국제 표준 프리모-멀티미디어 객체 표현환경의 기능 특성

김 민 홍[†] · 김 하 진[††]

요 약

ISO/IEC JTC1/SC24 는 컴퓨터 그래픽스 표준을 관장하는 전문위원회이다. 이 위원회는 기술발전에 부응하는 새로운 그래픽스 표준인 PREMO를 구상하고 현재 개발작업이 진행중이다. PREMO는 종래의 그래픽스 표준과 달리 그래픽스와 함께 멀티미디어 객체 모두를 대상으로 표현환경을 제공하는데 그 뜻을 두고 있다. 본 논문이 작성되는 시점에서, PREMO의 제1부와 제2부는 CD 단계에서 검토중이고, 제3부는 WD단계, 제4부는 도입단계에 있다. 본 논문은 PREMO가 갖출 기능적 특성을 제시하려 한다.

Functional Characteristics of PREMO - An ISO Standard for Presentation Environments for Multimedia Objects -

Min-Hong Kim[†] · Ha-Jine Kimn[††]

ABSTRACT

ISO/IEC JTC1/SC24 is responsible for the development and maintenance of computer graphics standards. This subcommittee recognized the need to develop a new graphics standard, and launched a new project named PREMO which stands for Presentation Environment for Multimedia Objects. As the name implies, PREMO is intended to address not only graphics but also multiple media presentation using object concepts and a new formal description techniques. At the time of writing this paper, Part 1 and Part 2 were in the stage of Committee Draft and Part 3 is a Working Draft. The Part 4 is at an earlier stage within ISO/IEC. This paper is to present the procedures and works taken to develop PREMO.

1. Introduction

The first international standard for computer graphics was GKS published in 1985. After the emergence of GKS, a series of complimentary standards addressing different areas of computer graphics followed. Those are PHIGS, PHIGS PLUS, and CGM. Quite recently, GKS has been revised to support 3D (GKS-3D). In spite of significant differences in their functionalities, these standards share a common architectural similarity-libraries of a set of predefined functions-. These

† 정 회 원:경기대학교 전자계산학과 교수
†† 비 회 원:아주대학교 컴퓨터공학과 교수
논문접수:1995년 10월 10일, 심사완료:1996년 1월 18일

standards, however, have little chance of providing appropriate answers to the rapidly changing today's computer graphics technology.

The subcommittee (ISO/IEC JTC1/SC24) responsible for the development and maintenance of computer graphics standards recognised the need to develop a radically new standards. To this end, a new project was launched at an SC24 meeting in Chiemsee, Germany, in October 1992. Subsequent meetings resulted in a Draft for a new standard called PREMO (Presentation Environment for Multimedia Object). This new work was approved by ISO/IEC JTC1 in February 1994, and is now a major ongoing activity in ISO/IEC JTC1/SC24/WG6.

PREMO concentrates on presentation techniques, and this differentiates it from other multimedia standardization projects. The purpose of this paper is to present the the functionality of PREMO and some of its characteristic features different from foregoing standards. This paper is organized as follows. After introducing the emergence of PREMO, the requirements for a new graphics standard are presented in section 2, and section 3 describes functional architecture of PREMO. Concluding remarks are given in section 4.

2. Requirements for PREMO

The first generation graphics standards have served the graphics community well in a variety of ways. In the mean time, application areas and technologies have evolved so much that a big step forward in graphics standards is urgently needed.

Traditional computer graphics system and graphics applications have primarily been concerned with what might be called the presentation of synthetic graphics, that is, displaying pictorial information, typically on a screen or paper. New technological trend has evolved to the point that the computer graphics has become an inherent part of most applications.

In general, aims of any two presentations may be very different. For example, one might expect photorealistic images using very complex models describing the reality (e.g., commercial films, or high quality animation) while the other needs ergonomically sound and easy-to-grasp images of complex computed or measured data (e.g., scientific visualization, or medical imaging). In addition, many applications use multiple presentation of media simultaneously.

This has resulted in a wide diversity and a large number of requirements. This new standard is intended to adequately satisfy the presentation requirements of such diverse application areas as:

a) medical imaging,
b) CAD/CAM,
c) virtual reality,
d) geographic information system,
e) entertainment,
f) real-time command control systems,
g) education/training, and
h) simulation;

and such presentation techniques as:

i) animation,
j) simultaneous use of multiple media,
k) user interfaces,
l) scientific visualization,
m) data exploration, and
n) realistic rendering (including various dimensionalities, such as 2D, 2.5D, 3D, and integrating various media, such as video, sound, and other non-visual data).

This new standard, PREMO, will provide a common underlying functional nucleus to support these application areas and techniques, as well as future areas and techniques. Those requirements could be characterized in three categories as follows:

- the acceptance of new media;
- the needs for extensible and configurable graphics packages;
- the adaptability to distributed environments.

These requirements have shaped the architecture of

PREMO and also support interaction among application areas and presentation techniques.

2.1 Integration of media

Technological developments have resulted in new applications where traditional graphics alone cannot satisfy the requirements. Examples of applications where video, still images, and sound, etc., and synthetic graphics coexist are very common. It is therefore a natural consequence to have development environments that can support the presentation of different media in a consistent way, and which allow for the various media-specific presentation techniques to coexist within the same system. Integration is rather preferable than simple coexistence. It should be possible to describe media objects integrated with geometry and with one another, and also to describe and control their mutual influence.

The complete integration of various media and their presentation techniques within the same consistent framework is one of the major goals of PREMO, and one of the features which will make it very different from earlier SC24 standards.

2.2 Extensibility

PREMO is extensible in that it makes provisions for extending the functionalities specified. In particular, additional part of this standard may be developed to respond to the needs of specific application areas. Users of PREMO would be able to integrate their own extensions. The mechanism used to produce extensions in PREMO is defined so that its use does not adversely affect the portability of applications.

Different application areas have specific presentation requirements which are exclusive to that application area, e.g., simultaneous use of multiple media, simulation and animation. Fundamental extensions which are exclusive to an application area are specified in a standard way including considerations relating to compatibility, portability, and interoperability.

Many aspects of PREMO are extensible by an ISO-administered registration mechanism, so that a uniform description of the extension is available to all implementations. This differs from application-specific extensions since the latter may not go through an ISO registration process.

2.3 Configurability

There is a need for configurability because different application areas have different demands in presenting data. It is almost impossible to find a universal presentation technique that satisfies the needs of all application areas. Even though this could be done, this approach would result in a large and cumbersome mechanism instead of being a handy tool. Applications that need only linear data structures should not be forced to carry the burden of hierarchical or even more complicated data structures. Conversely, specialized data structures are required in some applications and can be provided by PREMO.

PREMO solves these problems through a configurable system design. The system offers a framework where various object types (modules in conventional sense) can be identified.

In a configurable system, the user is free to choose object types according to the special needs of particular applications. The advantages of a configurable system are:

a) Applications do not reference the whole system but only the specific object types they require. For example, an application might need only audio or video object types.

b) When introducing new techniques, such as shading within the graphics pipeline, or a special purpose graphics data storage, there is no need to implement a completely new graphics system for the realization of these new approaches. They can be integrated as new object types that fit within the existing framework.

2.4 Incremental, separable development

PREMO is described and structured in such a way that it can be developed incrementally to cope with newly emerging requirement. Further, it is possible to develop parts of the standard in an evolvable manner.

PREMO envisages a broad scope of functionality which cannot be covered by a single activity. Therefore this standard is designed as a multipart standard. At the time of this writing, PREMO is divided into four parts. The first part deals key concepts, describes the overall architecture of PREMO, and specifies the common semantics for specifying the externally visible characteristics of PREMO objects in an implementation-independent way. The second part defines objects that any conforming PREMO implementation shall support. The third part defines a Modelling, Rendering, and Interaction component which is targeted at providing paradigm independent support for high-level modelling and rendering, enhanced by time control and interaction. The fourth part defines a Multimedia Systems Services component which provides an infrastructure for building computing platforms that support interactive multimedia applications dealing with synchronized, time-based media in a heterogeneous distributed environment. Further parts are anticipated that will be appropriate for specific application areas such as VR, audio, tactile or olfactory.

2.5 Simplicity

Aspects, such as portability and maintenance, are greatly enhanced by keeping the underlying concepts simple. Simplicity means that PREMO is based on a general framework under which various sets of objects may be utilized. Objects are defined in terms of their externally visible behavior, thereby hiding implementation details. Hierarchical structuring of objects is possible allowing more complex things to be assembled from simpler parts.

2.6 Ease of use

PREMO is easy to use for the following classes:

a) end users: who work with applications based on PREMO;

b) programmers: who use PREMO components to build applications;

c) vendors: who develop, sell, and service the implementations of PREMO; and

e) system administrators: who control and manage multimedia systems.

It is possible to tailor a PREMO implementation to the minimum requirements to fit a particular application. PREMO is structured as a collection of independent components so that the programmers need to learn only about those components required for their application. PREMO supports the definition and registration of collection of objects. The use of object-oriented technology helps to achieve these goals.

2.7 Distributed environment

PREMO adopts object-oriented design philosophy while it is not the goal itself. This allows the construction of graphics data structure to be described same as the application programmer's perception. Programming thus can be more easier and understandable. PREMO supports distributed co-operative applications and users. It supports the management of shared resources and the development of distributed applications. Object-oriented technology also provides a framework to describe distribution in a consistent manner. Objects can be considered as closed entities which provide "services" via their methods. From the point of view of the object specification, it is immaterial how an object method is realized within the same program or via calls across a network. Since PREMO supports distributed applications, as well as multiple processor implementations, the invocation of PREMO operations may involve communication. Objects can learn of each other's existence and invoke each other's operations. Synchronization should be provided, since two objects could request the same operation in an overlapping fashion. Communication among PREMO objects and between PREMO objects and their client applications requires the use of underlying support facilities that are not addressed in this standard.

3. Functional architecture

The functional architecture is a conceptual description of the PREMO functionality. It identifies the functional areas common to all media components, but possibly having different realization in each environment. The functional architecture explains the nature of the rules for components such that they can be combined and linked to other standard or non-standard components.

3.1 Description techniques

In the past, the graphics standards community have employed formal methods in only avry limited way. The semantics of the first generation computer graphics standards, such as GKS and PHIGS, were described using natural language, and in some cases this has meant that ambiguities have been unnecessarily included in the specifications. The PREMO RG planed to solve this problem by employing formal description techniques at an early stage and to continue this activity throughout PREMO development. This activity started after the July 1993 PREMO meeting by appointing a Special Rapporteur for Formal Description Languages and some early results are documented.

PREMO functionality is described in terms of object behavior. Each PREMO object is specified by giving:

a) a definition of its interface;

b) a complete description of the object's behavior which is only visible through its interface. (Descriptions of the object's behavior, using formal description techniques, are developed and published separately recommending the use of Object-Z.)

3.2 Object model

PREMO uses an object model to support design portability and reuse of object definitions. The use of an object-oriented design leads to a natural description and provides, in particular, a way for explaining PREMO's extensibility and configurability aspects. Although the PREMO object model defines types and operations as concepts, systems that conform with the model need not provide objects that correspond to these concepts.

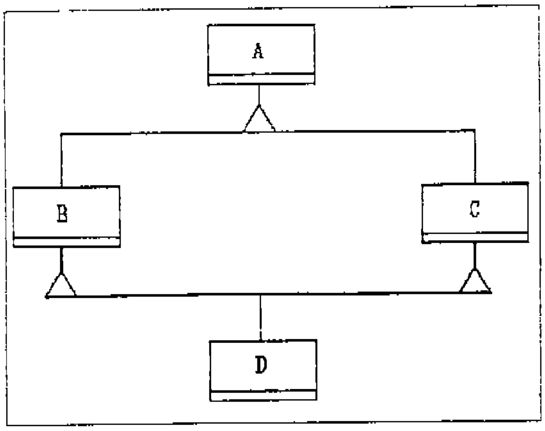
The PREMO object model is based on traditional concepts: objects,
object types, and subtyping. An object can model any kind of entity. A basic characteristic of an object is its distinct identity, which is immutable, persists for as long as the object exists, and is independent of the properties and behavior of the object.

3.2.1 Object types

Objects are created as instances of object types (e.g., person, color, segment). An object type defines the behavior of its instances by describing operations that can be applied to the object. Object types can be related to one another through supertype/subtype relationships. Subtyping is a relationship between types, based on their interfaces. It defines the rules by which objects of one type are determined to be acceptable in contexts expecting another type.

For example, type B is a subtype of type A, if B is a specialization or a refinement of A. Inheritance is a notational mechanism for reuse of operation. If type B inherits from type A then the definition of B inherits all the operations of A and may provide other operations(ref. Fig. 1). All these notions are well-known and are described in other publications; consequently, the details are not described here.

Although subtyping and inheritance are defined separately, the PREMO object model explicitly states how they are related. Indeed if B is declared to be a subtype of A, then B also inherits from A. The PREMO object model supports both multiple supertypes and multiple inheritance. As described above, subtyping and inheritance provide the basic mechanism in PREMO for extensibility and configurability.



(Fig. 1) Type graph

3.2.2 Operations

Operations are applied to objects. An operation describes an action that can be applied to an object via parameters. An operation invocation, called a request, specifies the operation and parameters. The operation associated with an object collectively characterize its behavior and has a signature which consists of a name, a set of parameters, and a set of return values.

The set of operation signatures defined for a type is called the interface of that type. The interface includes signatures that are inherited from supertypes. The interface of a type can be applied to all instances of that type.

3.2.3 Operation request

The external behavior of PREMO objects is based on the operations defined for the object. Requests for operations provide the only means of information transfer among PREMO objects. There is only one form of operation request. All requests are delivered at most once to the object.

Internally, an object has a finer control over the actions it has to perform to service the request. Each operation has an operation receptor, and an operation request amounts to putting a request into this receptor. The receptor of an operation may be in one of three modes: synchronous, asynchronous, or sampled. This mode is specified as part of the operation specification and immutable during the lifetime of the object. The default mode is synchronous. The intuitive meaning the three modes are as follows:

- If the operation receptor is synchronous, the caller is suspended until the callee has served the request. Data may be associated with the request, and the request may have return values.
- If the operation receptor is asynchronous, the caller is not suspended, and the request is queued on the callee's side. Data may associated with the request, but no return value is allowed in this case.
- If the operation receptor is sampled, the caller is not suspended, but the service requests are not queued on the callee's side. Instead, the respective requests will overwrite one another as long as the callee has not serviced the request. Data may be associated with the request, but the request shall not have return values.

The unusual feature of this model is the introduction of sampled messages. But this feature is not unusual in computer graphics. Consider the well known idea of sampling a logical input device, e.g., locator position values. A locator can send thousands of motion notification messages to a receiver object, and the latter can just "sample" these messages using the sampled message mode.

3.2.4 Operation dispatching

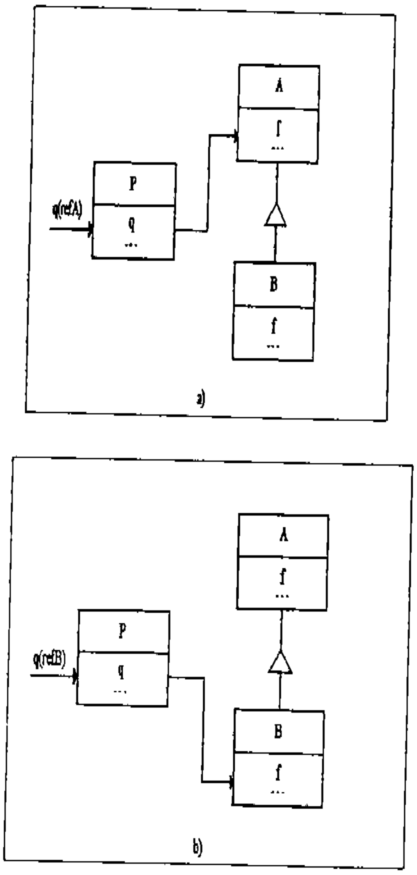
An operation request specifies the operation and the parameters possibly causing results to be returned. When an operation request is issued, a specific operation implementation is selected for execution. This selection is called operation dispatching.

If the signature of an operation Ω is:

$$\omega : (x_1 : \sigma_1, x_2 : \sigma_2, \cdots, x_n : \sigma_n) \rightarrow (y_1 : \rho_1, y_2 : \rho_2, \cdots, y_m : \rho_m)$$

where ω is the name of the operation. The operation signature specifies parameters with names x_i and types σ_i and results with names y_i and types ρ_i. And, say, σ_i is of type refA where A is an object type, and B is a subtype of A, the actual parameter x_i is permitted to be of type refB when invoking the operation ω.

An example is in Fig. 2. A and B are PREMO object types, and B is a subtype of A. The operation f is defined both in A and in B, i.e., both types provide an implementation for this operation. The two implementations will be denoted by A. f and B. f respectively. Let P be another PREMO object type, and p an instance of this type with operation q. The parameter list of q should include one of type refA. The operation p.q may be invoked with actual parameters of both refA and refB.



(Fig. 2) Operation dispatching

3.3 Component

A component in PREMO is a collection of object types and non-object types. Objects within one component are designed for a cooperation and offer a well-defined set of functional capabilities for use by other objects outside the component. A component offers a set of services and may also require services from other components. Components may also be defined by individual applications and be used as sets of objects directly linked to an application. A component also defines its dependencies on other components. For example, a component β may depend on other component α in two ways:

a) there are objects in β whose types are subtypes of object defined in α;

b) there are objects in β whose behavior depends on the availability of services offered by service object instances of objects defined in component α.

A component specification shall list all other PREMO components it depends on, and in which of the two ways listed above. The two ways of dependencies are not exclusive, and so, objects may rely both on the existence of services and may also inherit from the object types defined within the same component.

4. Conclusion

PREMO is a standard which is in progress. According to the timetable, the final text for an International Standard is scheduled by June 1997. As is mentioned in the introduction, we very briefly studied the characteristic features in PREMO. Formal description techniques based on Z and Object-Z, object model adopted from object oriented concepts, and incremental development policy are unique features in PREMO different from other graphics standards developed in SC24. In the process of developing PREMO, there still remain many open issues such as time and synchronization, the concept of quality of service, etc., unresolved.

References

[1] ISO, Information processing systems-Computer graphics-Graphical Kernel System (GKS) functional description (ISO IS 7942). Geneva, 1985.

[2] F. R. A. Hopgood et al., "Introduction to the Graphical Kernal System (GKS)", Academic Press, 1986.

[3] D. B. Arnold and D. A. Duce, "ISO Standards for Computer Graphics: The First Generation", Butterworths, 1990.

[4] G. Enderle et al., "Computer Graphics Programming-GKS-The Graphics Standard", Springer-Verlag, 1984.

[5] ISO, Information processing systems-Computer graphics-Metafile for the storage and transfer of picture description information (ISO IS 8632). Geneva, 1987.

[6] ISO, Information processing systems-Computer graphics-Programmer's Hierarchical Interactive Graphics System (PHIGS) (ISO IS 9592). Geneva, 1988.

[7] ISO, Information processing systems-Computer graphics-Programmer's Hierarchical Interactive Graphics System (PHIGS)-Part 4, Plus Lumiere und Surfaces (PHIGS PLUS) (ISO DIS 9592-4), 1991.

[8] ISO, Information processing systems-Computer graphics and image processing-Presentation Environment for Multimedia Objects (PREMO) (ISO/IEC CD14478) July 1994.

[9] G. J. Reynolds et al., "Report of the ISO/IEC JTC1/SC24 Special Rapporteur Group on Formal Description Techniques", ISO/IEC JTC1/SC24 N1152, 1994.

[10] R. Duke et al., "The Object-Z Specification Language" ver. 1, University of Queensland, 1991.

[11] J. M. Spivy, "The Z Notation: A Reference Manual", 2nd ed., Prentice Hall Int'l, 1992.

[12] D. A. Duce et al., "Formal Methods in the Development of PREMO", CWI, 1994.

[13] I. Herman et al., "PREMO: An ISO Standard for a Presentation Environment for Multimedia Objects", Proceedings of the Second ACM International Conference on Multimedia. ACM Press, 1994.

김 민 홍

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.