DEVICE_NUMBER;
     MODE       : in OPERATING_MODE;
     SWITCH     : in ECHO_SWITCH);

  procedure SET_PICK_MODE
    (WS         : in WS_ID;
     DEVICE     : in PICK_DEVICE_NUMBER;
     MODE       : in OPERATING_MODE;
     SWITCH     : in ECHO_SWITCH);

  procedure SET_STRING_MODE
    (WS         : in WS_ID;
     DEVICE     : in STRING_DEVICE_NUMBER;
     MODE       : in OPERATING_MODE;
     SWITCH     : in ECHO_SWITCH);

  procedure REQUEST_LOCATOR
    (WS             : in   WS_ID;
     DEVICE         : in   LOCATOR_DEVICE_NUMBER;
     STATUS         : out  INPUT_STATUS;
     VIEW_IND       : out  VIEW_INDEX;
     POSITION       : out  WC.POINT_3);
```

```
procedure REQUEST_LOCATOR
  (WS          : in   WS_ID;
   DEVICE      : in   LOCATOR_DEVICE_NUMBER;
   STATUS      : out  INPUT_STATUS;
   VIEW_IND    : out  VIEW_INDEX;
   POSITION    : out  WC.POINT_2);

procedure REQUEST_STROKE
  (WS                 : in   WS_ID;
   DEVICE             : in   STROKE_DEVICE_NUMBER;
   STATUS             : out  INPUT_STATUS;
   VIEW_IND           : out  VIEW_INDEX;
   STROKE_POINTS      : out  WC.ACCESS_POINT_LIST_3);

procedure REQUEST_STROKE
  (WS                 : in   WS_ID;
   DEVICE             : in   STROKE_DEVICE_NUMBER;
   STATUS             : out  INPUT_STATUS;
   VIEW_IND           : out  VIEW_INDEX;
   STROKE_POINTS      : out  WC.ACCESS_POINT_LIST_2);

procedure REQUEST_VALUATOR
  (WS       : in   WS_ID;
   DEVICE   : in   VALUATOR_DEVICE_NUMBER;
   STATUS   : out  INPUT_STATUS;
   VALUE    : out  VALUATOR_VALUE);

procedure REQUEST_CHOICE
  (WS       : in   WS_ID;
   DEVICE   : in   CHOICE_DEVICE_NUMBER;
   STATUS   : out  CHOICE_REQUEST_STATUS;
   CHOICE   : out  CHOICE_NUMBER);

procedure REQUEST_PICK
  (WS                 : in   WS_ID;
   DEVICE             : in   PICK_DEVICE_NUMBER;
   DEPTH_TO_RETURN    : in   PATH_DEPTH;
   STATUS             : out  PICK_REQUEST_STATUS;
   PATH               : out  PICK_PATH);

procedure REQUEST_STRING
  (WS                 : in   WS_ID;
   DEVICE             : in   STRING_DEVICE_NUMBER;
   STATUS             : out  INPUT_STATUS;
   CHAR_STRING        : out  INPUT_STRING);

procedure SAMPLE_LOCATOR
  (WS          : in   WS_ID;
   DEVICE      : in   LOCATOR_DEVICE_NUMBER;
```

```
    VIEW_IND  : out  VIEW_INDEX;
    POSITION  : out  WC.POINT_3);

procedure SAMPLE_LOCATOR
  (WS         : in   WS_ID;
   DEVICE     : in   LOCATOR_DEVICE_NUMBER;
   VIEW_IND   : out  VIEW_INDEX;
   POSITION   : out  WC.POINT_2);

procedure SAMPLE_STROKE
  (WS                 : in   WS_ID;
   DEVICE             : in   STROKE_DEVICE_NUMBER;
   VIEW_IND           : out  VIEW_INDEX;
   STROKE_POINTS      : out  WC.ACCESS_POINT_LIST_3);

procedure SAMPLE_STROKE
  (WS                 : in   WS_ID;
   DEVICE             : in   STROKE_DEVICE_NUMBER;
   VIEW_IND           : out  VIEW_INDEX;
   STROKE_POINTS      : out  WC.ACCESS_POINT_LIST_2);

procedure SAMPLE_VALUATOR
  (WS       : in   WS_ID;
   DEVICE   : in   VALUATOR_DEVICE_NUMBER;
   VALUE    : out  VALUATOR_VALUE);

procedure SAMPLE_CHOICE
  (WS       : in    WS_ID;
   DEVICE   : in   CHOICE_DEVICE_NUMBER;
   STATUS   : out  CHOICE_STATUS;
   CHOICE   : out  CHOICE_NUMBER);

procedure SAMPLE_PICK
  (WS                 : in   WS_ID;
   DEVICE             : in   PICK_DEVICE_NUMBER;
   DEPTH_TO_RETURN    : in   PATH_DEPTH;
   STATUS             : out  PICK_STATUS;
   PATH               : out  PICK_PATH);

procedure SAMPLE_STRING
  (WS                 : in   WS_ID;
   DEVICE             : in   STRING_DEVICE_NUMBER;
   CHAR_STRING        : out  INPUT_STRING);

procedure AWAIT_EVENT
  (TIMEOUT  : in   DURATION;
   WS       : out  WS_ID;
   DEVICE   : out  EVENT_DEVICE_NUMBER);
```

```
      procedure FLUSH_DEVICE_EVENTS
        (WS      : in WS_ID;
         DEVICE  : in EVENT_QUEUE_DEVICE_NUMBER);

      procedure GET_LOCATOR
        (VIEW_IND  : out VIEW_INDEX;
         POSITION  : out WC.POINT_3);

      procedure GET_LOCATOR
        (VIEW_IND  : out VIEW_INDEX;
         POSITION  : out WC.POINT_2);

      procedure GET_STROKE
        (VIEW_IND       : out VIEW_INDEX;
         STROKE_POINTS : out WC.ACCESS_POINT_LIST_3);

      procedure GET_STROKE
        (VIEW_IND       : out VIEW_INDEX;
         STROKE_POINTS : out WC.ACCESS_POINT_LIST_2);

      procedure GET_VALUATOR
        (VALUE : out VALUATOR_VALUE);

      procedure GET_CHOICE
        (STATUS : out CHOICE_STATUS;
         CHOICE : out CHOICE_NUMBER);

      procedure GET_PICK
        (DEPTH_TO_RETURN : in   PATH_DEPTH;
         STATUS          : out  PICK_STATUS;
         PATH            : out  PICK_PATH);

      procedure GET_STRING
        (CHAR_STRING : out INPUT_STRING);

    -- METAFILE FUNCTIONS

      procedure WRITE_ITEM_TO_METAFILE
        (WS    : in WS_ID;
         ITEM  : in METAFILE_DATA_RECORD);

      procedure GET_ITEM_TYPE_FROM_METAFILE
        (WS              : in   WS_ID;
         TYPE_OF_ITEM    : out  METAFILE_ITEM_TYPE;
         LENGTH          : out  METAFILE_ITEM_LENGTH);

      procedure READ_ITEM_FROM_METAFILE
        (WS              : in   WS_ID;
         MAX_LENGTH      : in   METAFILE_ITEM_LENGTH;
         ITEM            : out  METAFILE_DATA_RECORD);
```

```
      procedure INTERPRET_ITEM
        (ITEM : in METAFILE_DATA_RECORD);

    -- INQUIRY FUNCTIONS

      procedure INQ_SYSTEM_STATE_VALUE
        (STATE_VALUE : out SYSTEM_STATE);

      procedure INQ_WS_STATE_VALUE
        (STATE_VALUE : out WS_STATE);

      procedure INQ_STRUCTURE_STATE_VALUE
        (STATE_VALUE : out STRUCTURE_STATE);

      procedure INQ_ARCHIVE_STATE_VALUE
        (STATE_VALUE : out ARCHIVE_STATE);

      procedure INQ_LIST_OF_AVAILABLE_WS_TYPES
        (ERROR_INDICATOR  : out ERROR_NUMBER;
         LIST_OF_TYPES    : out WS_TYPES.LIST_OF);

      procedure INQ_PHIGS_FACILITIES
        (ERROR_INDICATOR                : out ERROR_NUMBER;
         MAX_SIMUL_OPEN_WS              : out PHIGS_POSITIVE;
         MAX_SIMUL_OPEN_ARCHIVES        : out PHIGS_POSITIVE;
         NUMBER_AVAIL_NAMES             : out PHIGS_POSITIVE;
         AVAIL_CHAR_SETS                : out CHAR_SETS.LIST_OF;
         MAX_ISS_NORMAL_FILTER_LIST     : out PHIGS_POSITIVE;
         MAX_ISS_INVERTED_FILTER_LIST   : out PHIGS_POSITIVE);

      procedure INQ_GSE_FACILITIES
        (ERROR_INDICATOR      : out ERROR_NUMBER;
         LIST_AVAIL_GSES      : out GSE_IDS.LIST_OF;
         LIST_WS_DEPENDENCIES : out WS_DEPENDENCIES.LIST_OF);

      procedure INQ_MODELLING_CLIPPING_FACILITIES
        (ERROR_INDICATOR      : out ERROR_NUMBER;
         NUMBER_DISTINCT_PLANES
                              : out PHIGS_POSITIVE;
         LIST_OF_OPERATIONS   : out MODELLING_CLIP_OPERATION_TYPES.LIST_OF);

      procedure INQ_EDIT_MODE
        (ERROR_INDICATOR  : out ERROR_NUMBER;
         MODE             : out EDIT_MODE);

      procedure INQ_SET_OF_OPEN_WS
        (ERROR_INDICATOR  : out ERROR_NUMBER;
         OPEN_WS          : out WS_IDS.LIST_OF);

      procedure INQ_STRUCTURE_IDENTIFIERS
```

```
  (ERROR_INDICATOR     : out ERROR_NUMBER;
   LIST_OF_STRUCTURES : out STRUCTURE_IDS.LIST_OF);

procedure INQ_ARCHIVE_FILES
  (ERROR_INDICATOR            : out ERROR_NUMBER;
   LIST_ARCHIVE_IDENTIFIERS : out ARCHIVE_IDS.LIST_OF;
   LIST_ARCHIVE_FILES          : out VARIABLE_FILE_IDS.LIST_OF);

procedure INQ_CONFLICT_RESOLUTION
  (ERROR_INDICATOR     : out ERROR_NUMBER;
   ARCHIVAL_CONFLICT   : out CONFLICT_RESOLUTION;
   RETRIEVAL_CONFLICT  : out CONFLICT_RESOLUTION);

procedure INQ_ALL_CONFLICTING_STRUCTURES
  (ARCHIVE_IDENTIFIER  : in  ARCHIVE_ID;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   LIST_OF_STRUCTURES  : out STRUCTURE_IDS.LIST_OF);

procedure INQ_CONFLICTING_STRUCTURES_IN_NETWORK
  (ARCHIVE_IDENTIFIER     : in  ARCHIVE_ID;
   STRUCTURE_IDENTIFIER : in  STRUCTURE_ID;
   SOURCE                  : in  STRUCTURE_NETWORK_SOURCE;
   ERROR_INDICATOR         : out ERROR_NUMBER;
   LIST_OF_STRUCTURES      : out STRUCTURE_IDS.LIST_OF);

procedure INQ_MORE_SIMULTANEOUS_EVENTS
  (ERROR_INDICATOR: out ERROR_NUMBER;
   EVENTS                : out MORE_EVENTS);

procedure INQ_WS_CONNECTION_AND_TYPE
  (WS                  : in  WS_ID;
   ERROR_INDICATOR: out ERROR_NUMBER;
   CONNECTION          : out VARIABLE_CONNECTION_ID;
   TYPE_OF_WS          : out WS_TYPE);

procedure INQ_LIST_OF_VIEW_INDICES
  (WS                     : in  WS_ID;
   ERROR_INDICATOR   : out ERROR_NUMBER;
   DEFINED_VIEW_LIST : out VIEW_INDICES.LIST_OF);

procedure INQ_VIEW_REPRESENTATION
  (WS                                  : in  WS_ID;
   VIEW_IND                            : in  VIEW_INDEX;
   ERROR_INDICATOR                     : out ERROR_NUMBER;
   UPDATE                              : out UPDATE_STATE;
   REQUESTED_ORIENTATION_MATRIX : out TRANSFORMATION_MATRIX_3;
   CURRENT_ORIENTATION_MATRIX   : out TRANSFORMATION_MATRIX_3;
   REQUESTED_MAPPING_MATRIX     : out TRANSFORMATION_MATRIX_3;
   CURRENT_MAPPING_MATRIX       : out TRANSFORMATION_MATRIX_3;
```