[ISO646] ISO 646 - 7-bit Coded Character Set for Information Interchange, ISO 1983 [IS2022] ISO 2022 - ISO 7-bit and 8-bit Coded Character Sets - Code Extension Techniques, ISO 1983
[IS7942] ISO 7942 [1985] - Information Processing - Computer Graphics - Graphical Kernel System [GKS] - Functional Description
[IS8632] ISO/IEC 8632: Information Processing Systems - Computer Graphics Metafile for the Storage and Transfer of Picture Description Information, ISO 1992 [IS8632A] ISO/IEC 8632-1:1987/Am.1:1990 : Information Processing Systems - Computer Graphics Metafile for the Storage and Transfer of Picture Description Information - Amendment 1 - part 1, ISO 1990
[IS8632B] ISO/IEC 8632-1:1987/DAM 3 Information Processing Systems - Computer Graphics Metafile for the Storage and Transfer of Picture Description Information - Amendment 3 - part 1, ISO 1987
[IS8632C] ISO/IEC 8632-1:1987 : Information Processing Systems - Computer Graphics Metafile (CGM) for the Storage and Transfer of Picture Description Information, ISO 1987
[IS8651] ISO 8651 - GKS Language Bindings, ISO 1988
[IS8805] ISO/DIS 8805 - Graphical Kernel System for Three Dimensions (GKS-3D), ISO 1987
[IS9592] ISO 9592 - Programmer's Hierarchical Interactive Graphics System (PHIGS), ISO 1989
[IS9636] ISO 9636 - Computer Graphics Interface (CGI), ISO 1991
[KrčJ91] KRČ-JEDINÝ, J.: ISO/IEC JTC1/SC24 (Computer Graphics) Plenary London, Apr 22-23 1991 - Results and Trends, str. 90-95, JŠPG, Bratislava 1991
[KrčJ92] KRČ-JEDINÝ, J.: ISO/IEC JTC1/SC24 Plenary Report, Amsterdam, Feb 92, str. 84-88, JŠPG, Bratislava 1992
[KrMe87] KRČ-JEDINÝ, J. - MEDERLY, P.: Vývoj medzinárodných noriem v počítačovej grafike, JŠPG, Richňava 1987
[KrMe88] KRČ-JEDINÝ, J. - MEDERLY, P.: Medzinárodné normy v počítačovej grafike, in: Objektovo orientované programovanie a modelovanie systémov, str. 98-109, Alfa Bratislava 1988
[Mede88] MEDERLY, P. - ERTL, J. - FERKO, A. - GAŠPAR, D. - KRČ-JEDINÝ, J.: Seminár Grafické systémy, skriptá ČSVTS, Bratislava 1988
[MSWi90] Microsoft Windows, User's Guide v. 3.0, 1990
[MuSk88] MUMFORD, A. M. - SKALL, M. W. (Eds.): CGM in the Real World, Springer-Verlag 1988
[Pavl82] PAVLIDIS, T.: Algorithms for Graphics and Image Processing, Springer Verlag 1982
[PEX390] PEX Introduction and Overview, PEX Version 3.20, str. 1-14, 1990 [Purg91] Grafische Datenverarbeitung, skriptum, TU Wien 1991
[Roge85] ROGERS, D.F., ROGERS, S.D.: A Raster Display Graphics Package for Education in Computer Graphics '85, Tokio 1985
[Rost89] ROST, R. - FRIEDBERG, J. - NISHIMOTO, P.: PEX: A Network-Transparent 3D Graphics System, IEEE CG and Applications 9 (4), 14-26, 1989
[RuFe89] RUŽICKÝ, E. - FERKO, A.: The Demonstration Programs for Teaching of Computer Graphics, zborník konferencie Computer Graphics, Smolenice 1989
[Ruži91] RUŽICKÝ, E.: Ortopolygons and Windows System, str. 101-106, JŠPG, Bratislava 1991
[Ruži91] RUŽICKÝ, E.: Úvod do počítačovej grafiky, MFF UK Bratislava 1991 [Sama85] SAMARSKIJ, A.: Současná aplikovaná matematika a počítačové modelování, Pokroky matematiky, fyziky a astronomie č. 1/85, Praha 1985
[ScGe86] SCHEIFLER, R. - GETTYS, J.: The X Window System, ACM Transactions on Graphics 5 (2), 79-109, 1986
[Skal90] SKALA, V.: Počítačová grafika, skriptum VŠSE, Plzeň 1990
[Skal92] SKALA, V.: Algoritmy počítačové grafiky I, II, III, VŠSE Plzeň 1992
[SoŽá93] SOCHOR, J. - ŽÁRA, J.: Algoritmy počítačové grafiky, skriptum ČVUT, Praha 1993
[Stuc91] STUCKI, P.: Graphics and Multimedia, tutorial No. 10, Eurographics Conference, Vienna 1991
[Sung90] SUNG, H. C. K. - ROGERS, G. - KUBITZ, W.: A Critical Evaluation of PEX, IEEE CG and Applications, Vol 10, No 6, str. 65-75, Nov 1990
[Thom90] THOMAS, S.W.: X and PEX Programming, Tutorial Note 3, Eurographics'90 Zurich 1990
[Žára92] ŽÁRA, J. a kol.: Počítačová grafika, principy a algoritmy, Grada, Praha 1992

GRAPHIGS SYSTEMS PHIGS and PHIGS+

Bohuslav Hudec, department of computers,
Electrotechnical faculty of Czech Technical University, Praque
e-mail hudec@cs.felk.cvut.cz

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

The PHIGS (Programmer s Hierarchical Interactive Graphics System) is an interantional ISO standard of a 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. PHICS concept and graphical workstation.

PHIGS is a high-level graphics library with over 400 functions. It allowes an application programmer to describe a model of a scene, to display the model on workstation, to manipulate and to edit the model interactivly. The models are stored in a graphical database, known as <u>centralized structure store</u> (CSS). The fundamental entity of data is a <u>structure element</u> and these are grouped together into units called <u>structures</u>. Structures are organized as acyclic directed graphs called <u>structure networks</u>.

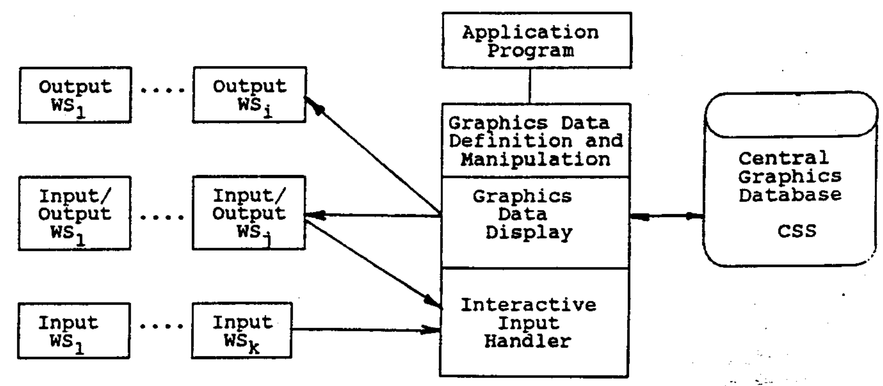

Fig. 1 Structure of the PHIGS

The two abstract concepts (abstract ouptup and abstract input) are the building bloks 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 - suports output only,
INPUT - suports input only,
OUTIN - suports output and input,
MO - suports output graphical and applicatin data to the external storage,
MI - suports 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 ie 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 spcific workstation description table may change at any time while the workstation is open. The application prodgram 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 struture 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 :

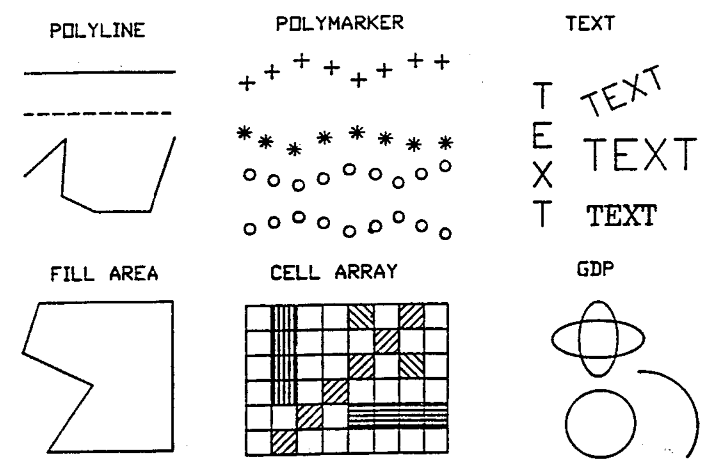

Fig. 2 Examples of output primitives

POLYLINE set of connected lines defined by a point sequens
POLYLINE3
POLYMARKER set of symbols of one type centred at given position
POLYMARKER 3
TEXT character string at a given position on an arbitrary
TEXT 3 plane in modelling coordinate space (see below)
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, eliptic 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-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 (or entry in the bundle table) may be different for different workstations.

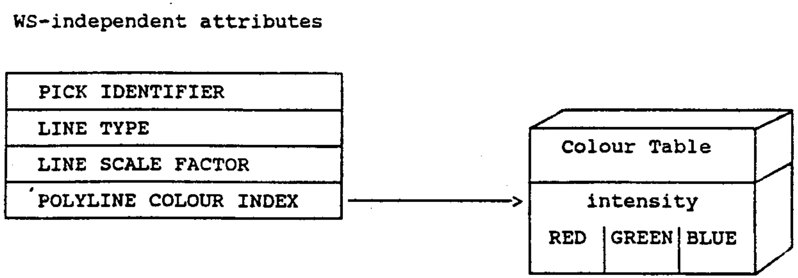

Fig. 3 Individual specification of attributes

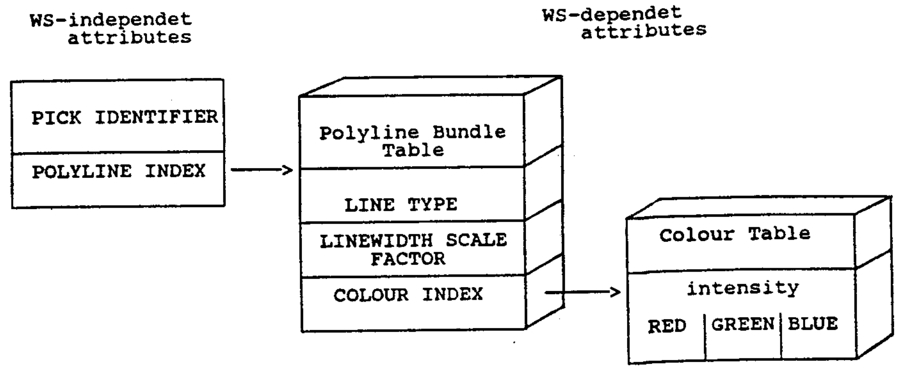

Fig. 4 Bundled specification of attributes

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 excample, 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.

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 is specified by the application.

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

Tab.1 List of attributes of graphics elements

Structure networks are oranized 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 recursivly (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 (POSTE 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)

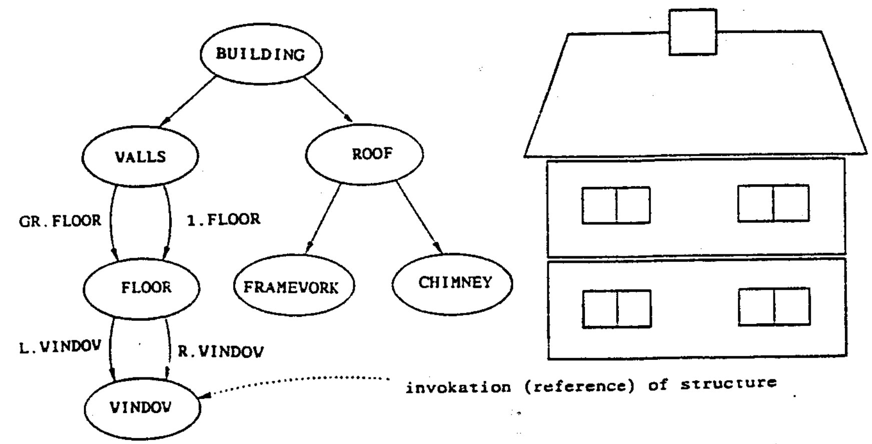

Fig. 5 Building and its structure network

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.

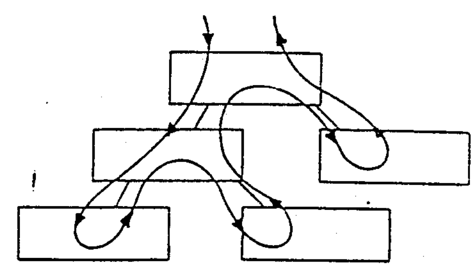

Fig. 6 Traversal process

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 <u>traversal state list</u> (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 acces and modify each element within a structure. Editing functins are available 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 by the OPEN STRUCTURE function. When a structure is opened for editing an <u>element pointer</u> is established whitch poits at the last element in the structure. The element poiter may be positioned using the function :

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

When a structure is open the applicatin program may insert additional elements into the structure or replace existing structure elements using the functins which define structure elements (e.g. POLYLINE, SET POLYLINE INDEX, ...). The <u>edit mode</u> 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 spetial 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 functin INQUIRE CURRENT ELEMENT TYPE AND SIZE returns the type of the element and its size, the function INQUIRE CURRENT ELEMENT CONTENT reterns the parametrs 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 ancesters,
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 fonctions for CSS:

INQUIRE PATHS TO ANCESTORS - returns path which reference the specified structure,
NQUIRE PATHS TO DESCENDANTS - returnspath which are refered 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 <u>archive file</u> (AF) is a medium for storing structure definitions. PHIGS funcionality provides for archiving from the CSS to an

AF, retrieving from an AF to the CSS, or deleteing from an AF. This set of functins may be applied to a list of structure networks, or to all structures defined either in CSS or in an AF. There are these function:

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,
RETRIVE STRUCTURES, RETRIVE STRUCTURE NETWORKS,
RETRIVE 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,
RETRIVE ANCESTORS OF STRUCTURE, RETRIVE DESCENDANTS OF STRUCTURE returns path which reference the specified structure resp. returns path which are refered by a particular structure.

4 Coordinate systems and transformations

Geometrical informatins are usualy coordinates of points. We supose that structures respresent parts of a hierarchical model of modelling scene. Each of these parts has own world space represented by <u>modelling coordinate system</u> (MC). The relative positioning of the separate parts is achieved by having a single <u>Word Coordinate Space</u> (WC) onto which all the defined modelling coordinate systems are mapped by a <u>composite modelling transformaton</u> during the traversal process. The WC space can be regarded as a workstation idipendent abstract viewing space.

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

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

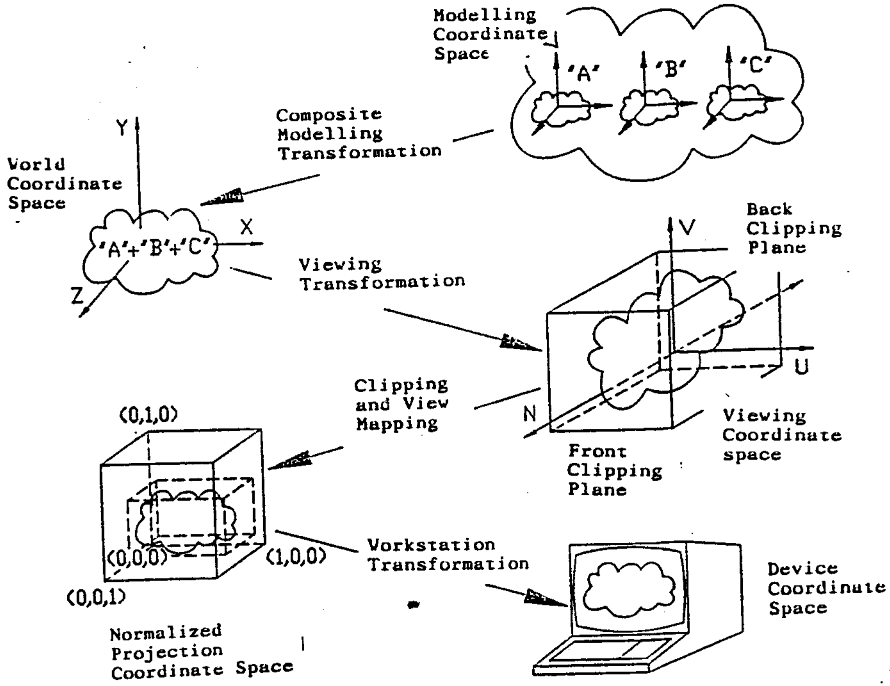

Fig. 7 Coordinate systems and transformations in PHIGS

Output primitives and attributes are mapped from WC to VRC by the <u>view orintation transformation</u>, from VRC to NPC by the <u>view mapping transformation</u> and finally from NPC to DC by the <u>worstation transformation</u>. 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.

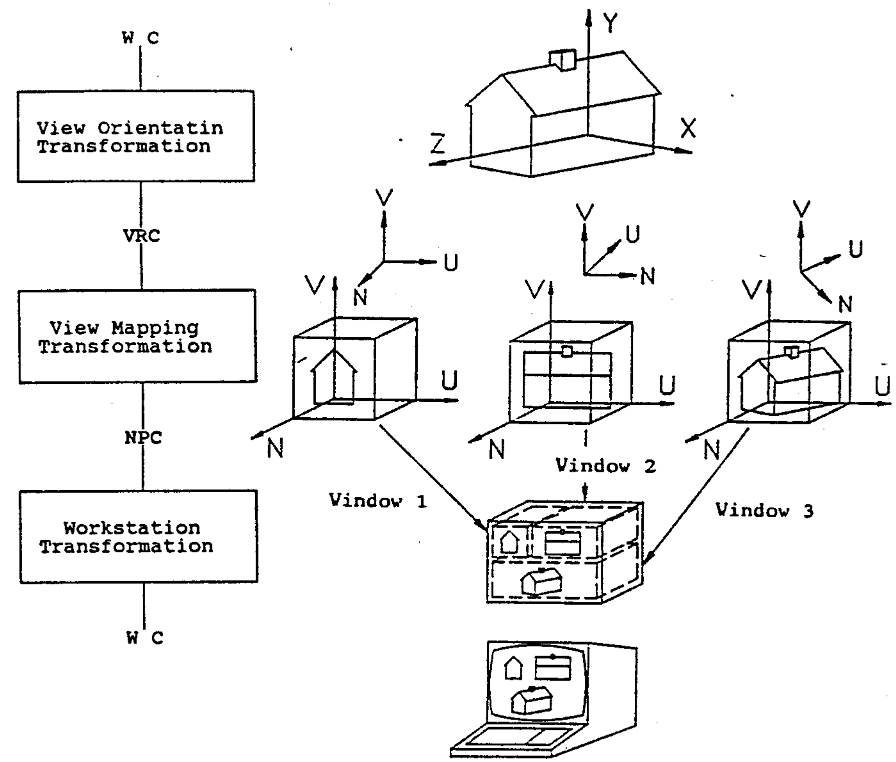

Fig.8 Viewing transformations in PHIGS

5 Graphical input

An applications program obtains graphical input from an operator by controlling the activity of one or more logical input devices (LID) , which deliver logical input values to the program. LID is identified by a workstation identifier, an <u>input class</u> and a device number. The input class determines type of logical input value that the LID delivers. The six input classes and the logical input values they provide are:

LOCATOR : a position in world coordinates and a index of view transformation,

STROKE : a sequence of points in world coordinates and index of view transformation,
VALUATOR: a real numbre,
CHOICE : a CHOICE status and positive integer that represents a selection from a numbre 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 <u>operating modes</u>. 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.

EVEN : 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 <u>promt</u> and they can terminate by a <u>echo</u>. For that reason it is posible to set certain parameters associated with LID of each input class. This initialization procedure is invocated by calling the INITIALIZATION function that provide the following information:

* An initial value aoppropriate to the class (e.g. locator position and index initial view transformation).
* A promt and echo type that selects the promting technique and, if echoing is on, the echoin technique for a LID. An implemanted dependet promt and echo type (type 1) is required for all LID. Further promt and echo types appropriate to each class are defined but not required.
* An echo area in device coordinate. Input device implementation may use the echo to display promts or echoes.
* A data record. Some input classes have mandatory control values in the data record.

6. Controlling picture changes

The state of the display surface is primarily affected by the state of the worsktation 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 to 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 fot this purpose, <u>deferral mode</u> and <u>modification mode</u>. 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 "<u>dynamic modification accepted</u>" in the workstation description table indicate which changes

* lead to an implicit regeneration (IRG);
* can be performed immediately (IMM);
* can be simulated immediately when in UQUM modification mode (CBS).

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 metods.

7. Language interfafaces

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 languagage dependent layer, other application dependent layers, and operations system resources. There are standards of the language dependet layers for the language FORTRAN, PASCAL and C.

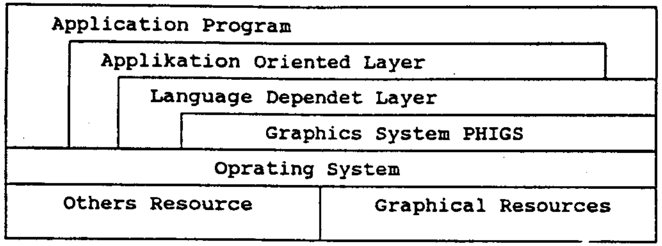

Fig. 9. Layer model of PHIGS

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 Ylib. Calls to Xlib functions cause transmission of X protocol to the server, while calls to PHIGS 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.

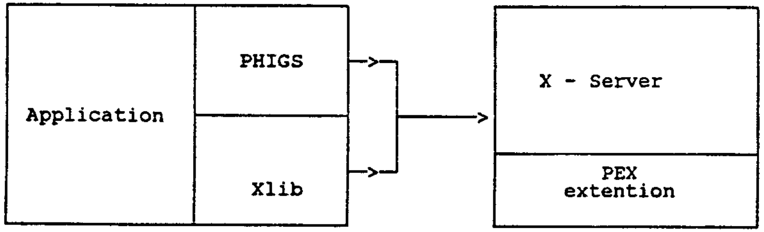

Fig. 10 PHIGS in the Environment - PEX

8 Graphics system PHIGS +

PHIGS dos not suport a shading of pictures and modelling a Non Uniform Rational B-Slinne curves end surfaces (NURBS). An extension of PHIGS for these functionalities is PHIGS+.

PHIGS+ anables to specify light sourcec. Parametrs of the light sources are described in WSDT and the application program may set up 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.

Attribut reflectance equotion specifies, whitch components of the light will be acsepted 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 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:

<u>Poratability of programs</u>
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 comercial 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.

Icreased 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. Futher, the production use of applications, which have well-defined inputs and outputs, can be streamlined by minimizing error logging overhead and utilizing run-time error recovery techniques. PHIGS provides these capabilities.

On the oposite 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 beginers.

BIBLIOGRAPHY

[1] Information processing systems - Computer Graphics - Programmer s Hierarchical Interactive Graphics System PHIGS, ISO 1989.

[2] Information processing systems - Computer Graphics - Metafile for the storage and transfer of picture description information ISO/ISO 8632-/1-4,1987.

[3] D.B.Arnolg : P.R.Bono : CGM and CGI. Springer 1988.

[4] Hudec, B.: Fundamets of Computer Graphics, textbook, CTU Praque, 1993 in Czech.

[5] Hudec, B.:Grafics Systems PHIGS and GKS, textbook, CTU,Praque, in Czech.

[6] L. Piegl : On NURBS A Survey, IEEE Computer Graphics & Applications, pages 55-71, January 1991.

[7] W. Tiller and L. Piegl. Curve and Surface Constructions using Rational B-splines.Computer Aided Design, 19(9) : 485-498,1987.

[8] Blake, J.W. : PHIGS and PHIGS+ An introduction to 3D Computer Graphics. Academic Press, 1992.

[9] Gaskins,T.:PHIGS Programming Manual, O Reilly & Associates Inc.

Démonstration Constructive du Théorème de Pohlke-Schwarz

Svatopluk Zachariáš

katedra matematiky Západočeské univerzity

Americká 42

306 14, PLZEŇ

République Tchèque

Quand j'ai vu ce que tant 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 $\mathbf{C}$. Soit $(\mathbf{0}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ un simplexe dans $\mathbf{E}_3$; alors existent un nombre positif t et une projection parallèle P: $\mathbf{E}_3 \mapsto \mathbf{C}$ tels que

$$P\left(\frac{\mathbf{e}_i}{t}\right) = z_i, \quad i = 1, 2, 3.$$

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 $\mathbf{E}_3$. La projection parallèle P, $P(\mathbf{e}_i) = t \cdot z_i$, $i = 1, 2, 3$, sera déterminée par un vecteur $\mathbf{S}$ non-nul satisfaisant à la condition $P(\mathbf{S}) = 0$. On en conclut que si $\mathbf{S} = \sigma_1 \cdot \mathbf{e}_1 + \sigma_2 \cdot \mathbf{e}_2 + \sigma_3 \cdot \mathbf{e}_3$, on aura $0 = \sigma_1 \cdot z_1 + \sigma_2 \cdot z_2 + \sigma_3 \cdot z_3$.

Pour satisfaire à cette condition il suffit de prendre

$$\mathbf{S} = \mathrm{real}(\mathbf{z}) \times \mathrm{imag}(\mathbf{z}).$$

En désignant

$$\mathbf{e}_i = (e_{1i}, e_{2i}, e_{3i})^T, \quad i = 1, 2, 3,$$

on a

$$S_i = \sigma_1 \cdot e_{i1} + \sigma_2 \cdot e_{i2} + \sigma_3 \cdot e_{i3}, \quad i = 1, 2, 3.$$

Le nombre $|\sigma_3|$ 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

$$\mathbf{s} = \frac{\mathbf{S}}{\sqrt{S_1^2 + S_2^2 + S_3^2}}.$$