PREMO를 기반으로 하는 그래픽스 객체 타입 분석 및 바인딩 모델 설계

이 영 철[†] · 김 민 홍[††] · 김 하 진[†††]

요 약

국제 표준 전문 위원회에서는 PREMO(PResentation Environment for Multimedia Object)의 언어 바인딩에서 제기된 문제점을 해결하기 위하여 언어바인딩의 표준화를 취소하고 어떤 특정 언어에 의존하지 않는 바인딩 모델의 연구가 진행중이다. PREMO의 바인딩 모델에서 그래픽스 바인딩 모델을 제시하기 위하여 PREMO 에서의 객체 함수들을 분석하고, 객체-Z 기능명세에 대한 함수 작용과 바인딩 모델에 관련된 데이터 사상을 보인다. 본 논문에서는 그래픽스 언어 바인딩 함수들을 기반으로 하는 그래픽스 바인딩 모델 설계를 제안한다.

A Binding Model Design and Graphics Object Types Analysis Based on the PREMO

Young-Chul Lee[†] · Min-Hong Kim[††] · Ha-Jine Kim[†††]

ABSTRACT

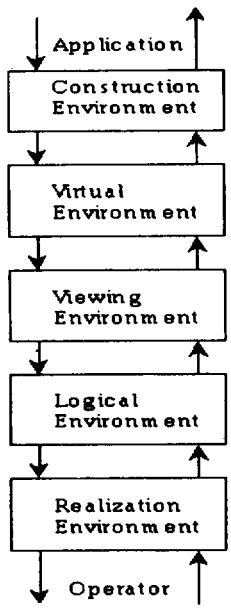
The ISO/IEC JTC1/SC24/WG6 is studying of a binding model because of the problems from language bindings [1]. That had cancelled the language binding standardization and this model is not depend on any specific language. To suggest the graphics binding model, we analysis the object functions on the PREMO. We see the data mapping relate to the binding model and descripted function operation for the object-Z functional specification. In this paper, We have proposed a graphics binding model design based on the graphics language binding functions of PREMO.

1. Introduction

In spite of the development of computer graphics technology, existing standards(GKS, PHIGS), however, have little chance of providing appropriate answers to rapidly changing today's computer graphics technology. Therefore, ISO/IEC JTC1 SC24, responsible for the development and maintenance of computer graphics standards recognized the need to develop a new computer graphics standard radically.

To this end, the new work named PREGO(PResentation Environment for Graphical Objects)was proposed. But it was renamed PREMO during the process of the Chiemsee meeting to reflect the applicability to not only graphical objects but also multimedia objects as well. A new project was launched at an SC24 meet-

† 정 회 원:아주대학교 대학원 컴퓨터공학과
†† 정 회 원:경기대학교 전자계산학과 교수
††† 정 회 원:아주대학교 정보통신대학 컴퓨터공학과 교수
논문접수:1997년 10월 8일, 심사완료:1997년 12월 31일



(Fig. 1) Environments in MRI component

ing in Chiemsee, Germany in October 1992, subsequent meetings resulted in a draft for a new standard called PREMO[3].

The major features of the PREMO can be briefly summarized as follows[4][5][6].

- PREMO is a Presentation Environment.
- PREMO aims at a Multimedia Presentation.
- PREMO is Object Oriented.

The computer graphics environment means the MRI (Modelling Rendering Interaction) environment that the environment model is called CGRM(Computer Graphics Reference Model)[7][8][9]. The main purpose of the CGRM is to define concepts that shall be used to develop computer graphics standards. Additional purposes are to explain relations between SC24 standards and to provide a from whereby areas outside computer graphics can identify their relationships to computer graphics. This model may be applied to:

- verify and refine requirement for computer graphics;
- identify needs for computer graphics standards and external interfaces;
- develop models based on requirements for computer graphics;
- define the architecture of new computer graphics standards;
- compare computer graphics standards.

2. Graphics Binding Model Design

The main area of the international standards for computer graphics contains GKS(Graphical Kernel System) and PHIGS(Programer's Hierarchical Interactive Graphics System). The main reasons for standardizing basic computer graphics are:

- to allow application programs involving graphics to be easily portable between different installations;
- to aid the understanding and use of graphics methods by application programmers.
- to serve manufacture of graphics equipment as a guideline in providing useful combinations of graphics capabilities in a device.
- to reduce program development costs and time; many of the functions currently performed by the application program will now be performed by standards.

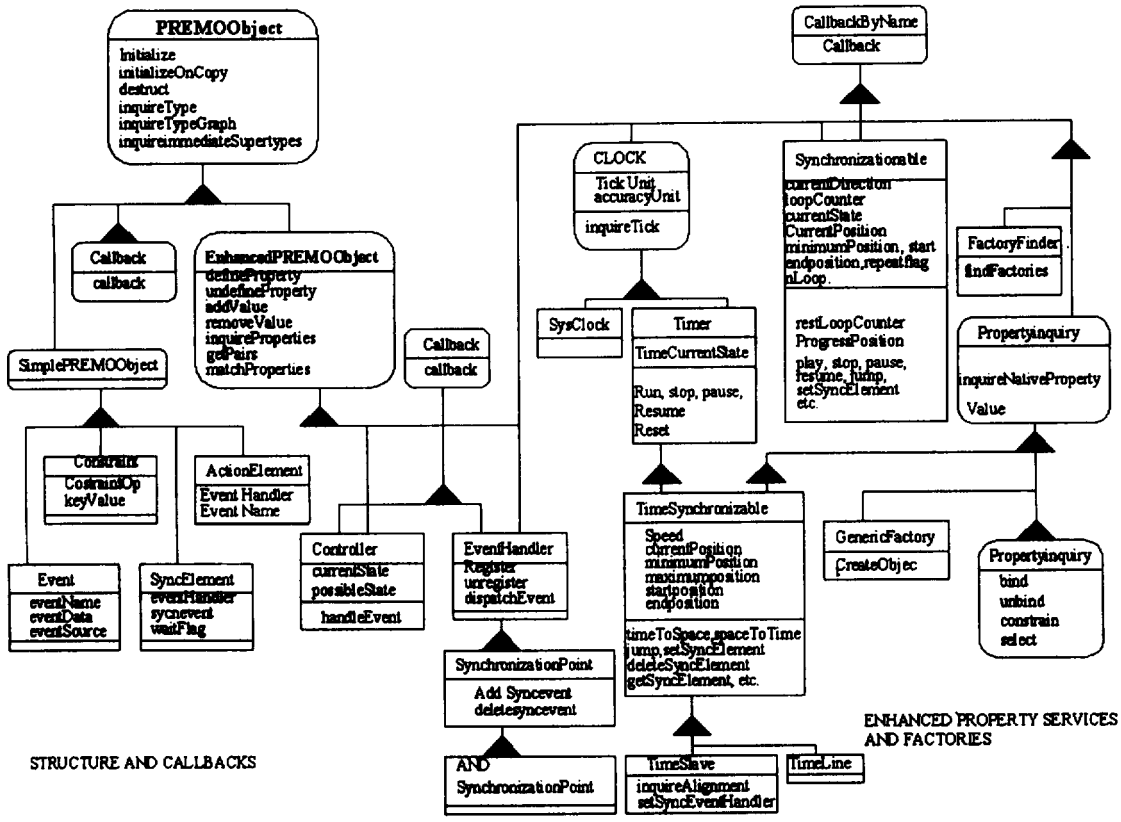
2.1 Analysis of the Object types

The PREMO objects consist of the structures which are callbacks and controller and event handler objects and event handlers time and synchronization object and enhanced property services and factories(Fig. 2).

The PREMO uses an object model to support design portability and reuse of object definitions. The use of an object-oriented design leads to natural description and provides, in particular, a way for explaining PREMO's extensibility and configuration aspects [5][9].

For the configuration of this object model following definitions and the description of object types are used.

The objects support only certain operations. The object type defines these operations, and thus charac-



(Fig. 2) PREMO foundation object types

terizes the behaviour of object. Objects are the PREMO Object Model, objects shall not change their type. Types are arranged into a type hierarchy that forms a directed, acyclic graph. Types can be related to one another through supertype/subtype relationship, and an abstract type in PREMO can be providing. PREMO objects inherit from type ***PREMOObject***(Fig. 2).

2.1.1 Fundamental Object Behaviour

The PREMO objects consist of many objects. They are covered by the specification of a so-called ***PREMO-Object*** object type, which is the common supertype for all PREMO object types. By virtue of subtyping, all operations defined on this object type are available for all other PREMO Objects[3][4][5][9].

- **Enhanced Data Objects**: The semantics associated with a data object define the construction and modification interface of a particular data object. Examples are geometric 2D or 3D points, colour matrices, with related operations and other attributes, video frames, frequency spectra, etc.

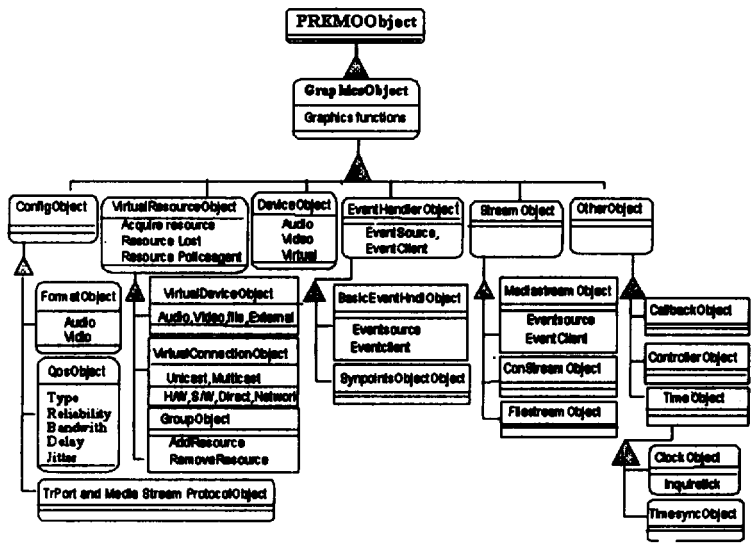
- **Event Handler Objects**: This object type offers the necessary operations for the implementation of event management within PREMO. Event management plays a fundamental role within interactive systems in general, and the "Event Model" of PREMO aims at providing this basic mechanism. The essential feature of the event model is the separation between the source of the events, and the recipient of these events. Sources broadcast the events without having any knowl-

edge of which objects would receive them; this is done by forwarding the event instance to special PREMO EventHandler objects. Prospective recipients of events register with these EventHandler object, placing a request based on the event type and optionally other more complex constraint specifications. The recipients are then notified by the EventHandler on the arrival of an event, together with information on the source of the event. This simple mechanism constitutes one of the main building blocks for the creation of more complex interaction patterns in the PREMO.

- Controller Objects: The role of a Controller is to coordinate cooperation among objects. A controller object is an autonomous and programmable finite state machine(FSM). correspond to messages sent to other objects. The actions of a Controller object may cause messages to be sent to other Controller objects, thus a hierarchy of Controllers can be defined.

- Clock Objects: These objects provide a unified interface to the system's view of real-time clock. The clock object type assumes the existence of two non-object types: **Time**, to measure elapsed ticks(realized, for example, as a 64 bits integer), and **TimeUnit**, which (as an enumerated type) defines the unit represented by each clock tick, for example an hour or a micro-second. Specifically, the clock object type supports an operation, **inquireTick**, that returns the number of ticks that have occurred since the start of era defined for PREMO. However, the accuracy in various units with which particular PREMO implementation can describe the elapsed duration since the start of era will vary, performance of the local object.

- Synchronization Objects: The synchronization facilities included in PREMO are based on an event-based synchronization model[10]. In this model, each synchronizable object is considered to progress autonomously along an internal, one dimensional coordinate space, which may be integer, real, or time-based (e.g., an integer coordinate may be used to describe frames in a video sequence) may have a set of reference points(e.g., video frames). Points on this space are referred to as reference points; for each reference point, an object and an operation can be registered, together with an event instance(the synchronization event). This operation is invoked by the synchronizable object when a reference point is crossed, using



(Fig. 3) Graphics object model

the event stored at the reference point as an argument. A reference point may also contain a boolean flag, which may instruct the synchronizable object to suspend itself and wait for an external message request to continue its progress. Typically, a reference point would refer to an event handler object or a controller object. The event based synchronization model can be used for synchronization patterns where traditional, purely time-based synchronization is not sophisticated enough. A purely time based synchronization can also be built on top of this model, and this is done by combining(through multiple inheritance) the purely event based synchronization objects with the clock object.

2.1.2 Graphics language bindings

The binding functions for providing graphics binding model based on standard language binding(GKS, GKS-3D, PHIGS, PHIGS PLUS), that used following standards in 〈Table 1〉 [11][12][13][14]:

〈Table 1〉 Graphics binding documents

Language Binding Documents	ISO Document No
GKS Fortran language bindings	IS 8651-1
GKS Pascal language bindings	IS 8651-2
GKS Ada language bindings	IS 8651-3
GKS C language bindings	IS-8651-4
GKS-3D C language bindings	IS-8806-4
GKS-94 Fortran90 language bindings	WD 8651-5
PHIGS Fortran language bindings	IS 9593-1
PHIGS Ada language bindings	IS 9593-2
PHIGS C language bindings	IS 9593-3

All list of standard function names depend on the documents(GKS, PHIGS) that function name for graphics bindings are:

- *Control functions*
- *Output primitive functions*
- *Attribute specification functions*
- *Transformation and clipping functions*
- *Structure content functions*
- *Structure manipulation functions*
- *Structure display functions*
- *Structure archive functions*
- *Input functions*
- *Metafile functions*

2.2 Formal specification for bindings

The importance of formalizing the specification of standards has been recognized for years. In this paper, we advocate the use of the formal specification language Object-Z in the definition of standards. ***Object-Z*** [2] is an extension to the Z language specification to facilitate in an object oriented style.

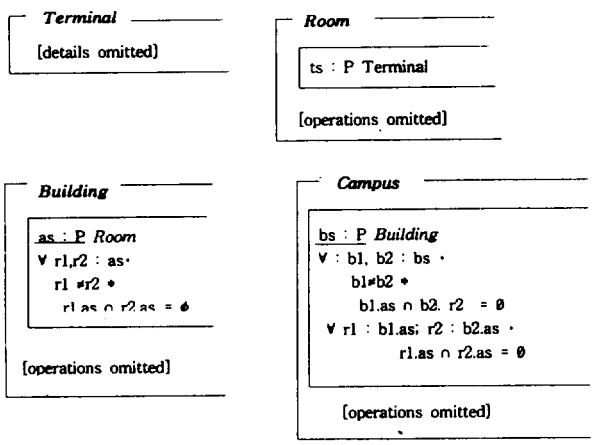
The Z specification defines a number of state and operation schemas. Inferring which operations may affect a particular state schema requires examining the signatures of every operation. In contrast, Object-Z associates individual operations with one state schema. The collective definition of a state schema with its associated operations constitutes the definition of a *class*. The *class* is template for objects: each object of the class has a state which conforms to the class' state schema and is subject to state transitions which conform to the class' operations. A class is also used as a type: instances of that type are identities which reference objects of that class. This enables objects to refer to other objects. An Object-Z specification of a system comprises a number of class definitions possibly related by inheritance, a mechanism for class adaptation by modification or extension. For the more detailed description of the Z semantics can be referred to[2].

2.3 Containment functionality of object-Z

To the specify the binding functions, we should apply the containment functionality of Object-Z specification method. Roger Duke et al[2] have given the following example to explain containment.

Considering the situation where a campus contains a set of buildings, each building contains a set of rooms and each room contains a set of terminals. A

specification in Object-Z would be



The class invariant of class ***Building*** specifies that no terminal can be in two distinct rooms in a ***building***. Similarly, the class invariant of class ***Campus*** specifies that no ***room*** can be in two distinct ***buildings*** of the ***campus***. Furthermore, despite the fact that the predicate of class ***Building*** states that no terminal can be in two distinct ***rooms*** in distinct ***buildings***.

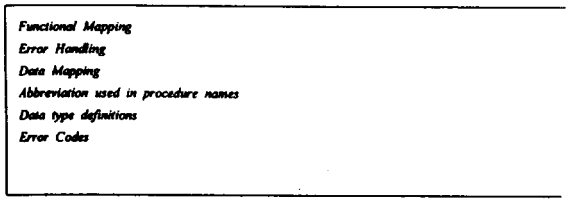
3. Functions for the binding model

The PREMO binding model is emerging standard developing area. Therefore, we do not compare with other graphics binding model. Existing graphics standards are descripted according to the specify imlementation languages(Fortran, C etc,.). In this paper, we are descripted the graphics functions based on the language binding. For the methodology of the binding functions not descripted detail. In the binding model descripted only language mapping functions of graphics functions, because of independ on the specific language.

As the explain above, we propose only a graphics binding model for PREMO, that is not depend on any specific language. 〈Table 2〉

〈Table 2〉 Graphics binding functions

GRAPHICS BINDING FUNCTIONS	
BINDING FUNCTIONS	
CONTROL FUNCTIONS	*Open, Close, Update, Redraw set display, Message, Activate, Create, Escape etc.*
OUTPUT PRIMITIVE FUNCTIONS	*Polyline, Polymarker Text, Annotation text relative, Fill area, Cell array, Generalized drawing primitive etc.*
OUTPUT ATTRIBUTE SPECIFICATION FUNCTIONS	*Polyline, Polymarker, Linetype, Marker, Text, Character, Fill area, Pattern, Aspect, Annotation Interior, Edge, Highlighting , Hlhar etc.*
TRANSFORMATION AND CLIPPING FUNCTIONS	*Window, Viewport, Local, Global, Modelling, Scale, Rotate, Transform, Evaluate etc.*
SEGMENT FUNCTIONS	*Create, Close, Delete, Rename, Associate, Copy, Insert, Visibility, Detectability etc.*
STRUCTURE CONTENT FUNCTIONS	*Open, Close, Execute, Label, Application, Edit, Element, Empty etc.*
INPUT FUNCTIONS	*Initialize, Locator, Choice, Locator, Request, Sample, Get, etc.*
STRUCTURE MANIPULATION FUNCTIONS	*Delete, Change, etc.*
STRUCTURE DISPLAY FUNCTIONS	*Post Structure, Unpost Structure etc.*
STRUCTURE ARCHIVE FUNCTIONS	*Open,. Close, Archive, Retrieve, Delete etc.*
METAFILE FUNCTIONS	*Write item, Get item, Read item, Interpret item etc.*
INQUIRY FUNCTIONS	*Operation state value, GKS state list, Workstation state list, Workstation description table, Segment state list, Pixel, Error state list.*
ERROR CONTROL FUNCTIONS	*Emergency close, Error handling, Error Logging,. Error handling Mode etc.*
SPECIAL INTERFACE FUNCTIONS	*Escape.*
APPLICATION FUNCTIONS	*Application function in GKS, Specific functions in binding etc.*



3.1 Function operations in the binding model

1) CONTROL FUNCTIONS: procedure OPEN_XXX(Graphics system or workstation name) (ERROR_FILE: in FILE ID: = system name STRING(DEFAULT ERROR FILE); AMOUNT_OF_MEMORY: in system_NATURAL: = DEFAULT_MEMORY_UNITS);

procedure CLOSE, REDRAW SET DISPLAY, MESSAGE, ACTIVATE, CREATE, ESCAPE etc,.

2) OUTPUT PRIMITIVE FUNCTIONS: POLYLINE
procedure (POINTS: in MCPOINT_LIST);

procedure POLYLINE, POLYMARKER, TEXT, ANNOTATION TEXT RELATIVE, FILLAREA, CELL ARRAY, GENERALIZED DRAWING PRIMITIVE etc,.

3) OUTPUT ATTRIBUTE SPECIFICATION FUNCTIONS: SET POLYLINE_INDEX
procedure (POLYLINE_IND: in POLYLINE_INDEX);

procedure SET_POLYMARKER_INDEX, LINETYPE, MARKER, TEXT, CHARACTER, FILL AREA, PATTERN, ASPECT, ANNOTATION, INTERIOR, EDGE, HIGHLIGHTING, HLHSR etc,.

4) TRANSFORMATION AND CLIPPING FUNCTIONS: SET LOCAL TRANSFORMATION
procedure SET_LOCAL_TRANSFORMATION (MATRIX: in TRANSFORMATION MATRIX_2; HOW_APPLIED: in COMPOSITION_TYPE);

procedure WINDOW, VIEWPORT, GLOBAL, MODELLING, SCALE, ROTATE, TRANSFORM, EVALUATE etc,.

5) SEGMENT FUNCTIONS: SET SEGMENT
procedure SET_SEGMENT

procedure CREATE, CLOSE, EXECUTE, LABEL, APPLICATION, EDIT, ELEMENT, EMPTY etc,.

6) STRUCTURE CONTENT FUNCTIONS: OPEN STRUCTURE
procedure OPEN STRUCTURE (STRUCTURE_IDENTIFIER: in STRUCTURE_ID);

procedure CLOSE, EXECUTE, LABEL, APPLICATION, EDIT, ELEMENT, EMPTY, etc,.

7) INPUT FUNCTIONS: SET PICK IDENTIFIER
procedure SET_PICK_IDENTIFIER (pick_identifier: in PICK_ID);

procedure SET_INITIALIZE, CHOICE, LOCATOR, REQUEST, SAMPLE, GET, etc,.

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);

8) STRUCTURE MANIPULATION FUNCTIONS: DELETE STRUCTURE
procedure DELETE STRUCTURE (STRUCTURE_IDENTIFIER: in STRUCTURE_ID);

procedure CHANGE, etc,.

9) STRUCTURE DISPLAY FUNCTIONS : POST STRUCTURE
procedure POST_STRUCTURE
(WS : in WS_ID;
STRUCTURE_IDENTIFIER : in STRUCTURE_ID;
PRIORITY : in WS_ID)

procedure POST STRUCTURE, UNPOST STRUCTURE etc,.

10) STRUCTURE ARCHIVE FUNCTIONS : OPEN ARCHIVE FILE
procedure_OPEN_ARCHIVE_FILE
(ARCHIVE_IDENTIFIER : in ARCHIVE_ID;
ARCHIVE_FILE : in FILE_ID);

procedure CLOSE ARCHIVE, RETRIEVE DELETE etc,.

11) METAFILE FUNCTIONS : WRITE ITEM TO METAFILE
procedure GET_CHOICE
(status : out CHOICE_STATUS;
(CHOICE : out Choice_NUMBER);

procedure GET ITEM, READ ITEM, INTERPRET ITEM etc,.

12) INQUIRY FUNCTIONS : OPERATION STATE VALUE
procedure INQ SYSTEM STATE VALUE, 한 STATE LIST, WORKSTATION STATE LIST, WORKSTATION DESCRIPTION TABLE, SEGMENT STATE LIST, PIXEL, ERROR STATE LIST etc,.

13) ERROR CONTROL FUNCTIONS : EMERGENCY CLOSE XXX(SYSTEM NAME)
procedure EMERGENCY_CLOSE_XXX(SYSTEM NAME)

procedure ERROR_HANDLING
(ERROR_INDICATOR : in ERROR NUMBER :
SUBPROGRAM : in SUBPROGRAM NAME;
ERROR_FILE : in FILE_ID := STRING(DEFAULT_ERROR_FILE));
procedure ERROR LOGGING, ERROR HANDLING MODE etc,.

14) SPECIAL INTERFACE FUNCTIONS : ESCAPE.

15) APPLICATION FUNCTIONS : APPLICATION FUNCTION IN xxx(system name),
SPECIFIC FUNCTIONS IN BINDING etc,.

3.2 Implications of the binding model

1) The Functions Mapping of graphics systems are all mapped to implementation language procedures. The mapping utilizes a one-to-one correspondence between the graphics system and Ada procedures.

2) ERROR_HANDLING procedure may be replaced by one defined by the user.

3) Data mapping

The general correspondence between the data types and implementation binding datatype is summarized below :

- Graphics types are integer, real, string, point, vector, enumeration types, filter, pick path item, element reference types etc,. These types are mapped to implementation language integer, floating-point, string, record, enumeration types.

4) Abbreviations used in procedure names

ASF	:	aspect source flag
CHAR	:	character
ESC	:	escape
GDP	:	generalized drawing primitive
GSE	:	generalized structure element
HLHSR	:	hidden line/hidden surface removal
INQ	:	inquire

AROP : archive open
ASF : aspect source flag
ASAP : as soon as possible
ASTU : at some time
BNIG : before next interaction globally
BNIL : before next interaction globally
CBS : can be simulated
CHAR : character
DC : divice coordinate
GDP : generalized drawing primitive
GSE : generalized structure element
HLHSR : hidden line/hidden surface removal
ID : identifier
IMM : immediate
IND : index
IRG : implicit regeneration
MAX : maximum
MC : modelling coordinates
MI : metafile input
MIN : minimum
MISC : miscellaneous
MO : metafile output
NIVE : no immediate visual effect
NPC : normalized projection coordinates
PT : point
REF : reference
SF : scale factor
STCL : structure closed
STOP : sturcture open
UQUM : use quick update method
UWOR : update without regeneration
VC : viewing coordinates
WC : workstation closed
WS : workstation
WSCL : workstation closed
WSHOP : workstation open

5) Data type definitions

Generally, graphics data types are summarized below:

ARCL : archive closed

4. Conclusion

In the process of developing PREMO, there still remain many open issues. In this paper, focusing on the graphics object functions and using the "object-Z" method for the functional specification of graphics objects. We have addressed two major issues on binding model. Firstly, we have analysed all object types for the PREMO which are related the object oriented concept. Next we have proposed a design of graphics binding model. The other object bindings will be following future research.

Reference

[1] Recommendations of JTC1/SC24/WG6/(N1580) PREMO RG Meeting-Kyoto, Japan, June 1996.

[2] Roger Duke, Gordon Rose and Graeme Smith, Object-Z: a Specification Language Advocated for the Description of Standards, Software verification Research center Department of computer science The university of Queensland4072, Technical Report No. 94-95, Australia, December 1994.

[3] Min Hong, Kim, "An Adaption of CGRM to PREMO" Ph.D. Thesis, Ajou University, Feb, 1996.

[4] Ivan Herman, Graham J.Reynolds, PREMO: An Emerging Standard for Multimedia Presentation, Part I: Overview and Framework, IEEE, pp. 83-89, Fall. 1996.

[5] Ivan Herman, Graham, Reynolds, Reports of the Center for Mathematics and Computer Sciences, CS-R9554, "PREMO: An Emerging Standard for Multimedia Presentation", (Multimedia: Premo-IEEE/Report.doc.html# REF46720).

[6] Information processing system-Computer graphics and image processing-Presentation Environment for Multimedia Objects (PREMO).(ISO/IEC CD 14478-1) Part 1: Fundamentals of PREMO, 1997.

[7] Information processing systems-Computer graph-

ics and image processing-Presentation Environment for Multimedia Objects (PREMO).(ISO/IEC CD 14478-4) Part 4:Modelling, Rendering, and Interaction Component, 1997.

[8] ISO/IEC, "Information technology-Computer graphics-Computer Graphics Reference Model (CGRM)(ISO/IEC 11072)", 1992.

[9] Information processing systems-Computer graphics and image processing-Presentation Environment for Multimedia Objects (PREMO).(ISO/IEC CD 14478-3) Part 3:Multimedia Systems Services Component, 1997.

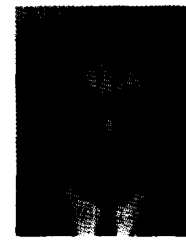
[10] Information processing systems-Computer graphics and image processing-Presentation Environment for Multimedia Objects (PREMO). ISO/IEC CD 14478-2) Part 2:Foundation Component, 1997.

[11] ISO/IEC 9593-2:1990, Information technology-Computer graphics-Programmer's Hierarchical Interactive Graphics System(PHIGS) language bindings-Part 2:Ada.

[12] ISO/IEC 9593-3:1990, Information technology-Computer graphics-Programmer's Hierarchical Interactive Graphics System(PHIGS) language bindings-Part 3:C.

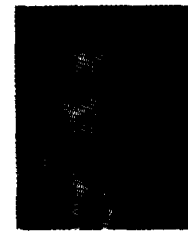
[13] ISO/IEC 8651-1:1988, Information technology-Computer graphics-Graphical Kernel System (GKS) language bindings-Part 1:Fortran.

[14] ISO/IEC 8651-2:1988, Information technology-Computer graphics-Graphical Kernel System (GKS) language bindings-Part 2:Pascal.



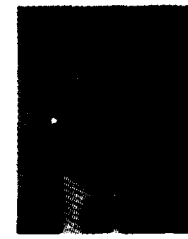
이 영 철

1990년 한국방송통신대학교 전자계산학과 졸업(학사)
1993년 연세대학교 산업대학원 전자계산학과(공학석사)
1997년 아주대학교 대학원 컴퓨터공학과 박사과정 수료
1975년~현재 한국통신 중앙통신관리단 재직중
관심분야:컴퓨터그래픽스, 이동통신, 멀티미디어



김 민 흥

1963년 한양대학교 공과대학 원자력공학과(학사)
1977년 정보처리 기술사
1978년 고려대학교 경영대학원 정보처리 전공(석사)
1993년 미국 Colorado 대학 방문교수
1996년 아주대학교 대학원 컴퓨터공학과(박사)
1981년~현재 경기대학교 이과대학 전자계산학과 교수
관심분야:운영체제, 정보기술 표준화 등



김 하 진

1962년 서울대학교 문리과대학 수학과(이학사)
1978년 Grenoble 1 대학교 대학원 응용수학과 D. E. A.(이학석사)
1980년 Saint-Etienne 대학교 대학원 응용수학과(이학박사)
1984년~1985년 프랑스 INRIA 초빙교수
1989년~1992년 한국 정보과학회 회장
1993년~1995년 아주대학교 공과대학 학장
1974년~현재 아주대학교 공과대학 컴퓨터공학과 교수
관심분야:컴퓨터그래픽스, 수치해석 등.