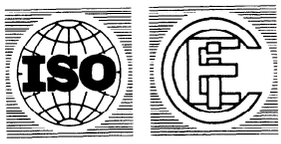
INTERNATIONAL STANDARD

ISO/IEC 11072

First edition
1992-10-01

Information technology – Computer graphics – Computer Graphics Reference Model

Technologies de l'information – Infographie – Modèle de référence



Reference number
ISO/IEC 11072:1992 (E)

Contents

© ISO/IEC 1992
All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland

Printed in Switzerland

Annexes

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 11072 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology.*

Annexes A to D of this International Standard are for information only.

Introduction

The Computer Graphics Reference Model (CGRM) describes the conceptual framework for computer graphics. Computer graphics is the creation of, manipulation of, analysis of, and interaction with pictorial representations of objects and data using computers.

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 forum whereby areas outside computer graphics can identify their relationships to computer graphics.

International Standards related to computer graphics include the following existing and emerging areas:

a) Open Systems Interconnection - Basic Reference Model;
b) Virtual Terminal Protocols and Terminal Management;
c) File Transfer, Access and Management Protocols;
d) Office Document Architecture and Interchange;
e) Text and Office Systems;
f) Exchange of Product Model Data;
g) Character Sets and Coding;
h) Open Distributed Processing;
i) Image Processing and Interchange.

This International Standard shall be the basis for the development of specific standards for computer graphics and will ensure their long term coherence based on objective rational foundations. Existing computer graphics standards will not necessarily fit precisely into the Reference Model. However, experience with current standards has significantly influenced the model.

This page intentionally left blank

Information technology – Computer graphics – Computer Graphics Reference Model

1 Scope

This International Standard, the Computer Graphics Reference Model (CGRM), defines a structure within which current and future International Standards for computer graphics shall be compared and their relationships described.

This International Standard defines a set of concepts and their inter-relationships which should be applicable to the complete range of future computer graphics standards.

This International Standard may be applied to:

a) verify and refine requirements for computer graphics;
b) identify needs for computer graphics standards and external interfaces;
c) develop models based on requirements for computer graphics;
d) define the architecture of new computer graphics standards;
e) compare computer graphics standards.

This International Standard does not define how computer graphics standards shall be defined and developed. It does not specify the functional descriptions of computer graphics standards, the bindings of those standards to programming languages, or the encoding of graphical information in any coding technique or interchange format. It is neither an implementation specification for systems incorporating computer graphics, nor a basis for appraising the conformance of implementations.

2 Definitions

For the purposes of this International Standard, the following definitions apply. An alphabetical list is given at the end of this clause.

2.1 computer graphics: The creation of, manipulation of, analysis of and interaction with pictorial representations of objects and data using computers.

2.2 application: The external object that uses computer graphics. Applications are not modelled in the CGRM, but their interactions with computer graphics are modelled.

2.3 operator: The external object that observes the contents of the display and generates physical input values. Operators are not modelled in the CGRM, but their interactions with computer graphics systems are modelled.

2.4 environment: A subdivision of the CGRM at a given level of abstraction. The definition of the environment includes the definition of its data elements and processing elements. Specific names are given to the five environments: construction, virtual, viewing, logical and realization (see 3.6.1).

2.4.1 construction environment: The environment that interfaces to the application.

2.4.2 virtual environment: The environment between the construction and viewing environments.

2.4.3 viewing environment: The environment between the virtual and logical environments.

2.4.4 logical environment: The environment between the viewing and the realization environments.

2.4.5 realization environment: The environment that interfaces to the operator.

2.4.6 higher environment: An environment closer to the application.

2.4.7 lower environment: An environment closer to the operator.

2.4.8 entity: An item of information stored within an environment or passed between environments. Entities are divided into three classes: input, output and control.

2.4.9 fan-in: The merging of entities from multiple, independent sources to produce a single stream (without changing individual entities) to be processed by a single environment.

2.4.10 fan-out: The generation of multiple, independent entities from a single entity without change. The generated entities are sent to independent environments.

2.5 external interfaces: The interfaces between the computer graphics system and the outside world, the interfaces communicate with the operator, application, data capture metafile and audit trail metafile.

2.5.1 operator interface: The interface between the realization environment and the operator. This is the only interface between the operator and the graphics system.

2.5.2 application interface: The interface provided by the construction environment to the application. This is the only interface between the application and the graphics system.

2.5.3 data capture metafile: An external object for representing all or part of a data element for storage, retrieval and transmission.

2.5.3.1 export: The process of generating a data capture metafile.

2.5.3.2 import: The action of setting part or all of a data element from a data capture metafile.

2.5.4 audit trail metafile: An external object for representing the sequential flow of information across the application interface.

2.6 processing element: A process in an environment: absorption, manipulation, distribution, assembly, and emanation.

2.6.1 absorption: A process which receives entities from the next higher environment and processes them for use within its own environment. Specific names are given to absorption at each environment level: preparation, production, projection, completion and presentation.

2.6.1.1 preparation: The name given to absorption in the construction environment.

2.6.1.2 production: The name given to absorption in the virtual environment.

2.6.1.3 projection: The name given to absorption in the viewing environment.

2.6.1.4 completion: The name given to absorption in the logical environment.

2.6.1.5 presentation: The name given to absorption in the realization environment.

2.6.2 emanation: A process which emanates token store and input control entities to the next higher environment after processing them. Specific names are given to emanation at each environment level: accumulation, abstraction, elevation, generation and utilization.

2.6.2.1 accumulation: The name given to emanation in the realization environment.

2.6.2.2 abstraction: The name given to emanation in the logical environment.

2.6.2.3 elevation: The name given to emanation in the viewing environment.

2.6.2.4 generation: The name given to emanation in the virtual environment.

2.6.2.5 utilization: The name given to emanation in the construction environment.

2.6.3 distribution: A process which distributes the composition and output control entities to the next lower environment.

2.6.4 assembly: A process which receives entities from the next lower environment for use within its own environment.

2.6.5 manipulation: A process which accesses and changes the contents of data elements.

2.7 data element: A store in an environment: composition, collection store, token store, aggregation store, and environment state.

2.7.1 composition: A spatially structured set of output primitives in a given environment. Specific names are given to the composition at each environment level: model, scene, picture, graphical image and display.

2.7.1.1 model: The name given to the composition in the construction environment.

2.7.1.2 scene: The name given to the composition in the virtual environment.

2.7.1.3 picture: The name given to the composition in the viewing environment.

2.7.1.4 graphical image: The name given to the composition in the logical environment.

2.7.1.5 display: The name given to the composition in the realization environment.

2.7.2 collection store: A storage facility for collections.

2.7.2.1 collection: A set of output entities which are named and may be structured. A collection may be manipulated to produce all or part of a composition in the same environment.

2.7.3 aggregation store: A storage facility for aggregations.

2.7.3.1 aggregation: A set of input entities which are named and may be structured. An aggregation may be manipulated to produce one or more input tokens in the token store in the same environment.

2.7.4 token store: A structured set of input tokens in a given environment. Specific names are given to the token store at each environment: lexeme store, information store, selection store, directive store and instruction store.

2.7.4.1 lexeme store: The name given to the token store in the realization environment.

2.7.4.2 information store: The name given to the token store in the logical environment.

2.7.4.3 selection store: The name given to the token store in the viewing environment.

2.7.4.4 directive store: The name given to the token store in the virtual environment.

2.7.4.5 instruction store: The name given to the token store in the construction environment.

2.7.5 environment state: Entities in the environment separate from other data elements: composition, collection store, token store, aggregation store.

2.7.6 editing: The change of entities within data elements in an environment.

2.8 output primitive: An atomic unit for graphical output in a given environment. There may be more than one class of output primitive. Geometric and other properties may be bound to an output primitive at its creation or later.

2.9 input token: An atomic unit for graphical input in a given environment. There may be more than one class of input token. Geometry and other properties may be bound to an input token at its creation or later.

2.9.1 property: A value that may be used by an output primitive or input token to specify its geometry or other characteristics.

2.9.2 geometric property: A property which is subject to modification by geometric transformations.

2.9.2.1 geometric transformation: A transformation that modifies the geometry of an input token or output primitive.

2.9.2.2 geometry: A property of an input token or output primitive used to define its shape, position, orientation and extent.

2.9.3 binding: The action of assigning a property to either an output primitive or an input token.

2.9.4 unbinding: The action of un-assigning a property from either an output primitive or an input token.

2.9.5 clipping: The action of constraining the geometric shape and extent of either an output primitive or input token to be within a specified region.

The following alphabetical list gives the sub-clause of each CGRM definition.

3 The Computer Graphics Reference Model

3.1 Environment model

The CGRM defines computer graphics in terms of five abstract levels called environments: construction, virtual, viewing, logical and realization (figure 1). The internal model of each environment is identical. The symmetry between input and output in the diagram reflects a symmetry of purpose rather than a symmetry of complexity. The CGRM defines operations on data elements in each environment.

The CGRM defines computer graphics output in terms of output primitives which make up a composition that is presented to the operator. The CGRM defines computer graphics input in terms of input tokens which make up a token store that is accumulated for the application in an appropriate form. Any connection between received input and generated output is conceptually handled by the application. The application may delegate this responsibility to specific environments. To allow complex graphical compositions, the CGRM defines a storage facility—the collection store—from which compositions may be derived. Similarly, a storage facility—the aggregation store—is defined from which entries in the token store may be derived.

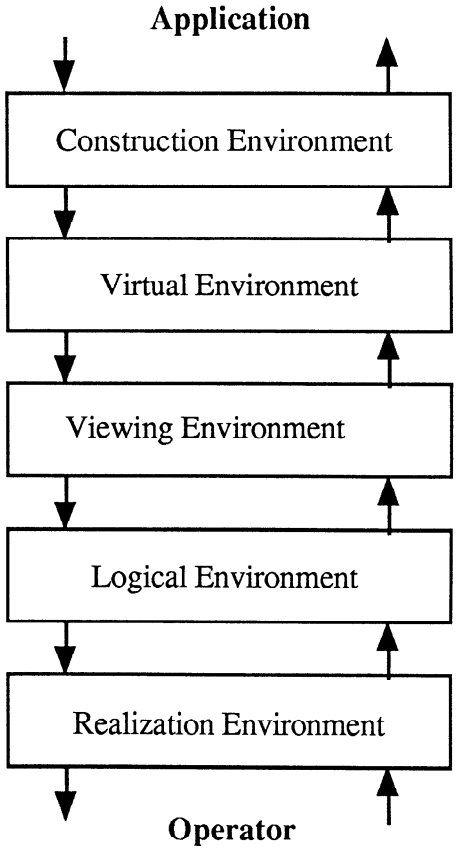


Figure 1 – Computer graphics environments

3.2 External relationships

The overall structure of the reference model is illustrated in figure 2.

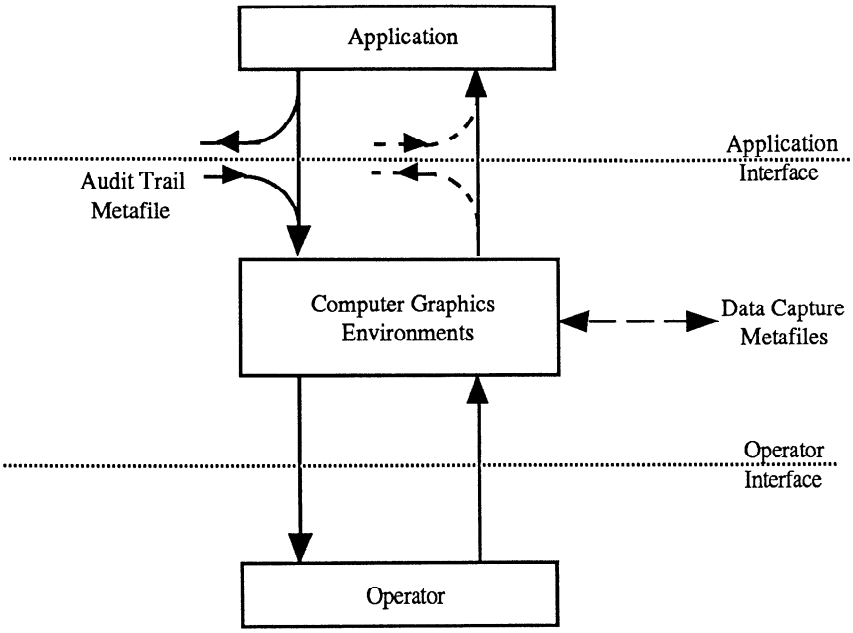


Figure 2 – External interfaces

Data capture metafiles may be generated by external agents and imported through the appropriate environment interface. Audit trail metafiles may be played back through the audit trail metafile interface at a later time. Thus communication between a computer graphics system and the "outside world" is described in the CGRM in terms of the following external interfaces:

a) *operator interface*: The interface provided by the realization environment to the operator. This is the only interface between the operator and the realization environment, and consequently between the operator and any computer graphics environment.

b) *application interface*: The interface provided by the construction environment to the application. This is the only interface between the application and the construction environment, and consequently between the application and any computer graphics environment.

c) *data capture metafile interfaces*: The interfaces provided by each environment for importing and exporting all or part of data elements: composition, collection store, token store, aggregation store and environment state.

d) *audit trail metafile interface*: The interface provided to record or playback the flow of information across the application interface.

The external objects, which are not part of the computer graphics environment, are:

e) *operator*: The external object that observes the contents of the display in the realization environment and provides physical input tokens.

f) *application*: The external object that interacts with the construction environment through the application interface.

g) *data capture metafile*: The external object for representing all or part of a data element for storage, retrieval, and transmission.

h) *audit trail metafile:* The external object for representing the flow of information across the application interface.

Individual standards may define additional external interfaces (different from the interfaces defined in the CGRM) to meet specific requirements of those standards. However interfaces different from those defined in the CGRM shall be given unique names distinct from those defined in the CGRM.

The CGRM does not specify the interfaces to either data capture metafiles or the audit trail metafile nor does it specify their format and internal organization. It does describe their conceptual contents.

3.3 Environment structure

Each environment consists of data elements and processing elements as depicted in figure 3. Entity flows are indicated by arrows:

a) An arrow coming from a data element means that entities in that data element may be used by the processing element to which the arrow is directed.

b) An arrow directed towards a data element means that entities in that data element may be set by the processing element from which the arrow is directed.

c) Dashed lines with arrows at each end coming from a data element mean that some or all of the contents of this data element may be exported to, and imported from, a data capture metafile.

d) An arrow between two processes means that entities may pass without storage to the process to which the arrow is directed.

e) An open arrow between two processes means that only control entities may pass between the two processes without storage.

No constraints are placed on the mechanisms by which the evolution of the behaviour of processing elements is controlled. Control entities are part of the environmental model though not explicitly shown in the diagram.

In each environment, there is a single interface for incoming entities from the immediately higher environment concerning graphical output and a single interface for incoming entities from the immediately lower environment concerning graphical input. The same coordinate system is used for both input and output entities passing between each pair of adjacent environments. The coordinate systems used by the composition, collection store, token store and aggregation store within an environment are the same. Consequently, all transformations occur in the absorption and emanation processes. There are interfaces for storage and retrieval of all or parts of data elements in data capture metafiles.

Some or all of a data element may be exported to a data capture metafile. All or part of the contents of a data capture metafile may be added to the appropriate current data element, or may replace all or part of it. A data element exported to a data capture metafile from one environment may only be imported into a data element of the same type in the same environment. A data capture metafile may only capture data elements from a single environment.

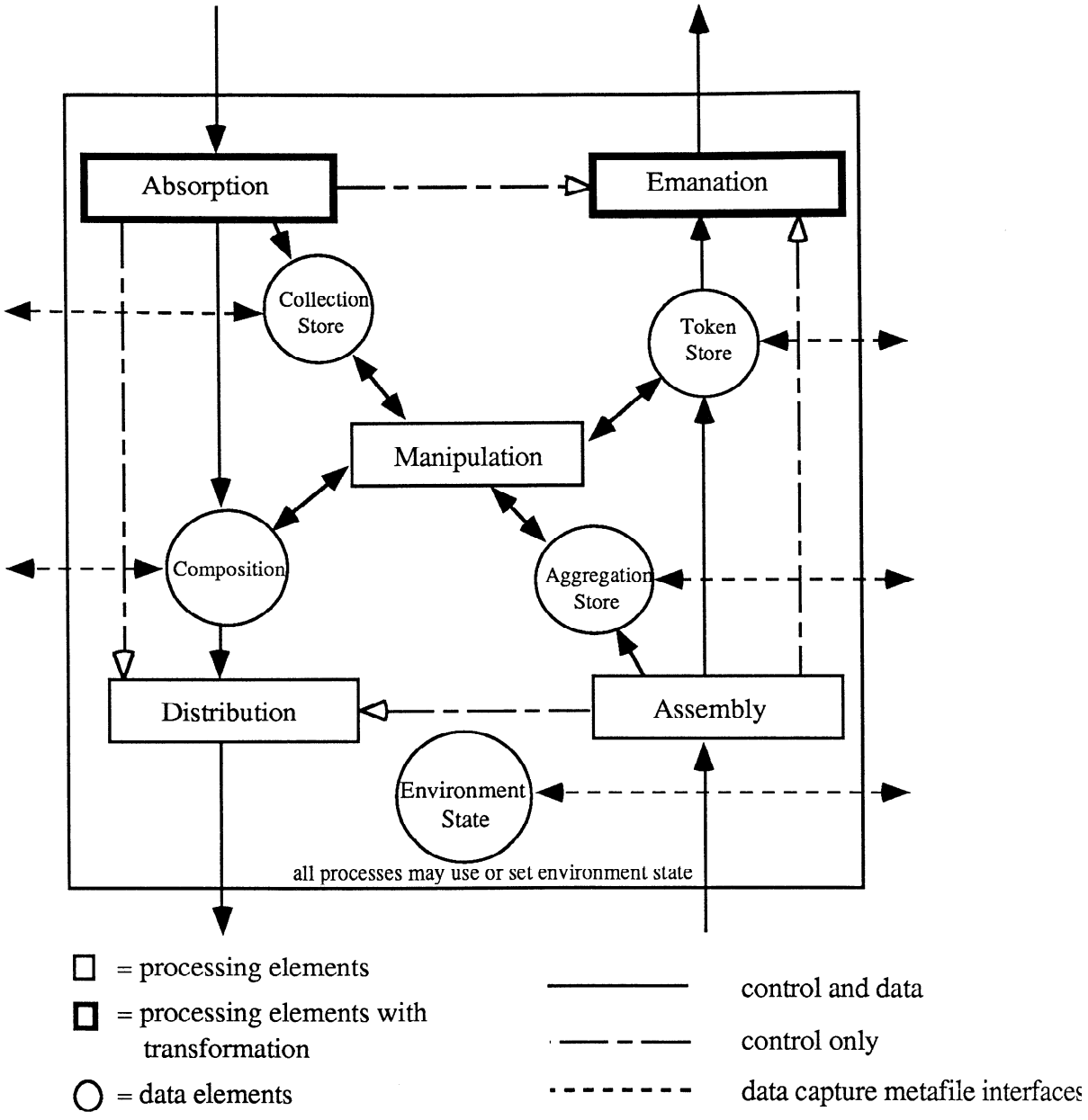


Figure 3 – Environment model

3.4 Data elements

3.4.1 Introduction

Each data element in an environment is affected by those processes having an arrow directed towards that data element (see figure 3). The operations on each data element may include editing (for example, create, insert, replace or delete) individual entities, groups of entities or the complete data element. Complete data elements or a subset of entities within a data element may be exported and imported. The entities within a data element may be inquired.

3.4.2 Composition

The composition is a spatially structured set of output primitives in a given environment ready for distribution. The composition represents the "output working set" for the environment. The CGRM places no constraints on the ordering of output primitives in the composition other than that the specification be well-defined and unambiguous.

The contents of the composition may be generated by manipulating the collection store (possibly as a result of input actions) in a given environment.

Specific names are given to compositions in each environment:

a) construction: model;
b) virtual: scene;
c) viewing: picture;
d) logical: graphical image;
e) realization: display.

3.4.3 Collection store

A collection is a set of output entities which are named and may be structured. A collection is intended for use within an environment. Collections are stored in a collection store. Structure may exist within collections which relate one entity to another. The manipulation process may change the contents of the composition using the contents of the collection store.

3.4.4 Token store

The token store is a structured set of input tokens in a given environment ready for emanation. The token store represents the "input working set" of the environment. Changes of input token class occur by manipulation of entries in the aggregation store to create new entries in the token store. Consequently, coordinate transformations to the coordinates of the next higher level occur during emanation and not assembly. The CGRM places no constraints on the ordering of input tokens other than that the specification be well-defined and unambiguous. Input tokens may result from assembly or manipulation.

Specific names are given to token stores in each environment:

a) realization: lexeme store;
b) logical: information store;
c) viewing: selection store;
d) virtual: directive store;
e) construction: instruction store.

3.4.5 Aggregation store

An aggregation is a set of input entities which are named and may be structured. An aggregation is intended for use within an environment. Aggregations are stored in an aggregation store. Structure may exist within aggregations which relate one entity to another. The manipulation process may use the contents of the aggregation store to change the contents of the token store .

3.4.6 Environment state

An environment may contain an environment state separate from the other data elements in the environment. When a process sets or uses environment state entities, it shall set or use environment state entities in its own environment. Environment state entities may be propagated to adjacent environments through control operations. Some or all of the environment state entities in an environment may be exported to or imported from a data capture metafile.

State information may be used and shared by all processes within an environment, as well as by all operations on data elements within an environment. Environment state entities may be used to determine whether or not a particular operation shall be performed.

There may be environment state entities which cannot be inquired from other environments.

3.5 Processing elements

3.5.1 Absorption

Absorption is the process which receives output entities from the next higher environment and applies the geometric and other transformations necessary to produce the entities in the form appropriate to its environment. Absorption transforms that part of the composition distributed from the next higher environment into a composition or collection(s) in its environment. This may involve transforming entities with geometric properties into the appropriate local coordinate system, elaborating output primitives into one or more replacement output primitives, as well as applying any clipping or non-geometric operations appropriate to the environment.

The absorption process in one environment may automatically cause changes made to its composition to be distributed to cause changes to be made to the composition at the lower environment. Alternatively, absorption may occur only when explicitly requested, for example by the next higher environment. The CGRM does not constrain when output primitives are received from the next higher environment.

Specific names are given to absorption in each environment:

a) construction: preparation;
b) virtual: production;
c) viewing: projection;
d) logical: completion;
e) realization: presentation.

Control information for input or output control may be consumed by the environment and change the environment state to dictate the behaviour of processing elements. Control entities destined for a lower environment may be modified by the absorption process. The results of the absorption process may be directed to the environment state, collection store, composition, directly to the next lower environment through the

distribution process, or back to the next higher environment directly through the emanation process.

3.5.2 Manipulation

Manipulation may process entities in any of the data elements, thereby creating entities in the same data element or any of the other data elements. Geometric and other transformations are applied as necessary. In this way, manipulation provides linkages between input and output within a specific environment.

3.5.3 Distribution

Distribution is the process which passes entities to the next lower environment. Only control entities shall be passed directly from the absorption process. No computer graphics transformations, geometric or otherwise, are applied to an entity as it is distributed. The CGRM does not constrain when entities are dispatched to the next lower environment.

3.5.4 Assembly

Assembly is the process which receives input entities from the next lower environment and passes them to the aggregation store or token store of the current environment. No computer graphics transformations, geometric or otherwise, are applied to the entities as they are assembled. Control entities may be consumed by the environment and change the environment state to dictate the behaviour of processing. The results of the assembly process may be directed to the next higher environment through the emanation process, or back to the next lower environment directly through the distribution process.

3.5.5 Emanation

Emanation is the process which passes entities to the next higher environment. Emanation takes control entities passed directly from the assembly process or input tokens from the token store of the current environment. Emanation transforms some or all tokens into tokens in a form suitable for assembly in the next higher environment. It may also receive and consume control entities directly from the absorption process. Transformations, geometric and otherwise, may be applied to the entity as it is emanated. Emanation may modify control entities destined for a next higher environment.

The emanation process in one environment may automatically propagate changes made to its token store to cause changes to be made to the token store at the next higher environment. Alternatively, emanation may occur only when explicitly requested, for example by the next higher environment. The CGRM does not constrain when tokens are dispatched to the next higher environment.

Specific names are given to emanation in each environment:

a) realization:	accumulation;
b) logical:	abstraction;
c) viewing:	elevation;
d) virtual:	generation
e) construction:	utilization.

3.6 Characteristics of specific environments

3.6.1 Environment details

Each of the five environments is conceptually present in the specification of a graphics system which interfaces to both the application and the operator (see figure 4). It is permissible for an environment to be null.

The main characteristics of each environment are:

a) *Construction environment*: In this environment, the application data to be displayed is prepared as a model from which specific graphics scenes may be produced. The application may only edit the model (composition) and collection store in the construction environment. Naming of parts of the collection is allowed and may be used to aid interaction. Input tokens in the instruction store are constructed in the precise form utilized by the application.

b) *Virtual environment:* In this environment, a scene of the model is produced. The scene consists of a set of virtual output primitives ready for viewing. The geometry of these virtual primitives is completely defined so that scenes are geometrically complete. Input tokens in the directive store are defined in the coordinate system used in the virtual environment. Input tokens may be differentiated by their associated properties.

c) *Viewing environment*: In this environment, a picture of the scene is generated by projection.The picture consists of a set of viewing output primitives ready for completion. Output primitives in the viewing environment may have a lower geometric dimensionality than in the virtual environment. Input tokens in the selection store are elevated to the virtual environment.

d) *Logical environment*: In this environment, a graphical image of the picture is completed ready for realization. The graphical image consists of a set of logical output primitives. Associated with each graphical output primitive is a set of properties associated with completion. In the logical environment, the complete set of properties shall be bound to the logical output primitive. Input tokens in the information store are converted, on abstraction, to device independent form with properties added to differentiate the origin of the input if required. The precise interpretation of the information store by the application may not be known in this environment.

e) *Realization environment*: In this environment, a display of the graphical image is presented. The display consists of a set or realization output primitives. This display need not directly correspond to a physical display. Tokens in the lexeme store are as received from a device, as presented by the operator interface, which need not directly correspond with ta physical input device. Only the operator may edit the token store and aggregation store in the realization environment. There may not be a one-to-one correspondence between the contents of the lexeme store and the information store. All properties associated with the physical input devices used will be known at this stage.

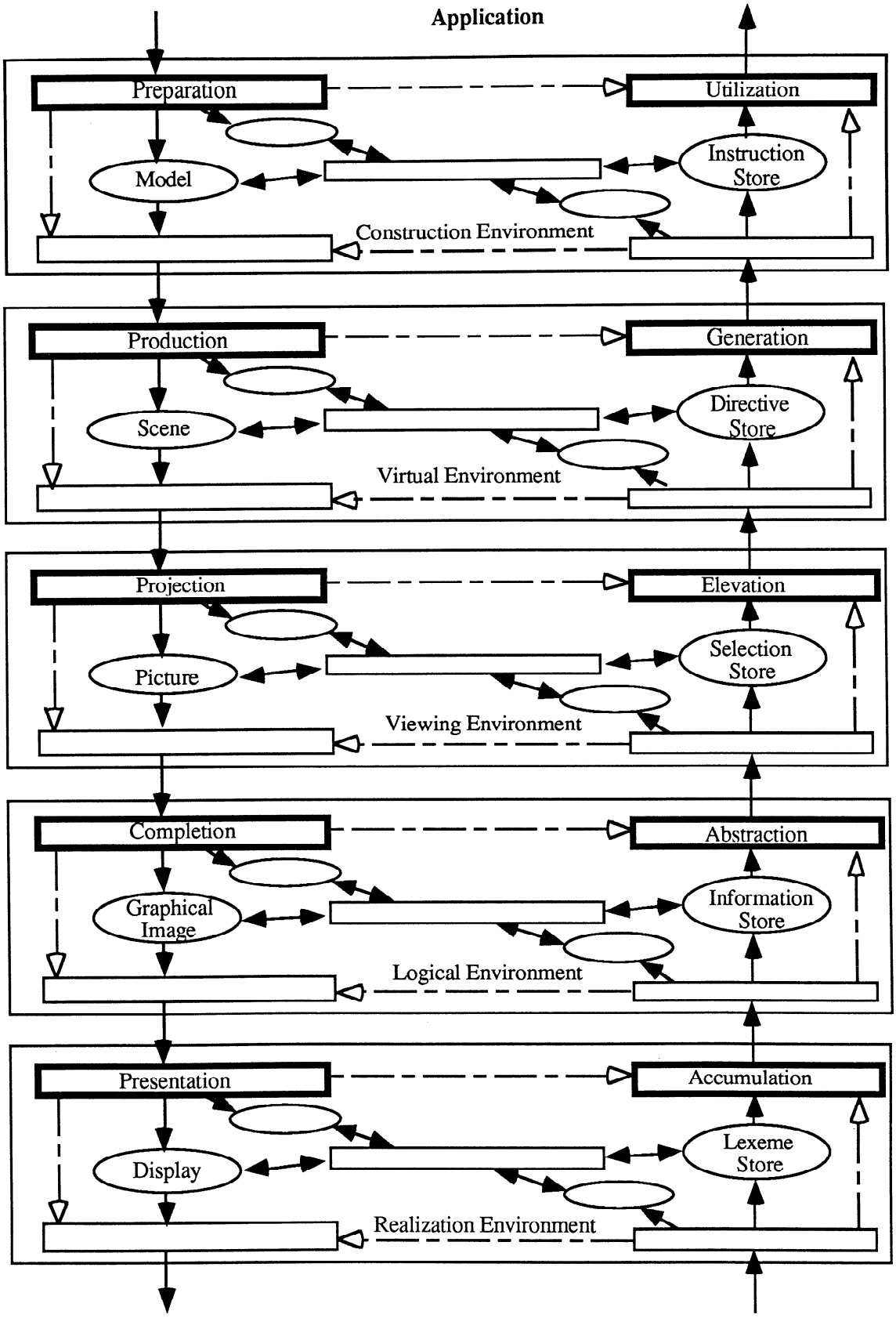


Figure 4 – Detailed environment model

3.6.2 Output primitives

Output primitives are atomic units used to describe graphical output in each of the five environments. A class of output primitives may have associated classes of properties and output primitives may be differentiated by their bound properties.

The properties of an output primitive may be bound to the output primitive if specified on creation or may be separately bound at some later time. For example, modal assignment from the environment state entities or inheritance are techniques which may be used. Properties may subsequently be changed or unbound.

An output primitive may be elaborated into one or more new replacement output primitives by absorption. Some or all properties of the original output primitive may be consumed in this process, thus losing their identity as properties. A property may be inherited or it may be determined from environment state entities.

Within each environment, the set of output primitives shall be closed under geometric transformations within that environment. That is, whenever the locus of points defining an output primitive is transformed, the resulting locus of points shall be represented as one or more output primitives in that environment. If clipping is provided in a specific environment above the realization environment, the result of clipping operations shall be expressible in terms of output primitives in the specific environment.

The application may only edit output entities in the construction environment. The geometry of output primitives in the construction environment may not be fully defined.

The geometry of primitives in the virtual environment is completely defined. Output primitives in the virtual environment cannot be completed because the required properties may not be fully defined in this environment.

Output primitives in the viewing environment may have a lower geometric dimensionality than in the virtual environment.

Output primitives in the logical environment have both completion properties and geometry completely defined.

Output primitives in the realization environment may be clipped. The result of clipping an output primitive in the realization environment must constrain the geometry of the resulting display to be within the specified region. The resulting display may not be expressible in terms of the output primitives defined in the realization environment.

3.6.3 Input tokens

Input tokens are atomic units used to describe graphical input in each of the five environments. The properties of an input token may be bound to the input token if specified when the input token is created or may be separately bound at some later time. For example, identification properties may be added dependent on the current contents of the composition. Input tokens may be transformed into one or more new input tokens by emanation. Some or all of the properties of the original input token may be consumed in this process, thus losing their identities as properties. A property may be determined from environment state entities.

Input tokens are produced in the realization environment using values provided by the operator interface. Properties associated with input tokens in the realization environment describe the input device used. The operator may only edit input entities in the realization environment.

Input tokens are produced in the logical environment by accumulation from the realization environment. Properties associated with input tokens in the logical environment describe generic input devices.

Input tokens are produced in the viewing environment by abstraction from the logical environment. Properties which may be associated with input tokens in the viewing environment include the geometry associated with the picture.

Input tokens are produced in the virtual environment by elevation from the viewing environment. If tokens in the viewing environment contain geometric information, the assembled input tokens in the virtual environment may have a geometric dimensionality different from that of the input tokens in the viewing environment.

Input tokens are produced in the construction environment by generation from the virtual environment. Properties associated with input tokens in the construction environment include the geometry understood by the application and any naming associated with both input and output by the application.

Properties of output primitives may constrain the possible input tokens allowed and affect the transformation applied to input tokens in defining tokens at the higher environment.

3.6.4 Properties

The properties of an output primitive specify its geometry and appearance. The nature of the actions performed in each environment implies some constraints on binding and consumption of properties:

a) *Construction environment*: Since properties controlling the interaction of the application or the operator with the model have to be bound prior to being processed by this environment, such properties have to be specified at the time of creation of the output primitive. Properties of input tokens specifying the actions to be taken on the model in this environment have to be bound to the input tokens prior to their arrival in this environment.

b) *Virtual environment*: Properties defining the geometry of output primitives in a scene have to be bound to the output primitives prior to being processed in this environment. Properties of input tokens defining their geometry have to be bound to the input tokens prior to their arrival in this environment.

c) *Viewing environment*: Properties specifying the viewing and projection of output primitives into a picture have to be bound to the output primitives prior to being processed in this environment. Properties of input tokens providing information about the view to be associated with the input tokens have to be bound to the input tokens prior to their arrival in this environment.

d) *Logical environment*: Properties precisely defining the completion of output primitives into a graphical image have to be bound to the output primitives prior to being processed in this environment. Properties of input tokens in this environment indicate generic device characteristics, and have to be bound to the input tokens prior to their arrival in this environment.

e) *Realization environment*: Properties precisely defining the presentation of output primitives on the physical display have to be bound to the output primitives prior to being processed in this environment. Properties of input tokens in this environment describe the physical input devices in use, as presented by the operator interface. The properties enter the environment already bound by these physical input devices.

3.6.5 Transformations

Transformations are operations that change the form of output primitives or input tokens within an environment. Rotation, scaling, translation and clipping are examples of transformations. Absorption may transform some or all of the composition distributed by the next higher environment before it is added to the composition or the collection store. Similarly, emanation may transform some or all of the token store in one environment before it is passed to the assembly process at the next higher environment level.

A collection may be used to generate entities in a composition at the same environment. The distribution of such a composition to the next lower environment may be realized by distributing the collection in the higher environment to an appropriately transformed collection in the lower environment and generating the corresponding entities in the composition of the lower environment from the collection at the lower environment. A similar procedure can be used for emanation of token stores generated from an aggregation.

3.6.6 Fan-in and fan-out

Fan-in is the merging of entities from multiple, independent sources to produce a single stream without changing individual entities to be processed by a single environment. Fan-out is the generation of multiple, independent entities from a single entity without change. The generated entities are sent to independent environments. The interfaces between adjacent environments are the only points where fan-out and fan-in may occur. Fan-out and fan-in may occur between any two adjacent environments. Both fan-out and fan-in may exist at the same time in a graphics system. Figures 5 through 8 illustrate fan-in and fan-out.

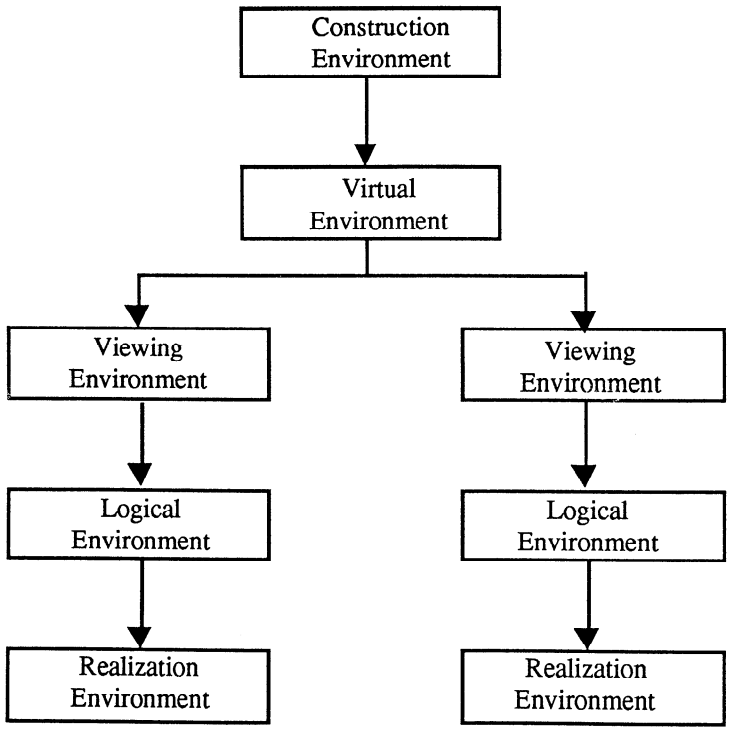


Figure 5 – Example of fan-out of output data

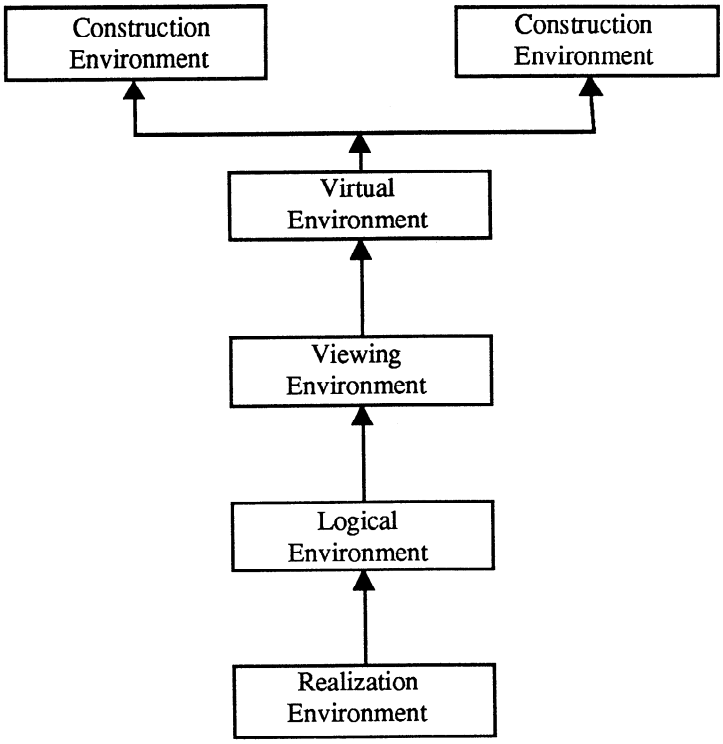


Figure 6 – Example of fan-out of input data

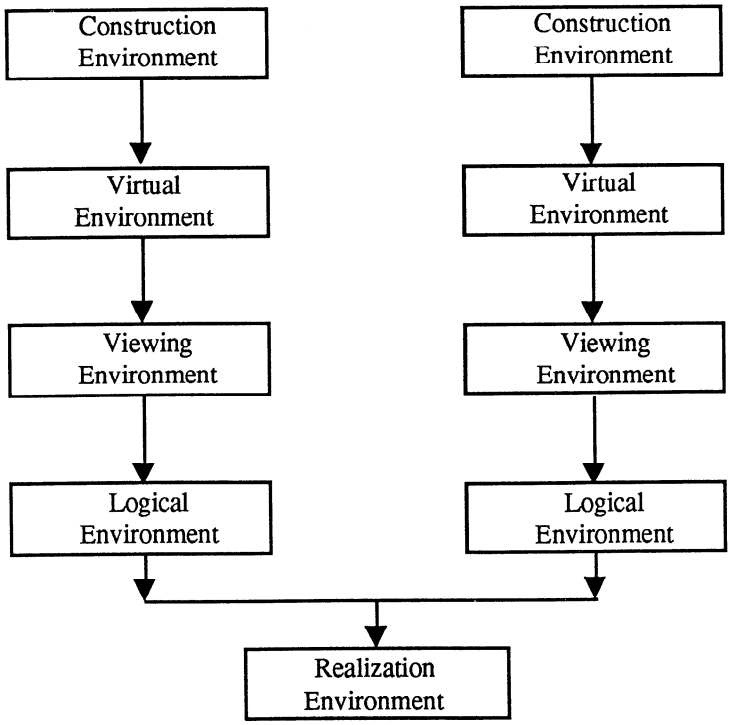


Figure 7 – Example of fan-in of output data

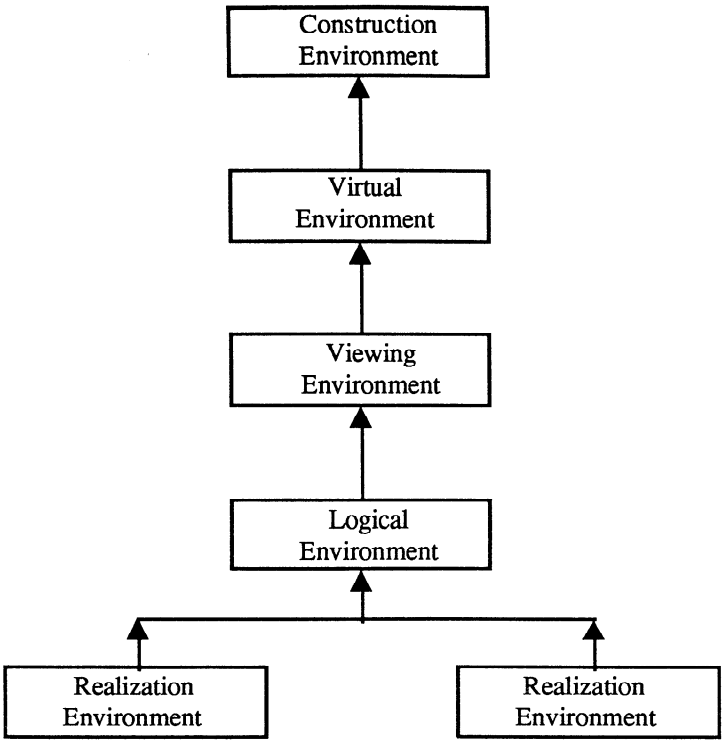


Figure 8 – Example of fan-in of input data

3.7 Relationship between output and input

There is a symmetry between output and input which is exemplified by the definition of output primitives and input tokens. The application expresses output in terms of output primitives. The operator observes the output and constructs the input on which the application is to act from input tokens.

There may be a linkage between input tokens and output primitives since properties of output primitives—for example, a coordinate transformation—may be controlled by entities. For example, the range of allowable values may be indicated by output primitives in the composition. The linkage between input tokens and output primitives is described in the CGRM by linked properties and storage of input tokens and output primitives within appropriate environments.

Graphically represented feedback and echoes of operator input are thus no different from other graphical output. The graphics system may provide facilities to enable the application to choose in which composition they appear for the first time. Similarly, the application may choose at which environment the naming of output primitives or their geometry can be used by input tokens.

Conceptually, input produces responses in the construction environment. However the application may delegate this responsibility to lower environments. For example, suppose echoes are defined to first appear at the logical environment and input values have a transformation which generates logical coordinate values and the echo is defined to be dependent on this value. In this case, the manipulation process in the logical environment may take the input logical coordinate value and use it in specifying the echo without application intervention. Similarly, the recipient of such delegated responsibility may further delegate this activity to an environment lower than itself.

3.8 Internal interfaces

The CGRM defines computer graphics in terms of five environments with an internal interface between each pair of adjacent environments. These internal interfaces do not provide access to computer graphics for either operators or applications. However they do serve to identify points where standard internal interfaces might be developed. Such interfaces could then be used to ensure the successful inter-working of separately-defined computer graphics standards.

Output primitives and control entities are distributed by each environment for absorption by the next lower environment. Entities that may be passed in this way include:

a) output primitives;
b) inquiry of stored entities, such as environment state;
c) editing instructions to be applied to data elements;
d) changing or setting stored entities, such as environment state.

Similarly, input tokens and control entities are emanated by each environment for assembly by the next higher environment. Entities that may be passed in this way include:

e) input tokens;
f) inquiry of stored entities, such as environment state;
g) editing instructions to be applied to data elements;
h) changing or setting stored information, such as environment state.

Annex A

(informative)

Existing standards and the CGRM

A.1 Graphical kernel system — ISO 7942

Figure A.1 illustrates GKS principles in the context of the CGRM. In GKS, the production of primitives in NDC space with attributes bound corresponds to the virtual environment. Since GKS is a 2D standard, the viewing environment performs only the identity transformation. GKS workstations correspond to the logical and realization environments. Binding of bundled aspects is done in the logical environment. Transformation of coordinates from NDC to DC is performed in the logical environment during completion.

In GKS, those aspects that are definitely geometric are bound at the virtual (NDC) environment However, some geometric text attributes (alignment) cannot be completely bound until the logical environment (contrary to the CGRM). The individual/bundled model fits into the CGRM as long as the complete geometry is specified at the NDC level. In individual mode, PHIGS has corrected this GKS deficiency for at least two fonts.

Segment store in GKS is identified as a virtual collection store (WISS) and a viewing collection store (WDSS).

The event queue in GKS corresponds to a token store in the virtual environment. Transformation of locator and stroke input values from NDC to WC coordinates is performed by an emanation in the virtual environment. Echoing occurs when manipulation creates the appropriate output primitives in the graphical image.

GKS has no clear concept of composition at any level. Its poor compatibility with CGM is a result of this.

A.2 Graphical kernel system for three dimensions — ISO 8805

GKS-3D has the same overall architecture as GKS. GKS-3D workstations correspond to the viewing, logical and realization environments. There is a clear separation between the scene created by the application in NDC3 coordinate space in the virtual environment and the picture to be completed and presented on a particular workstation. The viewing transformation is performed in the viewing environment.

The attribute binding model is the same as the GKS model.

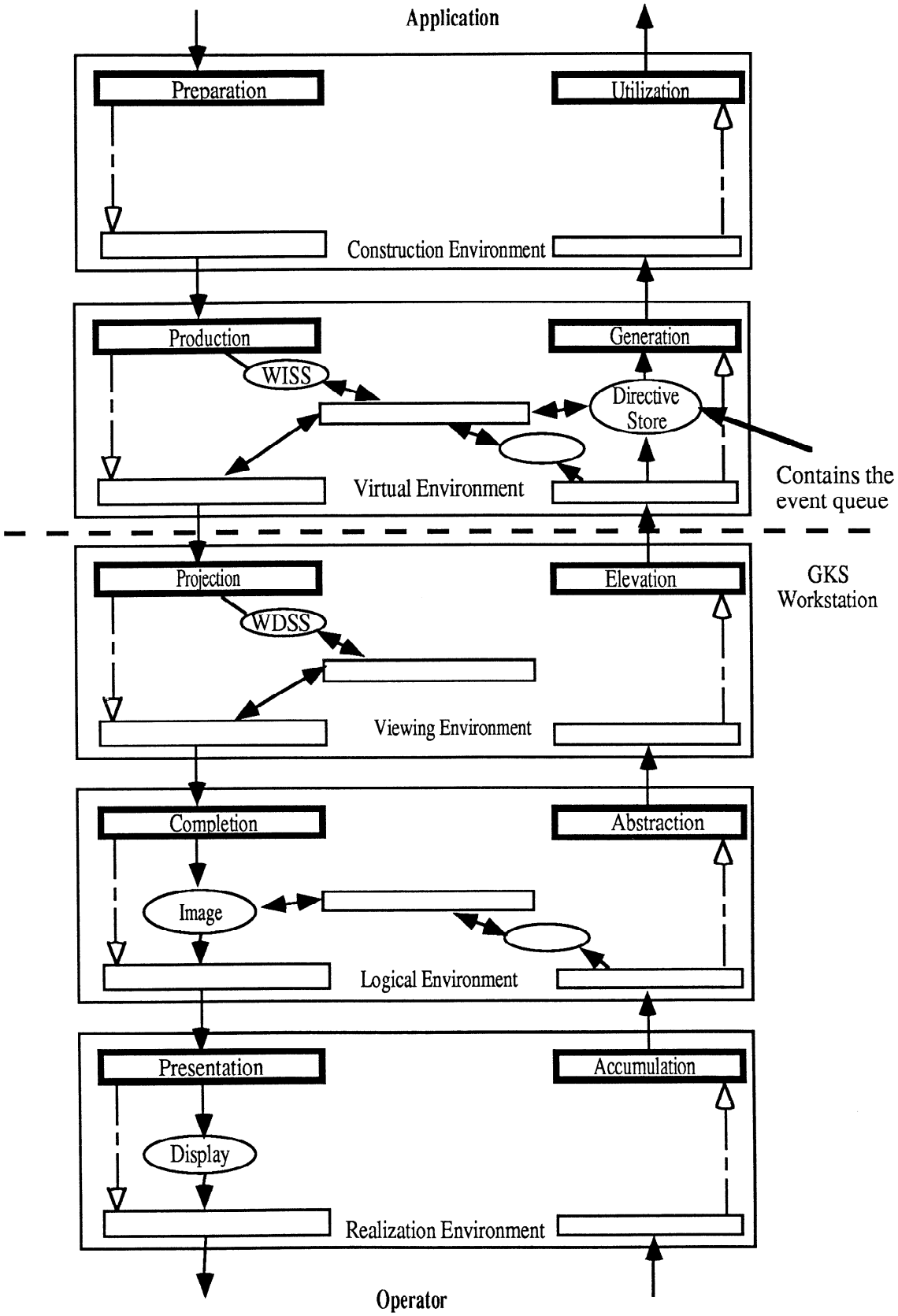


Figure A.1 GKS expressed in terms of the CGRM

A.3 Programmer's hierarchical interactive graphics system — ISO 9592

Figure A.2 illustrates PHIGS principles in the context of the CGRM. The PHIGS central structure store (CSS) corresponds to a collection store in the construction environment. Graphical output is generated by traversal, and display of graphics on a workstation is initiated by posting structures to a workstation. The PHIGS workstation corresponds to the viewing, logical, and realization environments. The set of structures posted to a workstation forms the conceptual collection store at the viewing environment of that workstation. The PHIGS traversal process is performed by manipulation in the viewing environment for each workstation. Posting is a result of distribution from the construction and virtual environments.

The output of the traversal process is a picture defined as the viewing environment composition specified in NPC coordinate space. This picture is distributed to the logical environment for completion and then on to the realization environment for display.

The appearance control mechanisms of PHIGS are essentially the same as those in GKS and GKS-3D. Attributes are bound to primitives when posted structures are traversed.

Picking is performed by the manipulation in the viewing environment using the information in the collection and aggregation stores to produce pick report tokens in the token store.

The PHIGS archive file is a collection data capture metafile at the construction environment.

A.4 Interfacing techniques for dialogues with graphical devices — ISO/IEC 9636

The CGI provides an interface to the viewing, logical and realization environments. The CGI interface corresponds to the interface between the virtual and viewing environments in the CGRM.

In the CGI, attributes are bound to output primitives in the viewing environment. Output primitives may be collected in a segment store which is a collection store in the viewing environment. Properties may be unbound from output primitives contained in the segment store. In the logical environment, coordinates are transformed from VDC to DC and bundled aspects are associated with primitives. There is no logical collection store. The CGI provides raster functions in the realization environment. These correspond to manipulation operations on the realization collection store, called the bitmap store in CGI.

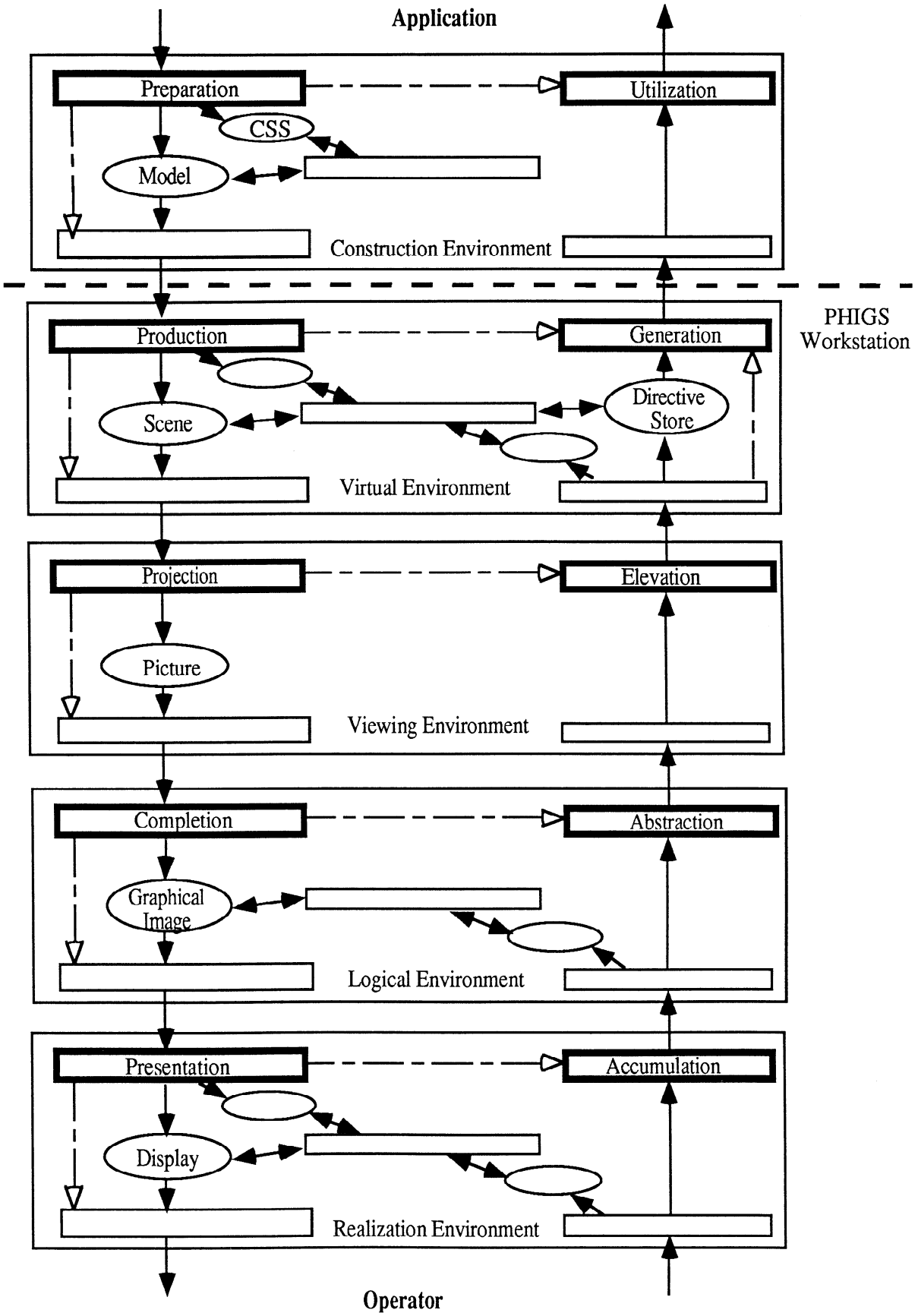


Figure A.2 PHIGS expressed in terms of the CGRM

A.5 Metafile for the storage and transfer of picture description information — ISO 8632

Figure A.3 illustrates CGM principles in the context of the CGRM. The CGM is primarily a data capture metafile for capturing 2D pictures (compositions) in the viewing environment. The segments of Amendment 1 to CGM are not parts of a collection. Instead they are simply a shorthand notation for representing the CGM picture.

The term "metafile" is used differently by CGM and the CGRM. In CGRM, a data capture metafile contains a single composition (picture). A CGM metafile may contain multiple pictures. A CGM may be thought of as a structured set of CGRM data capture metafiles.

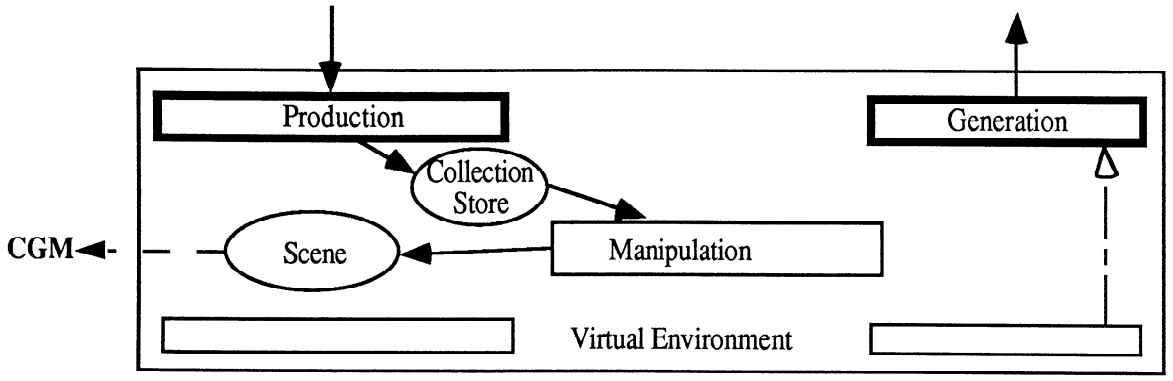


Figure A.3 CGM expressed in terms of the CGRM

Annex B

(informative)

The relationship of computer imaging to computer graphics

This annex describes computer imaging in terms of a model consisting of data and processes which transform that data. This is illustrated in figure B.1. While this model is sufficient for the purposes of this annex, it is anticipated that at some point in the future a more refined Computer Imaging Reference Model will be developed as a separate International Standard. Terms from the Computer Graphics Reference Model are italicized in this annex.

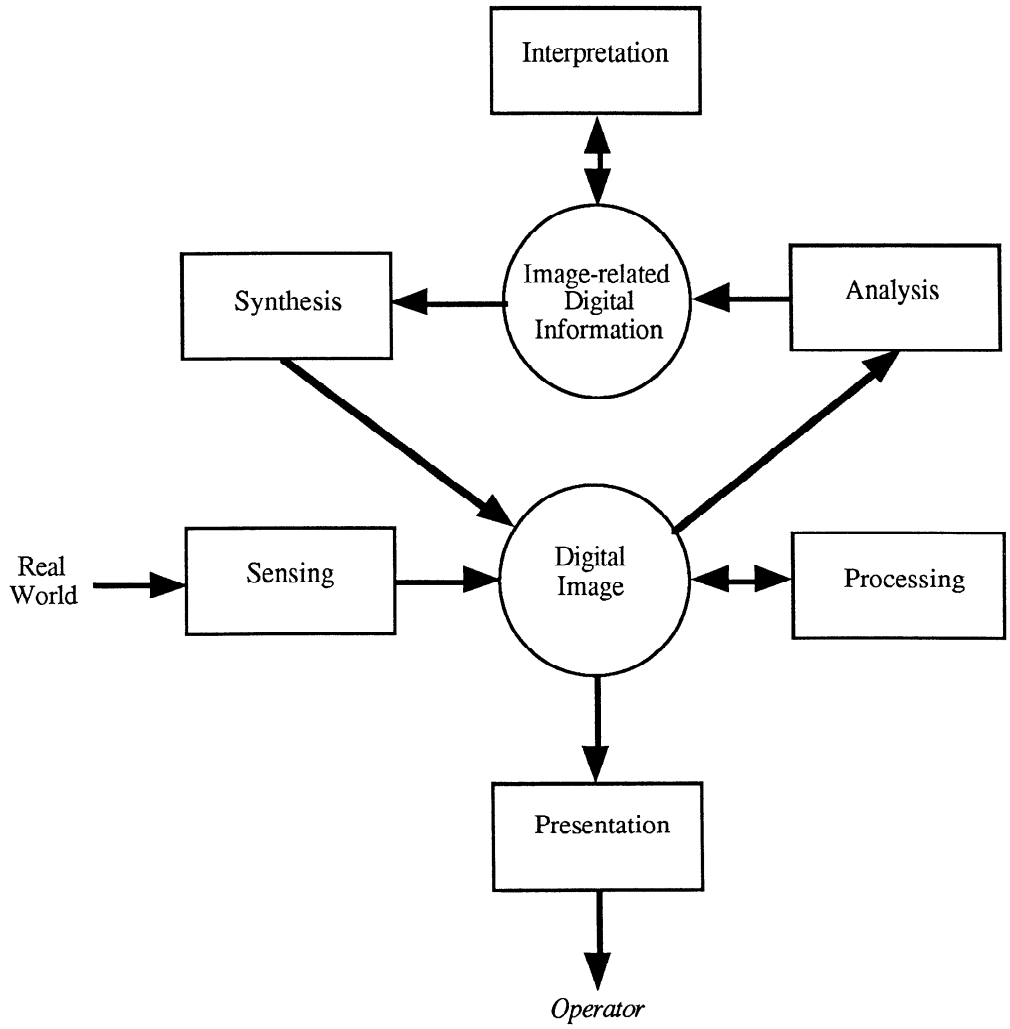


Figure B.1 – Computer imaging model

The fundamental functions of computer imaging depicted in figure B.1 are: Sensing, Processing, Presentation, Analysis, Synthesis, and Interpretation. The fundamental types of data are digital images and image-related digital information. A digital image can be thought of as an abstract function f. The domain of the function f is sampled at regular intervals to produce an array of image elements. Thus there is a spatial and temporal relationship between the values in the domain of f. Within this model an image may be thought of as existing in a potentially multi-dimensional space: real world information is temporally, spatially, or spectrally sensed through physical values such as light intensity, wavelength, pressure, distance, or temperature. Semantic information may be partially restored from a raw digital image through analysis and interpretation. Image-related digital information is typically non-image numeric measurements or structural models of objects and scenes. There is no inherent spatial relationship in image-related digital information.

Each function may be thought of as changing one of these data types into the same or a different type. **Sensing** transduces real world information into a digital image, while **Presentation** serves as its (partial) complement, changing a digital image into real world (physical) data, such as light from a CRT display or ink on paper. As an example of presentation relevant to the CGRM, image-related digital information may be converted to graphical *output primitives*. In addition, the *composition* in the *logical environment* (a *graphical image*) may often be converted into a digital image in a straightforward way. **Processing** may involve restoration, enhancement or other image-to-image operations. **Analysis** transforms digital images into image-related digital information, while **Synthesis** serves as its complement, changing image-related digital information into digital images. **Interpretation** transforms image-related digital information into other useful or refined forms of image-related digital information.

Computer imaging relates to computer graphics in several important ways that can be explained by the CGRM. These include:

a) *Output primitives* may be used to synthesize a digital image;

b) Certain *input tokens* may be converted into (sensed) digital images;

c) Image transformation techniques may relate to *completion*;

d) *Output primitives* may be derived from digital images; hence digital images may be presented by converting them into *output primitives*;

e) Image-related digital information may be converted into *output primitives*.

Figure B.2 illustrates the most important of these relationships in the context of the CGRM. Each relationship is explained in more detail in a subsequent paragraph.

The **Presentation** process may use computer graphics systems for much of its functionality. Digital images may be presented by converting them into *output primitives*. A common way to do this is to represent the digital image by colour or intensity values associated with each tile in the domain of the digital image, where the tiles are represented as regions in a particular space. In the *construction environment*, a digital image could be modelled using *output primitives* such as Cell Array. The form in which a digital image is held may often be suitable for presentation directly, without further conversion. In such cases, there is still, however, a logical conversion taking place.

Image-related digital information, such as a geometric model, may be used to synthesize digital images. *Output primitives* may be used in some cases to generate a *graphical image* in the *logical environment*. Such a *graphical image* may be transformed into a digital image in a straightforward manner.

One type of input covered by the CGRM is scanned (graphical) input. Such scanned input creates area *input tokens* in the *aggregation store* of the *logical environment*. These area *tokens* may be *manipulated* into the *composition* (*graphical image*) of the *logical environment* and may be *emanated* through the layers to be translated into a digital image. The *input tokens* may be echoed by each *environment*. Scanned images may be input through *data capture metafiles*. It is also possible that scanned images enter applications through mechanisms outside the scope of computer graphics.

The *completion* of graphical *output primitives* is accomplished in the *logical environment*. By converting the *graphical image* in this environment to a digital image, it becomes possible to apply image processing and image synthesis techniques for *completion*.

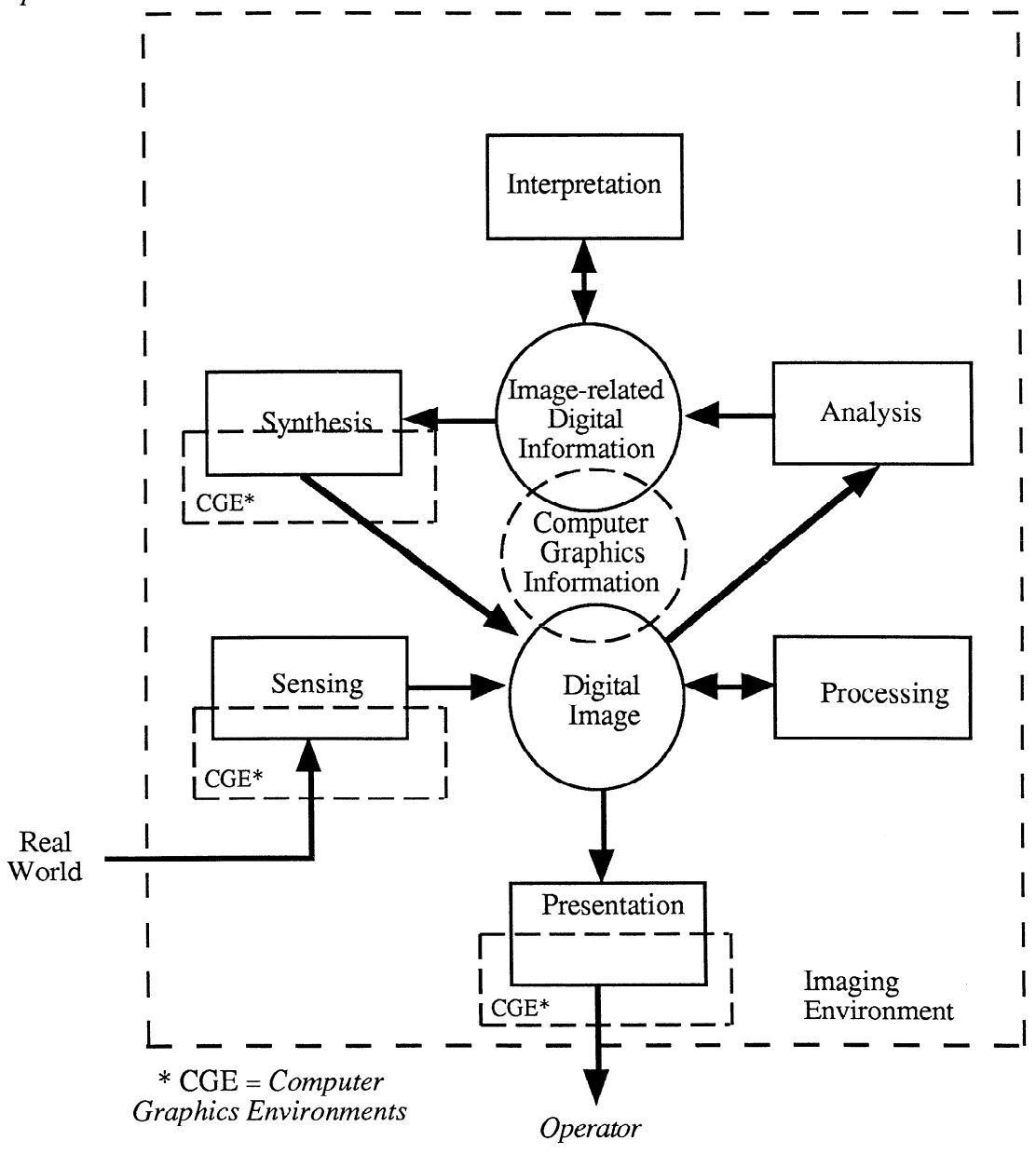


Figure B.2 – Relationship of computer imaging and computer graphics in the context of related environments

Annex C

(informative)

The relationship of window systems to computer graphics

C.1 Introduction

Both computer graphics systems and window systems are concerned with the creation of, manipulation of and interaction with objects displayed on shared display surfaces. However their major functions are different. Computer graphics systems are primarily concerned with graphical input and output while window systems are primarily concerned with the management of resources shared by multiple applications.

This annex describes two methods showing how a window system could be described using the CGRM. The terminology used here approximates to that used by the X Window System. The first method emphasizes the difference of purpose while the second emphasizes the similarity of structure.

C.2 Window systems

Window systems explicitly address many concepts that are outside of computer graphics. However, two important types of activities associated with applications using a window system are graphical:

a) control of displaying output primitives required by an application within a window and control of associated input data for the application;

b) displaying visual decorations required by the window manager to delineate and manage the display surface and associated inputs to the window system.

In terms of graphical output, the window manager's role is to manage the set of windows and other graphical entities required by applications and the window manager itself.

The following two sections give alternative ways in which the window system and window manager may be described in terms of the CGRM. With the variation in window systems currently available, the aim is to give a flavour of how a particular system could be described rather than a precise specification.

C.3 Windowing considered as an operator

In this approach, the window manager and window system can be defined as fitting below the computer graphics system. In the CGRM the operator is the external object that observes the contents of the display from the application and generates input values into the realization environment of the computer graphics system. The window system can be defined as the operator for several applications using the screen through the window manager.

Figure C.1 shows the window manager controlling the interface of two applications (A and B) with a human operator. An application produces graphical output in a pixmap possibly using X Window System graphics services. The window system is responsible for mapping the pixmap into a window, positioning it on the screen,

establishing its stacking order relative to other windows, decide whether it is currently visible or transformed to an icon.

The environment model of the window manager and window system need not be expressed in terms of the CGRM as there is no requirement for the operator to conform to it. However, the window manager is more abstract than the window system and has approximately the role of the construction and virtual environments. The window system corresponds to the logical and realization environments with the viewing environment not providing much additional functionality.

The set of pixmaps that might be displayed can be regarded as the collection store of the construction or virtual environment. The composition of the virtual environment is where the position of windows and icons is established.

Stacking of windows, drawing of borders, and colour mapping are functions performed by the window system and correspond to the kind of operations that would be performed in the logical and realization environments in a computer graphics system.

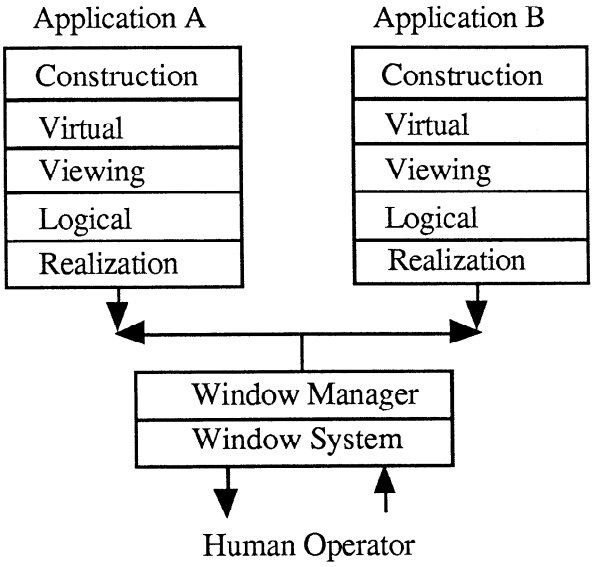


Figure C.1 – Windowing as operator

On the input side, the aim is for the window manager to deliver the logical input values required by the set of applications. These could be more abstract than the physical inputs by the operator. The type of changes possible within the window manager and window system might be:

a) *realization*: the raw (dx,dy) values from mouse movements and button state would be accumulated as tokens with absolute device coordinates and button events, respectively, at the logical environment.

b) *logical:* abstraction from the logical environment will identify the windows having the position within their focus and produce the relevant coordinate positions relative to the window origin. Iconized windows would be identified.

c) *virtual or construction:* tokens provided by the window system may be transformed to other tokens during elevation to the virtual environment. Positions relative to menus will be transformed to the associated CHOICE values; icons may be transformed to windows. The conversion of coordinate values to PICK names could be achieved here also. Tokens are converted to the logical input values required by the individual applications.

The description above is not complete. It gives an idea of how the various environment levels can be used to partition the functionality associated with graphics in the window manager and window system.

C.4 Windowing considered as part of the computer graphics system

C.4.1 Overview

In this approach, windowing is considered to be part of the computer graphics system. As such, windowing will be shown to have some of the characteristics of other computer graphics systems and to fit nicely into the CGRM in a manner similar to that of more traditional computer graphics systems.

Figure C.2 illustrates this concept. To more clearly highlight the role of the window system, the realization environment has been split into two sub-environments. One of these manages a composition in which all windows are merged, while the other produces the merged graphical image on some physical medium.

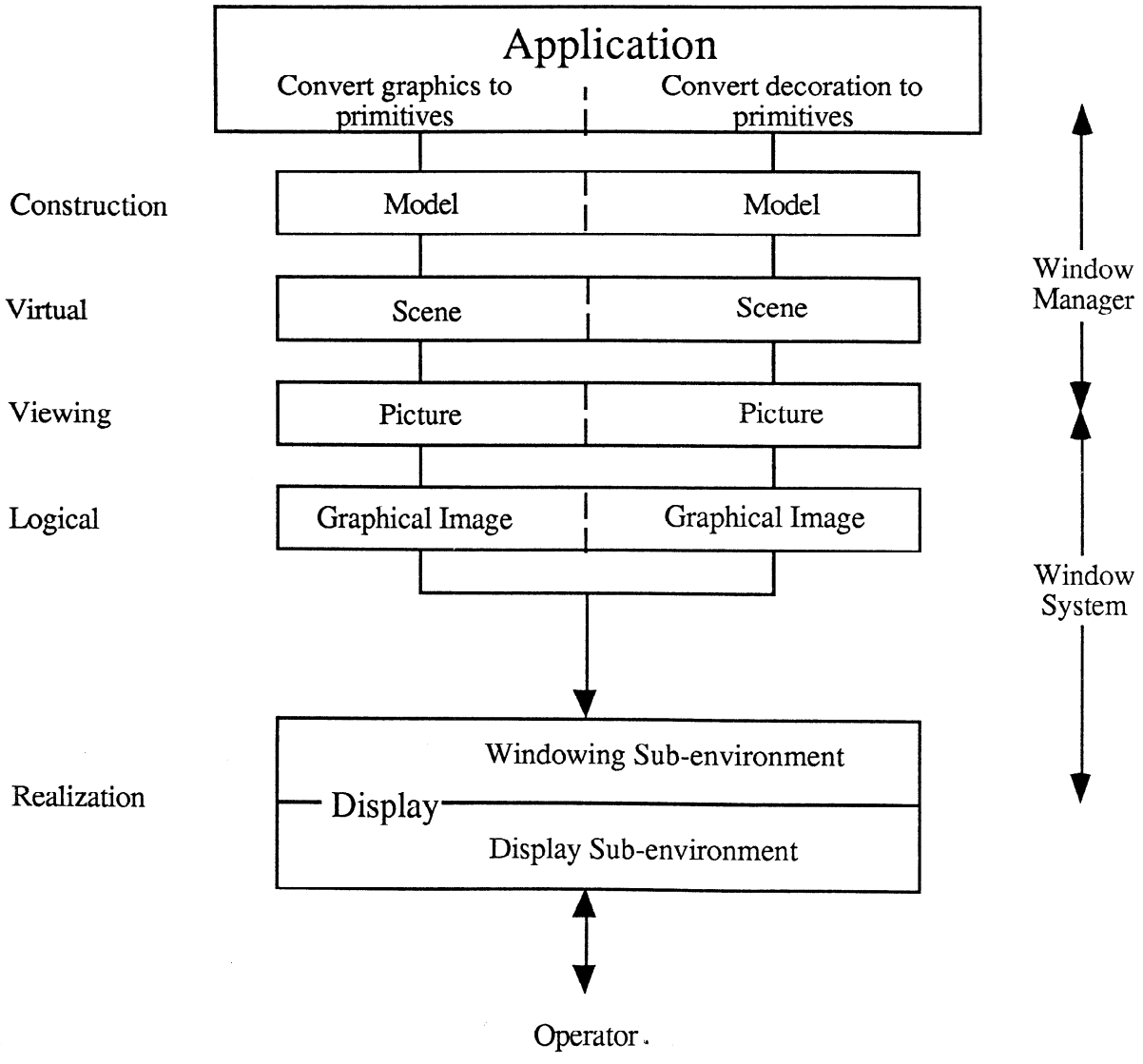


Figure C.2 – Windowing as a graphics system

C.4.2 Operations on windows

The following subclauses indicate where the various windowing functions would take place in the computer graphics environments.

C.4.2.1 Window creation

The window manager receives requests for window creation within the application and passes the request through the construction, virtual, viewing, and logical environments with alteration by the various absorption processes as appropriate. The realization environment receives this request in its windowing sub-environment where the window system establishes appropriate controls for the required resources. Typically, an identifier for the window is assigned by the window system and is sent back up through the environments to the application.

C.4.2.2 Window manipulation

Operations typically provided by a window manager include moving a window, resizing a window, iconizing a window, minimizing and maximizing a window, and changing the stacking order of a window (front/back) relative to other windows. The movement of a window and the resizing of a window can be considered to occur as an activity within the viewing environment. This is also true of the operations which minimize and maximize the window. Changing a window's stacking order is an operation on all windows managed by the window system and occurs in the windowing sub-environment of the realization environment.

C.4.2.3 Window destruction

When a window's client wishes to destroy the window, a request for this service is sent to the window manager. The window manager then asks the window system in the windowing sub-environment of the realization environment to release all resources associated with the specified window. This request passes through construction, virtual, viewing, and logical environments with these environments taking any action suitable to the request.

C.4.3 Operations on window content

The content of a window is comprised of graphical output primitives created by its client. An application modifies window content by using graphical operations in the construction environment of the computer graphics system associated with the client of the window.

The production of title bars, menus, and other visual decorations is handled by the window manager, which is a computer graphics system with graphical operations appropriate to the task of generating these elements. For example, the standard shapes for elements may be maintained in collections within the environments and incorporated into the composition as appropriate during the processing of the graphical operation request. The composition in any environment within the window manager consists of all visual elements placed in windows owned by the window manager.

Producing contents of windows owned by the application is delegated to the graphics system. For example, a PHIGS system might own one window, a text system might own another window, and a video system might own a third window. All of these are considered examples of graphics systems. In each of these cases, the graphical operations that produce the content of these windows are conceptually requested by the associated application which usually delegates some or all of the work to the relevant graphics system.

C.4.4 Displaying windows

The composition in the windowing sub-environment is comprised of the set of all portions of all windows which are visible (according to stacking order and each window's user-visible state). The composition of the display sub-environment is the screen.

Changes to the composition in the window sub-environment result either from changes in window content or from changes in the relationship between windows. In the former case, the changes are directed from the window's client. In the latter case, the window system may either regenerate the window content from the collection store in the window sub-environment or, if this is not possible, notify the client that the window content needs updating. Conceptually such notifications are directed to the application. However, an application may delegate responsibility for refreshing the composition to the computer graphics system.

C.4.5 Input

Input is produced by the operator using input devices. In the display sub-environment, the input data consists of raw values generated by the physical input device (e.g., mouse increments). The raw input data is converted into input tokens by the manipulation process in the display sub-environment. These tokens then become input data to the windowing sub-environment.

In the manipulation process of the windowing sub-environment, the window system identifies the client associated with the input data and creates input tokens which are sent to the logical environment of the computer graphics system which requests the input. Input devices in the windowing sub-environment are resources managed by the window system. The policy for establishing where input values are delivered is a function of the window manager. Once an input token has been emanated to the appropriate logical environment, it is processed as graphical input data.

Annex D
(informative)

Bibliography

[1] ISO 7942:1985, *Information processing systems – Computer graphics – Graphic Kernel System (GKS) functional description.*

[2] ISO 8632-1:1987, *Information processing systems – Computer graphics – Metafile for the storage and transfer of picture description information – Part 1: Functional specification.*

[3] ISO 8632-2:1987, *Information processing systems – Computer graphics – Metafile for the storage and transfer of picture description information – Part 2: Character encoding.*

[4] ISO 8632-3:1987, *Information processing systems – Computer graphics – Metafile for the storage and transfer of picture description information – Part 3: Binary encoding.*

[5] ISO 8632-4:1987, *Information processing systems – Computer graphics – Metafile for the storage and transfer of picture description information – Part 4: Clear text encoding.*

[6] ISO 8805:1988, *Information processing systems – Computer graphics – Graphical Kernel System for Three Dimensions (GKS-3D) functional description.*

[7] ISO/IEC 9592-1:1989, *Information processing systems – Computer graphics – Programmer's Hierarchical Interactive Graphics System (PHIGS) – Part 1: Functional description.*

[8] ISO/IEC 9592-2:1989, *Information processing systems – Computer graphics – Programmer's Hierarchical Interactive Graphics System (PHIGS) – Part 2: Archive file format.*

[9] ISO/IEC 9592-3:1989, *Information processing systems – Computer graphics – Programmer's Hierarchical Interactive Graphics System (PHIGS) – Part 3: Clear-text encoding of archive file.*

[10] ISO/IEC 9636-1:1991, *Information technology – Computer graphics – Interfacing techniques for dialogues with graphical devices (CGI) – Functional specification – Part 1: Overview, profiles, and conformance.*

[11] ISO/IEC 9636-2:1991, *Information technology – Computer graphics – Interfacing techniques for dialogues with graphical devices (CGI) – Functional specification – Part 2: Control.*

[12] ISO/IEC 9636-3:1991, *Information technology – Computer graphics – Interfacing techniques for dialogues with graphical devices (CGI) – Functional specification – Part 3: Output.*

[13] ISO/IEC 9636-4:1991, *Information technology – Computer graphics – Interfacing techniques for dialogues with graphical devices (CGI) – Functional specification – Part 4: Segments.*

[14] ISO/IEC 9636-5:1991, *Information technology – Computer graphics – Interfacing techniques for dialogues with graphical devices (CGI) – Functional specification – Part 5: Input and echoing.*

[15] ISO/IEC 9636-6:1991, *Information technology – Computer graphics – Interfacing techniques for dialogues with graphical devices (CGI) – Functional specification – Part 6: Raster.*

This page intentionally left blank

This page intentionally left blank

UDC 681.3.04(084)

Descriptors: data processing, information interchange, graphic data processing, models.

Price based on 35 pages