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 a workstation, to manipulate and to edit the model interactively. The models are stored in a graphical database, known as a 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 graphs called structure networks.

![Diagram of the PHIGS system](image)

**Fig. 1. Structure of the PHIGS**

**Key words:** Computer Graphics, Graphics Systems, Graphics Standards.

**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 a 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 graphs called structure networks.
The two abstract concepts (abstract output and abstract input) are the building blocks of an abstract workstation. A PHIGS workstation represents a unit consisting of zero or one display surfaces an zero or more input devices, such as keyboard, tablet, mouse and lightpen. 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 applicatin data to the external storage,
- MI - supports input graphical and application data from the external storage to the applicatin program.

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 specific workstation description table (WSDT) is created for that workstation containing information which is derived from the generic workstation description table, obtained from the device itself, and possibly from other implementation dependent sources. 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 behaviour 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 is kept in a workstation state list (WSSL) for that workstation. State values of the WSSL may be set up by "set functions" and may be inquired by "inquire functions".

2 Structure elements

A structure element is the fundamental entity of data. Structure elements are used to represent application specified graphics data for output primitives, attribute selections, modelling 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.
- Modelling transformation and clipping structure elements.
- Control structure element (execute structure).
- Editing structure element (label).
- Generalized structure element.
- Application data.

2.1 Graphical output

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

![Fig. 2 Examples of output primitives](image)
POLYLINE set of connected lines defined by a point sequens
POLYLINE3 set of symbols of one type centred at given position
POLYMARKER 3 character string at a given position on an arbitrary
TEXT plane in modelling coordinate space (see below)
TEXT 3 ANNOTATION TEXT RELATIVE character string at specified position
ANNOTATION TEXT RELATIVE 3 in an x-y plane parallel to view plane
FILL AREA single polygonal area which may be hollow or filled
FILL AREA 3 with uniform lcolour, a pattern, or a hatch style
FILL AREA SET set of polygonal areas which may be filled similar
FILL AREA SET 3 as FILL AREA
CELL ARRAY two dimensional array of cells with individual colours
CELL ARRAY 3
GENERALIZED DRAWING PRIMITIVE special geometrical output capa-
GENERALIZED DRAWING PRIMITIVE 3 bilities of a workstation such as
the drawing of spline curves, circular arcs, elliptic arcs,

For individual specification 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-geometrical 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 (or entry in the bundle table) may be different for different workstations.

WS-independent attributes

<table>
<thead>
<tr>
<th>PICK IDENTIFIER</th>
<th>POLYLINE INDEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINE TYPE</td>
<td>LINE TYPE</td>
</tr>
<tr>
<td>LINE SCALE FACTOR</td>
<td>SCALE FACTOR</td>
</tr>
<tr>
<td>POLYLINE COLOUR INDEX</td>
<td>COLOUR INDEX</td>
</tr>
</tbody>
</table>

Fig. 3 Individual specification of attributes

Fig. 4 Bundled specification of attributes

2.2 Output primitive attributes

Attributes of the first type control the geometric aspects of primitives. These are aspects that affect the shape or size of the entire primitive (for example, CHARACTER HEIGHT for TEXT). Hence they are sometimes referred to as geometric attributes. Geometric attributes are workstations independent and, if they represent coordinate data (points or displacements), they are expressed in modelling coordinates.

Attributes of the second type control the non-geometric aspects of primitives. These are aspects that affect a primitive's appearance (for example, linetype for POLYLINE, or colour 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 a primitive may be specified either via a bundle or individually, the geometric aspects only individually.


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 is specified by the application.
Structure networks are organized as directed acyclic graphs. That is, 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 (structure networks is acyclic graphs).

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

a) when a reference to the non-existent structure is inserted into a structure in the CSS,

b) when the structure is opened for the first time (function `OPEN STRUCTURE`),

c) when the structure is posted for display on a workstation (POST STRUCTURE),

d) when the structure is referenced in any function changing the structure identifier,

e) when the not existing structure in CSS is retrieved from an archive (RETRIEVE STRUCTURE),

f) when the not existing structure is emptied (EMPTY STRUCTURE)

Fig. 5 Building and its structure network

3.1 Structure traversal and display

A structure network is identified for display on a workstation by the 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.
A traversal state list (TRSL) 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 TRSL is initialized.

The effect of interpreting a structure element depends on the type of the structure element. Output primitive elements result in creation of output primitives. Structure elements of other types either modify values in TRSL, or are ignored. When an “execute structure element is encountered during the traversal, the following actions occur:

a) traversal of the current structure is suspended,
b) the current state of TRSL is saved,
c) global and local modelling transformation are saved (see below),
d) executed structure network is completely traversed,
e) the saved TRSL values are restored,
f) traversal of the structure is resumed.

3.2 Structure editing

PHIGS provides the ability to individually access and modify each element within a structure. Editing functions are available to insert 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 by the OPEN STRUCTURE function. When a structure is opened for editing an element pointer is established which points at the last element in the structure. The element pointer may be positioned using the function:

SET ELEMENT POINTER - set to an absolute position,
SET OFFSET ELEMENT POINTER - set relative to current position,
SET ELEMENT POINTER AT LABEL - set a position of the specified label structure element.

When a structure is open the application program may insert additional elements into the structure or replace existing structure elements using the functions which define structure elements (e.g., POLYLINE, SET POLYLINE INDEX, ...). The edit mode as 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. There are spatial functions for editing:

COPY ALL ELEMENTS FROM STRUCTURE - copy all the elements of a structure into the open structure,
DELETE ELEMENT - delete the element at which the element pointer is pointing,
DELETE ELEMENT RANGE - delete a group of the elements between two element positions,
DELETE ELEMENT LABELS - delete group of elements delimited by labels,
EMPTY STRUCTURE - delete all elements of the structure.

The function INQUIRE CURRENT ELEMENT TYPE AND SIZE returns the type of the element and its size, the function INQUIRE CURRENT ELEMENT CONTENT returns the parameters of the element.

3.3 Manipulation of structures in CSS

The following operations apply to structures as whole units in the CSS:
DELETE STRUCTURE - delete structure and all references to it,
DELETE ALL STRUCTURES - delete all structures 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 the specified structure,
CHANGE STRUCTURE IDENTIFIER AND REFERENCES - change the identifier and all references of the specified structure.

PHIGS provides searching and inquiry functions for CSS:

INQUIRE PATHS TO ANCESTORS - returns path which reference the specified structure,
INQUIRE PATHS TO DESCENDANTS - returns path which are referenced by a particular structure,
ELEMENT SEARCH - searching within a single structure for an element of a particular element type or one of a set of element types,
INCREMENTAL SPATIAL SEARCH - allows a structure network to be searched for the next occurrence of a structure element which satisfied search criteria.

3.4 Structure archival and retrieval

An archive file (AF) is a medium for storing structure definitions. PHIGS functionality provides for archiving from the CSS to an
AF, retrieving from an AF to the CSS, or deleting from an AF. This set of functions may be applied to a list of structure networks, or to all structures defined either in CSS or in an AF. There are these functions:

OPEN ARCHIVE FILE, CLOSE ARCHIVE FILE - initiates or terminates access to AF,
ARCHIVE STRUCTURES, ARCHIVE STRUCTURE NETWORKS, ARCHIVE ALL STRUCTURES - storing of structures to AF,
RETRIEVE STRUCTURE IDENTIFIERS - returns the identifiers of structures in an AF,
RETRIEVE STRUCTURES, RETRIEVE STRUCTURE NETWORKS,
RETRIEVE ALL STRUCTURES - recovers structures from an AF,
DELETE STRUCTURES FROM ARCHIVE, DELETE STRUCTURE NETWORKS FROM ARCHIVE, DELETE ALL STRUCTURES FROM ARCHIVE - deletes structures in an AF,
RETRIEVE ANCESTORS OF STRUCTURE, RETRIEVE DESCENDANTS OF STRUCTURE returns path which reference the specified structure resp.
returns path which are referred by a particular structure.

4 Coordinate systems and transformations

Geometrical informations are usually coordinates of points. We suppose that structures represent parts of a hierarchical model of the modelling scene. Each of these parts has its own world space represented by a modelling coordinate system (MC). The relative positioning of the separate parts is achieved by having a single World Coordinate Space (WC) onto which all the defined modelling coordinate systems are mapped by a composite modelling transformation during the traversal process. The WC space can be regarded as a workstation independent abstract viewing space.

The workstation dependent stage then performs a transformation on the geometrical information contain in output primitives, attributes and logical input values (see below). These transformations perform mapping between four coordinate systems, namely:

a) World Coordinates used to define a uniform coordinate system for all abstract workstation,
b) View Reference Coordinates (VRC), used to define a view,
c) Normalized Projection Coordinates (NPC) used to facilitate assemblies of different views,
d) Device Coordinates (DC), one coordinate system per workstation representing its display space.

Fig. 7 Coordinate systems and transformations in PHIGS

Output primitives and attributes are mapped from WC to VRC by the view orientation transformation, from VRC to NPC by the view mapping transformation and finally from NPC to DC by the workstation transformation. During the mapping may be applied the clipping of portions of graphical output which lie outside one or more of the specified regions. Hidden lines and hidden surfaces may be also removed.
STROKE : a sequence of points in world coordinates and index of view transformation,
VALUATOR: a real number,
CHOICE : a CHOICE status and positive integer that represents a selection from a number of choices,
PICK : a PICK status, a pick path depth, and a pick path of picked structure,
STRING : a character string.

Each logical input device can be operated in three modes, called operating modes. At any time a logical input device is in one, and only one, of the modes set by the invocation of a function in the group SET <input class> MODE. The three operating modes are REQUEST, SAMPLE and EVENT. Input from devices is obtained in different ways depending on the mode as follows:

REQUEST : A specific invocation of REQUEST <input class> causes an attempt to read a logical input value from a specified logical input device. This can only occur when the logical input device is in REQUEST mode. PHIGS waits until the input is entered by the operator or a break action is performed by the operator. The break action is dependent on the logical input device and on the implementation. If a break occurs, the logical input value is not valid.

SAMPLE : A specific invocation of SAMPLE <input class> causes PHIGS, without waiting for an operator action, to return the current logical input value of a specified logical input device. This can only occur when the logical input device is in SAMPLE mode.

EVENT : PHIGS maintains one input queue containing temporally ordered event reports. An event report contains the identification of a logical input device and a logical input value from that device. Event reports are generated asynchronously, by operator action only, from input devices in EVENT mode. The application program can remove the oldest event report from the queue, and examine its contents. The application can also flush from the queue all event reports from a specified logical input device.

When an input device is in REQUEST mode, input operations can start by a prompt and they can terminate by a echo. For that reason it is possible to set certain parameters associated with LID of each input class. This initialization procedure is invoked by calling the INITIALIZATION function that provide the following information:
6. Controlling picture changes

The state of the display surface is primarily affected by the state of the workstation tables and the CSS. The application can control the degree to which the display surface reflects the actual state of the CSS and workstation tables. The application can control the timing of display modification to gain certain visual effects and use the capabilities of a workstation more efficiently. Display changes can be delayed. Display changes can also be simulated. These simulations may exist for only a certain period of time and, during this time, the relationship between the state of the display and the CSS or workstation tables will be workstation dependent.

Two attributes are defined for this purpose, deferral mode and modification mode. Deferral mode governs when visual effects take place, while modification mode governs how visual effects take place. Deferral mode controls the possible delaying of functions which have visual effect: for example, data sent to a device may be buffered to optimize data transfer. The values of deferral mode are:

- ASAP: The display on the workstation becomes visually correct as soon as possible (ASAP).
- BNIG: The display on the workstation becomes visually correct before the next interaction globally (BNIG), i.e., before the next interaction with a logical input device gets underway on any workstation.
- BNIL: The display on the workstation becomes visually correct before the next interaction locally (BNIL), i.e., before the next interaction with a logical input device gets underway on that workstation.
- ASTI: The display on the workstation becomes visually correct at some time (ASTI).

- WAIT: Becomes visually correct when the Application requests it (WAIT) (e.g., by changing deferral mode).

The action which workstations perform to make display changes corresponding to changes in the CSS and workstation tables are extremely varied. Certain functions can be performed immediately on some workstations, but on other workstations imply a regeneration of the whole picture to achieve their effect. The entries "Dynamic modification accepted" in the workstation description table indicate which changes:

- lead to an implicit regeneration (IRG);
- can be performed immediately (IW);
- can be simulated immediately when in UQUM modification mode (CS).

An implicit regeneration is equivalent to an invocation of the function REDRAW ALL STRUCTURES. Its possible delay is controlled by the modification mode, a single entry in the workstation state list. The values of modification mode are:

- NIVE: No Immediate Visual Effect mandated. When modification mode is set to NIVE, the only changes to the display are those which would be done in accordance with the deferral mode.
- UWOR: Update Without Regeneration. All updates that can be realized immediately, without regeneration, are performed when the modification mode is set to UWOR.
- UQUM: Use Quick Update Methods. The effects of functions are to be simulated using workstation dependent quick update methods.

7. 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, for example, parameter and name assignment.

The layer model represented in Figure 9 illustrates the role of PHIGS in a graphics system. 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.
PEX is an X Window System protocol extension for 3D graphics. Just as Xlib generates X protocol, PHIGS in the X environment generates PEX protocol. Figure 10 shows the relationship between a graphics application, PHIGS, PEX, Xlib, and the X server. A single application can use both PHIGS and Xlib. Calls to Xlib functions cause transmission of X protocol to the server, while calls to PEX functions cause transmission of PEX protocol. The X server dispatches PEX requests to the server's PEX extension. The PHIGS library uses Xlib too, mainly to manage X resources and control the flow of information to and from the server.

Fig. 10 PHIGS in the Environment - PEX

8 Graphics system PHIGS +

PHIGS does not support a shading of pictures and modelling 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 in WSDT and the application program may set them. The predefined light source types are ambient, direc-

tional, positional, and spot light source. Each light source may be activated or deactivated.

A shading method is an attribute of the graphics primitives. Predefined shading methods are:
- NONE - no shading,
- COLOUR - colour interpolation shading,
- NORMAL - normal interpolation shading,
- DOT_PRODUCT - dot product interpolation shading.

Attribute reflectance equation specifies which components of the light will be 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 colour and shading data,
POLYLINE SET3 WITH DATA - creates a 3D polyline set structure element that includes colour and shading data,
QUADRILATERAL MESH3 WITH DATA - creates a 3D quadrilateral mesh primitive with colour and shading data,
TRIANGLE STRIP3 WITH DATA - creates a 3D triangle strip primitive with colour and shading data,
NON UNIFORM B-SPLINE CURVE NURBS - creates a structure element containing the definition of a non-uniform B-spline curve,
NON UNIFORM B-SPLINE SURFACE NURBS - creates a structure element containing the definition of a non-uniform B-spline surface,

9 Conclusion

PHIGS benefits application and application programmers. Many tasks previously performed within application programs are now handled by the PHIGS graphics support system. There are a good reason to use PHIGS:

Portability of programs

PHIGS is computer and device independent graphics system. The application program that utilize PHIGS can be easily transported between host processors, graphics devices and PHIGS implementations.
Sophisticated capabilities save development and maintenance time
PHIGS manages the storage and display of 2D and 3D graphical data,
creates and maintains a hierarchical database. Also, it is a true
3D system: PHIGS generates 2D drawings of 3D models in any view.
A commercial package from a long-standing vendor also guarantees the
availability of future enhancements, additional programming tools
to facilitate system use, and ongoing development of intelligent
workstation interfaces.

Increased program performance because of fewer error conditions
In support of the desired interactive, distributed environment,
PHIGS is designed to be a system with few error states. This helps
reduce I/O traffic between nodes of a distributed system. Further,
the production use of applications, which have well-defined inputs
and outputs, can be streamlined by minimizing error logging over-
head and utilizing run-time error recovery techniques. PHIGS provid-
es these capabilities.

On the opposite side we must say that, like as the most of
universal systems, PHIGS is robust system. PHIGS library has about
400 functions and it is rather difficult use it for beginners.

BIBLIOGRAPHY

[1] Information processing systems - Computer Graphics - Program-
for the storage and transfer of picture description information
Prague, 1993 in Czech.
[5] Hudec, B.: Grafics Systems PHIGS and GKS, textbook, CTU,Prague,
in Czech.

Démonstration Constructive Du Théorème de Pohlke-Schwarz

Svatopluk Zachariáš
katedra matematiky Západočeské univerzity
Americká 42
306 14, PLZEN
République Tchèque

Quand j’ai vu ce que tain de grands hommes,
En France, en Angleterre et en Allemagne,
ont écrit avant moi, j’ai été dans l’admiration,
mais je n’ai point perdu le courage.
Montesquieu, L’esprit des lois.

Le théorème de Pohlke-Schwarz dit:
Soient \((0,z_1,z_2)\) un simplexe et \(z_3\) un nombre dans un plan complexe \(C\).
Soit \((0,e_1,e_2,e_3)\) un simplexe dans \(E_3\); alors existent un nombre positif \(t\) et une projection
parallèle \(P: E_3 \rightarrow C\) tels que

\[
P \left( t e_1 \right) = z_1, \quad i = 1,2,3.
\]

Il existe, en général, deux projections obliques qui coïncident si, et seulement si, la projection
est orthogonale.

Cette démonstration vaut pour n’importe quel repère orthonormal en \(E_3\). La projection
parallèle \(P, P(e_i) = t \cdot z_i, i = 1,2,3,\) sera déterminée par un vecteur \(S\) non-nul satisfaisant à
la condition \(P(S) = 0\). On en conclut que si \(S = e_1 \cdot e_2 + e_2 \cdot e_3 + e_3 \cdot e_1\), on aura
\(0 = e_1 \cdot z_1 + e_2 \cdot z_2 + e_3 \cdot z_3\).

Pour satisfaire à cette condition il suffit de prendre

\[
S = \text{real}(z) \times \text{imag}(z).
\]

En désignant

\[
e_i = (e_1, e_2, e_3)^T, \quad i = 1,2,3,
\]
on a

\[
S_i = e_1 \cdot e_2 + e_2 \cdot e_3 + e_3 \cdot e_1, \quad i = 1,2,3.
\]

Le nombre \(|e_3|\) est égal au double de l’aire non-nul du simplexe \((0,z_1,z_2)\). Maintenant nous pouvons remplacer le vecteur \(s\) par un vecteur unitaire

\[
s = \frac{S}{\sqrt{S_1^2 + S_2^2 + S_3^2}}.
\]