PHIGS

PHIGS is an international **ISO standard of functional interface** between an application program and a configuration of graphical input and output devices.

This interface contains basic functions for dynamic interactive 2D and 3D graphics on wide variety of graphics equipment

1. PHIGS concept and graphical workstation

PHIGS is a high level graphics library **with over 400 functions**.

It allows an application programmer **to describe a model of a scene, to display the model on workstation, to manipulate and to edit the model interactively**

The models are stored in a graphical database known as **centralized structure store** (CSS).

The fundamental entity of data is a **structure element** and these are grouped together into units called **structures**.

Structures are organized as acyclic directed graphics called **structure networks**

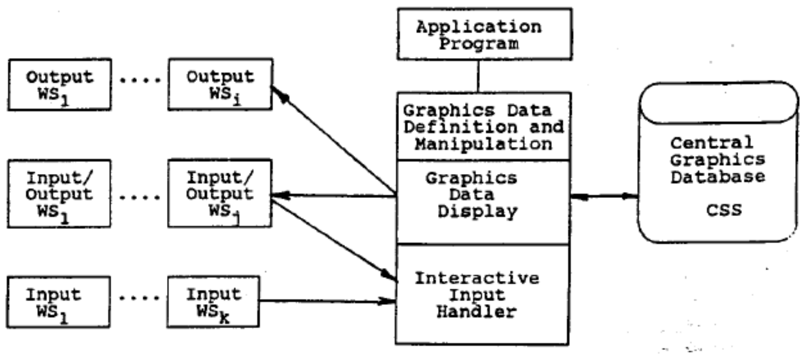


Fig. 1 Structure of the PHIGS

The two abstract concepts (input and output) are the building blocks of an **abstract workstation**.

A **PHIGS workstation** represents a unit consisting of zero or one **display surfaces** and zero or more **input devices** such as keyboard, tablet , mouse and light pen.

The workstation presents this devices to the application program as a configuration of abstract devices thereby **shielding the hardware peculiarities**

Each workstation has a type falling into one of five categories

```
OUTPUT – supports output only

INPUT – supports input only
```

OUTIN – supports output and input

MO – supports output graphical and application data to the external storage

MI- supports input graphical and application data from the external storage to the application

For every type of workstation present in a PHIGS implementation**, there exists a generic workstation description table** which describes the standard capabilities and characteristics of the workstation.

When the workstation is opened, a **new specification workstation description table is created** for that workstation description table obtained from the device itself. And possibly from other implementation dependent source.

The content of the specific workstation description table may change at any time while the workstation is open.

The application program can inquire which generic capabilities are available before the workstation is open.

The specific capabilities may be inquired while the workstation is open by first inquiring the workstation type of an open workstation to obtain the workstation type of the specific workstation description table, and then using this workstation type as a parameter to the inquiry functions which query the workstation description table. This information may be used by the application program to adapt its behavior accordingly

The application program references a workstation by means of a workstation identifier. Connection to a particular workstation is established by the function OPEN WORKSTATION which associates the workstation identifier with a generic workstation type and a connection identifier.

The current state of each open workstation . state values of the WSSL maybe set up by the “set functions” and may be inquired by “inquire functions”.

2. Structure Entity

A structure element is the fundamental entity of data. Structure elements are used to represent application specified graphics data for output primitives, attribute selections, modeling transformations and clipping , invocation of other structures, and to represent application data.

The following types of structure elements are defined in PHIGS

- Output primitive structure elements
- Attribute specification structure elements
- Modeling transformation and clipping structure elements

- Control structure element
- Editing structure element
- Generalized structure element
- Application data
- Graphical output

Graphical Output

Picture generated by PHIGS are build up of basic pieces called output primitives. Output primitives are generated from structure elements by structural traversal

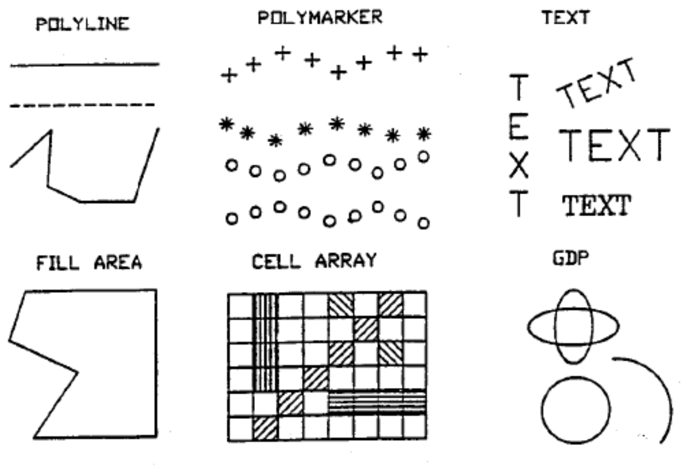


Fig. 2 Examples of output primitives

POLYLINE set of connected lines defined by point sequences POLYLINE3

POLYMARKER set of symbols of one type centered at a given position POLYMARKER 3

TEXT character string at a given position on an arbitrary plane in modeling coordinate space

ANNOTATION TEXT RELATIVE character string at specified position in an x y plane parallel to view plane ANNOTATION TEXT RELATIVE 3

FILL AREA single polygonal area which may be hollow or filled with uniform color, pattern or hatch style

FILL AREA SET a set of polygonal areas which may be filled similar as FILL AREA

CELL ARRAY two dimensional array of cells with individual colo9ur4s

GENERALIZED DRAWING PRIMITIVE special geometrical output capabilities of a workstation such as the drawing of spline curves, circular arcs, elliptic arcs

for individual specifications of aspects, there is a separate attribute for each aspect. These attributes are workstation independent

bundled aspects are selected by a bundle index into a bundle table each entry of which contains non geometric aspects of a primitive. The non geometric aspects are workstation dependent in that each workstation has its own set of bundle tables (stored in the workstation state list) the values in a particular bundle may be different for different workstations

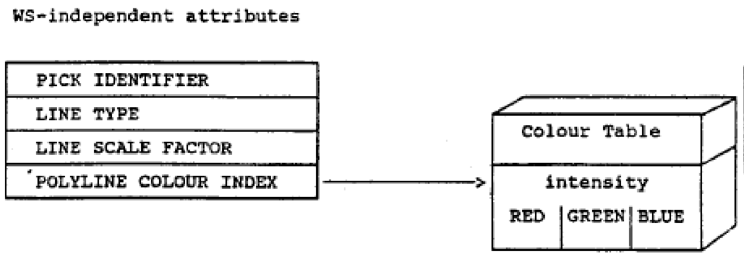


Fig. 3 Individual specification of attributes

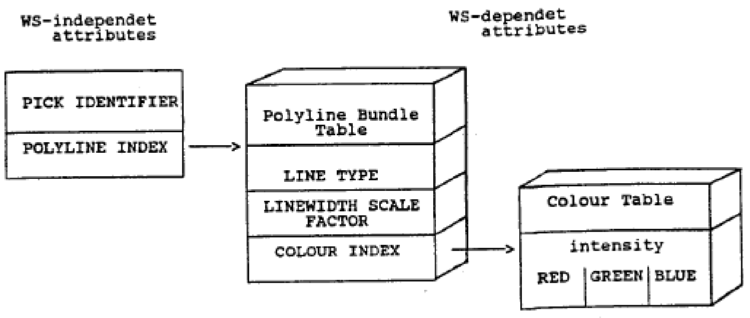


Fig. 4 Bundled specification of attributes

Output primitive attributes

Attributes of the first type control the geometric aspects of primitives. These are aspects that affect the shape or size for the entire primitive (e.g. CHARACTER HEIGHT for TEXT)

Hence they are sometimes referred to as geometric attributes. Geometric attributes are workstation independent and if they represent coordinate data they are expressed in modeling coordinates

Attributes of the second type control the non geometric aspects of primitives these are aspects that affect a primitive's appearance(for example. Line type for POLYLINE, or color index for all primitives except CELL ARRAY) or the shape or size of the component parts of the primitive (for example , marker size scale factor for POLYMARKER)

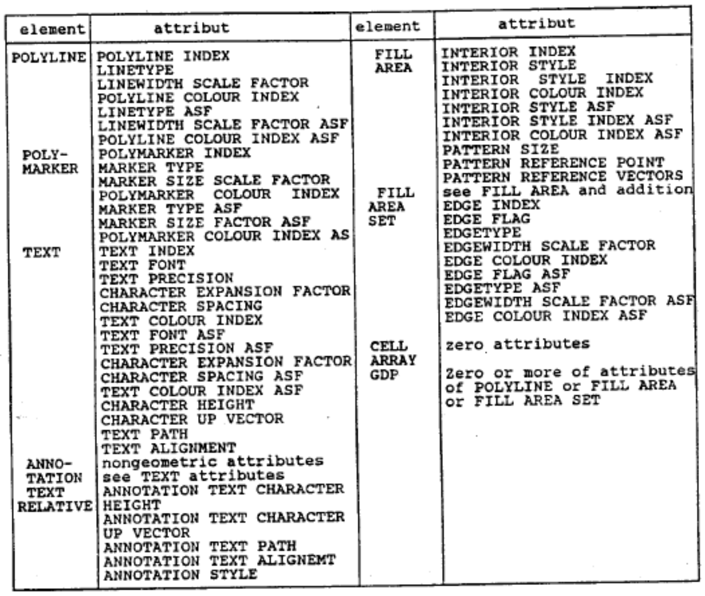
Non geometric aspects do not represent coordinate data

The non geometric aspects of primitive may be specified either via a bundle or individually the geometric aspects only individually

3. Structure and structure networks

PHIGS supports the storage and manipulation of data in CSS the CSS contains graphical and application data organized into units called **structures** which may be related each other hierarchically to form structure networks

Each structure is identified by unique name which is specified by the application



element	attribut	element	attribut
POLYLINE	POLYLINE INDEX LINETYPE LINEWIDTH SCALE FACTOR POLYLINE COLOUR INDEX LINETYPE ASF LINEWIDTH SCALE FACTOR ASF POLYLINE COLOUR INDEX ASF	FILL AREA	INTERIOR INDEX INTERIOR STYLE INTERIOR STYLE INDEX INTERIOR COLOUR INDEX INTERIOR STYLE ASF INTERIOR STYLE INDEX ASF INTERIOR COLOUR INDEX ASF PATTERN SIZE PATTERN REFERENCE POINT PATTERN REFERENCE VECTORS
POLY-MARKER	POLYMARKER INDEX MARKER TYPE MARKER SIZE SCALE FACTOR POLYMARKER COLOUR INDEX MARKER TYPE ASF MARKER SIZE FACTOR ASF POLYMARKER COLOUR INDEX AS	FILL AREA SET	see FILL AREA and addition EDGE INDEX EDGE FLAG EDGETYPE EDGEWIDTH SCALE FACTOR EDGE COLOUR INDEX EDGE FLAG ASF EDGETYPE ASF EDGEWIDTH SCALE FACTOR ASF EDGE COLOUR INDEX ASF
TEXT	TEXT INDEX TEXT FONT TEXT PRECISION CHARACTER EXPANSION FACTOR CHARACTER SPACING TEXT COLOUR INDEX TEXT FONT ASF TEXT PRECISION ASF CHARACTER EXPANSION FACTOR CHARACTER SPACING ASF TEXT COLOUR INDEX ASF CHARACTER HEIGHT CHARACTER UP VECTOR TEXT PATH TEXT ALIGNMENT	CELL ARRAY	zero attributes
ANNO-TATION TEXT RELATIVE	nongeometric attributes see TEXT attributes ANNOTATION TEXT CHARACTER HEIGHT ANNOTATION TEXT CHARACTER UP VECTOR ANNOTATION TEXT PATH ANNOTATION TEXT ALIGNEMT ANNOTATION STYLE	GDP	Zero or more of attributes of POLYLINE or FILL AREA or FILL AREA SET

Structure networks are organized as directed acyclic graphs

So a structure may contain invocations of other structures containing in CSS. The invocation of a structure is achieved using the **execute structure element**. Such an invocation is known as a **structure reference**. Structure can not be referenced recursively

A structure can be created in one of the following ways:

- When a reference to the nonexistent structure is inserted into a structure in the CSS
- When the structure is opened for the first time (function `OPEN STRUCTURE`)
- When the structure is posted for display on a workstation (`POST STRUCTURE`)
- When the structure is referenced in any function changing the structure identifier
- When the not existing structure in CSS is retrieved from an archive (`RETRIEVE STRUCTURE`)
- When the not existing structure is emptied (`EMPTY STRUCTURE`)

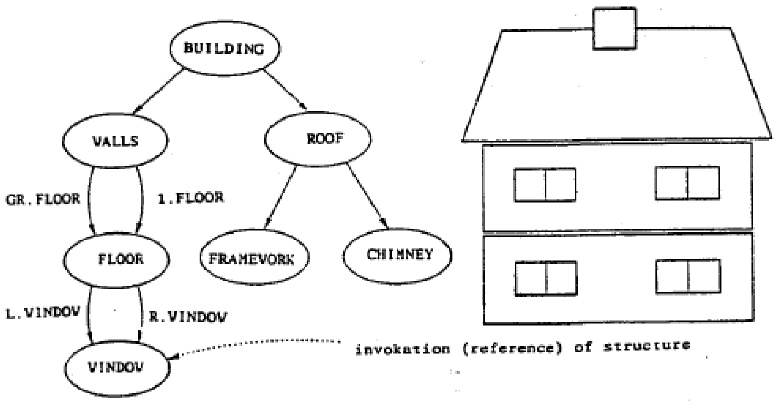


Fig. 5 Building and its structure network

Structure traversal and display

- A structure network is identified for display on a workstation by posting function POST STRUCTURE

A structure may be displayed only if it is a member of a posted structure network

To display a network the structure elements have to be extracted from the CSS and processed . that process is called traversal process

The traversal process interprets each structure element in the structure network sequentially starting at the first element of the top of the network

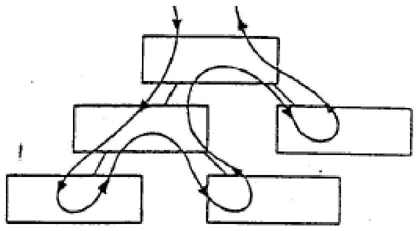


Fig. 6 Traversal process

A traversal state list is associated with each traversal process. Values in this state list ;may be accessed when the structure elements are interpreted

Each time a traversal is initiated the associated list is initialized

Structure editing

Each element within a structure can be accessed and modified individually with editing functions to inset new elements, replace elements with new structure elements, delete structure elements. Navigate within the structure and inquire structure element content

A structure is identified for editing an element pointer is established which points at the lat element in the structure . functions for positioning element pointer

SET ELEMENT POINTER – set to an absolute position

SET OFFSET ELEMENT POINTER – set relative to current position

SET ELEMENT POINT AT LABEL – set a position of the specified label structure element

The edit mode defined by the function SET EDIT MODE defines whether new elements replace the element pointed to by the element pointer or are added after the element pointed to by the element pointer

Functions for editing:

COPY ALL ELEMENTS FROM STRUCTURE- copy all the elements of a structure into the open structure

DELETE ELEMENT – delete element at which the element pointer is pointing

DELETE ELEMENT RANGE- delete a group of elements between two element positions

DELETE ELEMENT LABELS-delete group of elements delimited by labels

EMPTY STRUCTURE-delete all elements of the structure

Manipulation of structures in CSS

Operations for manipulation of structure

DELETE STRUCTURE- delete structure ;and all references to it

DELETE ALL STRUCTURES- delete all structure from the CSS

DELETE STRUCTURE NETWORK-delete the indicated structure and all its ancestors

CHANGE STRUCTURE IDENTIFIER-change identifier specified structure

CHANGE STRUCTURE REFERENCES-change all references of specified structure

ELEMENT SEARCH –search within a single structure for an element of a particular element type

Structure archival and retrieval

OPEN ARCHIVE FILE , CLOSE ARCHIVE FILE-initiates or terminates access to archive file

ARCHIVE ALL STRUCTURES-storing of structure to archive file

RETRIEVE ALL STRUCTURES- recovers structures from an archive file

DELETE STRUCTURES FROM ARCHIVE-deletes structure in an archive file

4. Coordinate systems and transformations

Structures represent parts of a hierarchical model of modeling scene. Each of these parts has own world space represented by modeling coordinate system .

The relative positioning of the separate parts is achieved by having a single World coordinate space onto which all the defined modeling coordinate systems are mapped by a composite modeling transformation during the traversal process

The world coordinate space can be regarded as a workstation independent abstract viewing space

The workstation dependent stage then performs a transformation on the geometrical information contained in output primitives , attributes and logical input values

These transformations perform mapping between four coordinate systems:

- **World coordinates** used to define uniform coordinate system for all abstract workstation
- **View reference coordinate** used to define a view
- **Normalized projection coordinates**; used to facilitate assemblies of different views

- **Device coordinates**: one coordinate system per workstation representing its display space

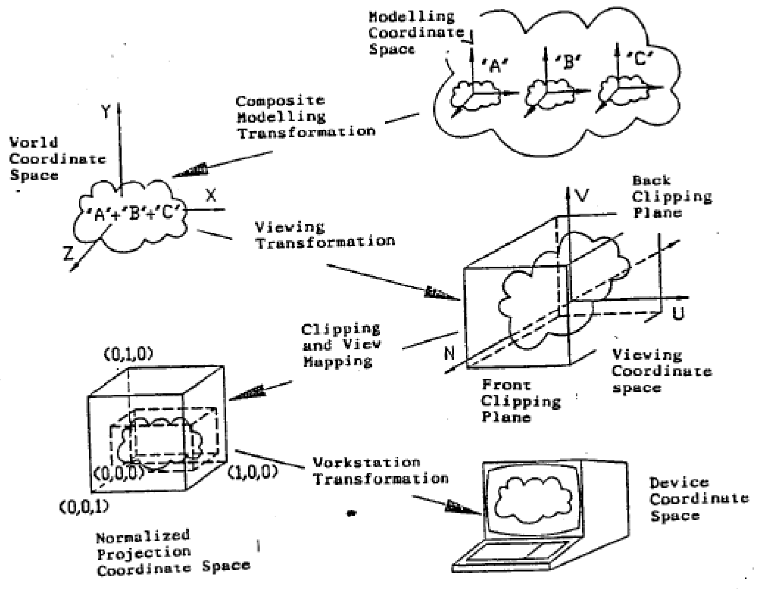


Fig. 7 Coordinate systems and transformations in PHIGS

Output primitives and attributes are mapped from world coordinate to view reference coordinate by the **view orientation transformation**, from view reference coordinates to normalized projection coordinates by the **view mapping transformation** and from normalized projection coordinates to device coordinates by the **workstation transformation**. Hidden lines removals and clippings are also done during the mapping process

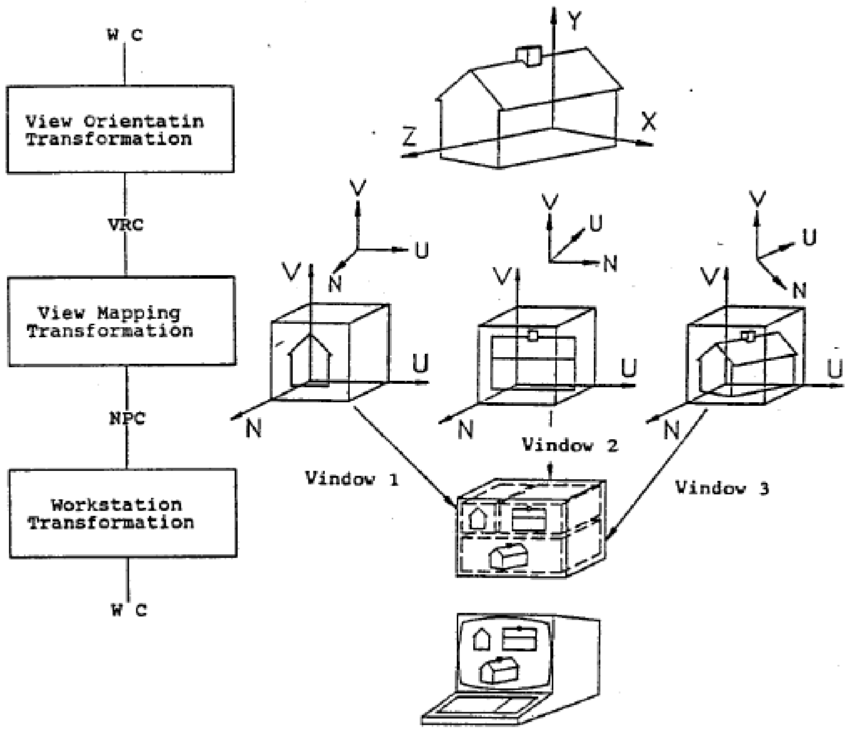

W C
View Orientatin Transformation
VRC
View Mapping Transformation
NPC
Workstation Transformation
W C
Y
X
Z
V
U
N
Window 1
Window 2
Window 3

Fig.8 Viewing transformations in PHIGS

5. Graphical input

An application program gets graphical input from an operator by controlling the activity of one or more logical input devices.

6. Language Interfaces

PHIGS defines only a language independent nucleus of a graphics system

For integration into a language, PHIGS is embedded in a language dependent layer containing the language conventions, e.g. parameter and name assignment

In case of the layer model, each layer may call the functions of the adjoining lower layers. In general the application program uses the application oriented layer, the language dependent layer, other application dependent layers and operations system resources. There are standards of the language dependent layers for the language FORTRAN, PASCAL and C

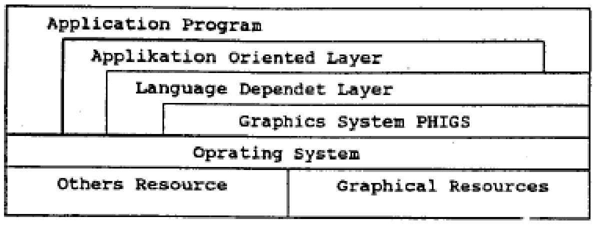


Fig. 9. Layer model of PHIGS

7. Graphics system PHIGS +

PHIGS does not support a shading of pictures and modeling a non uniform rational B Spline curves and surfaces (NURBS)

An extension of PHIGS for these functionalities is PHIGS+

PHIGS+ enables to specify light sources

Parameters of the light sources are described and application program s may set the up

The predefined light sources are ambient, directional, positional, spot light source.

Shading method is an attribute of the graphics primitives.

```
NONE- No shading

COLOUR-color interpolation shading

NORMAL-normal interpolation shading

DOT PRODUCT- dot product interpolation shading
```

Light type accepted for shading

NONE – no reflectance calculation performed

AMBIENT-use ambient term

AMB_DIF-use ambient and diffuse terms

AMB_DIF_SPEC- use ambient, diffuse and specular terms

PHIGS+ offers additional output primitives and functions;

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

FILL AREA SET3 WITH DATA-creates a 3D fill area set structure element that includes color and shading data

QUADRILATERAL MESH3 WITH DATA-creates a 3D quadrilateral mesh primitive with color and shading data

TRIANGLE STRIP3 WITH DATA-creates a 3D triangle strip primitive with color and shading data

NON UNIFORM B-SPLINE CURVE NURBS-crates a structure element containing the definition of a non-uniform B-Spline Curve

Significance of PHIGS

Portability of program

- PHIGS is computer and device independent graphics system. The application program that utilize PHIGS can be easily transported between host processors and graphics devices

Sophisticated capabilities save development time

- PHIGS manages the storage and display of 2D and 3D graphical data, crates and maintains a hierarchical database

Increased program performance because of fewer error conditions

- Application using PHIGS have well defined inputs and outputs that minimizes errors