A Format for a Graphical Communication Protocol

STATUS OF THIS MEMO

   This paper describes the requirements for a graphical format on which
   to base a graphical on-line communication protocol. The proposal is
   an Interactive Graphical Communication Format using the GKSM session
   metafile. Distribution of this memo is unlimited.

ABSTRACT

   This paper describes the requirements for a graphical format on which
   to base a graphical on-line communication protocol. It is argued that
   on-line graphical communication is similar to graphical session
   capture, and thus we propose an Interactive Graphical Communication
   Format using the GKSM session metafile.

   We discuss the items that we believe complement the GKSM metafile as
   a format for on-line interactive exchanges. One key application area
   of such a format is multi-media on-line conferencing; therefore, we
   present a conferencing software architecture for processing the
   proposed format. We make this format specification available to those
   planning multi-media conferencing systems as a contribution toward
   the development of a graphical communication protocol that will
   permit the interoperation of these systems.

   We hope this contribution will encourage the discussion of multimedia
   data exchange and the proposal of solutions. At SRI, we stay open to
   the exploration of alternatives and we will continue our research and
   development work in this problem area.

I. INTRODUCTION

   A. Use of a Graphical Communication Protocol

      In the field of computer communications, a protocol is a procedure
      executed by two cooperating processes in order to attain a
      meaningful exchange of information. A graphical communication
      protocol is needed to exchange interactive vector graphics
      information, possibly in conjunction with other information media
      like voice, text, and video. Within this multi-media communication
      environment, computer vector graphics plays a key role because it
      takes full advantage of the processing capabilities of
      communicating computers and human users, and thus it is far more
      compact than digital images which are not generated from data
      structures containing positional information. Vector graphical
      communication trades intensive use of storage and processing, at
      the communicating ends, in return for a low volume of exchanged
      data, because workstations with graphical hardware exchange
      graphics commands in conjunction with large data structures at the
      transmitter and receivers. In this manner, the transmission of a
      single command can produce extensive changes in the data displayed
      at the sending and receiving ends.

      It is helpful to situate the aforesaid protocol at one of the
      functional levels of the ISO Open Systems Interconnection
      Reference Model [1]. Within such a model, a graphical protocol
      functionality belongs primarily in the application level, though
      some of it fits in the presentation level. We can distinguish the
      following components of a communication protocol:

         a) a data format
         b) rules to interpret transmitted data
         c) state information tables
         d) message exchange rules

      A format for a graphical protocol should provide the layout of the
      transmitted data, and indicate how the formated data are
      associated with interpretation rules. The choice of format
      influences the state tables to be maintained for the correct
      processing of the transmitted data stream. The graphical format
      has a minor influence on the exchange rules, which should provide
      for the efficient use of transmission capacity to transport the
      data under such a format. Besides the graphical format, there are

other aspects of a graphical protocol that determine state tables
and exchange rules. This paper concentrates in the data format,
and it does not discuss the message exchange. Nevertheless, we
discuss a simple software architecture for generating and
interpreting data streams written in our proposed format. Further,
we give an example of an application of a proposed format (in
Appendix B), and it illustrates the type of message exchanges that
are needed for establishing a communication session and exchanging
graphical information.

Those in the computer communication field are well aware of the
importance of widely accepted protocols in order to achieve
meaningful communication. Those who need to implement interactive
graphical communications today are confronted with the lack of an
standard for computer graphics communication among application
programs. Nevertheless, we can use some of the work already done
by the computer graphics standard bodies. As a matter of fact, ISO
and ANSI have already appended, to the Graphical Kernel System
(GKS) standard, the GKSM session metafile specification that has
many of the features needed for an on-line graphical protocol.

It is pertinent to mention an example of graphical communication
that illustrates the real-time nature of the interaction and also
illustrates the use of graphics in conjunction with other
information media. With audio-graphics conferencing, a group of
individuals at two or more locations can carry on an electronic
meeting. They can converse over voice channels and concurrently
share a graphics space on which they can display, point at, and
manipulate vector graphics pictures [2, 3, 4, 5, 6, 7].

The conference voice channels can be provided by a variety of
transmission technologies. The shared graphics space can be
implemented on workstations that display the pictures and permit
graphical interaction and communication with other locations. The
communication of operations upon pictures involves modifications
to the underlying data structures, but we are concerned with
graphical database updating only to the extent that such updating
supports the communication.

In order to play out a recorded graphical session, we will need
indications of the rate at which the graphical elements must be
shown and the graphical operations recreated. We do not include
the means for indicating the timing of a session in a format
because our main purpose is to use it in mixed-media communication
environments. In these environments, the play-out timing must be
compatible across information media in order to coordinate them.
Therefore, we leave the timing mechanisms to conference-control

        modules. We also leave to conference control processes the manner
        in which a conferee station emulates a graphical capability that
        it lacks. One example is the representation of color in monochrome
        displays.

    B. Relationship to Other Work

        There are a number of actual, and proposed, standards for graphics
        information exchange. In the following, we explain the reasons
        why, at present, none of them can be used as the basis of an
        on-line protocol. As some of these standards evolve, however, some
        may become suitable. Moreover, the experience gained with early
        on-line graphics communication systems will provide insight into
        the proper standard extensions to support more advanced systems.
        Such insight could also be used to modify the format proposed in
        this paper, which we consider an initial approach to the problem.
        In the future, the format proposed in this paper could be replaced
        by one of the aforesaid extended standards.

        The North American Presentation Level Protocol Syntax, NAPLPS,
        specifies a data syntax and application semantics for one-way
        teletext information dissemination and two-way videotex database
        access and transaction services. The two-way videotex operational
        model is based on the concept of a consumer and an information
        provider or service operator. Because of this asymmetry, it is
        assumed that almost all graphical information will flow from the
        provider toward the consumer. In the reverse direction, the
        consumer is expected to manipulate and transmit alphanumeric
        information, for the most part. Although this standard includes
        geometric drawing primitives, a user cannot directly modify shapes
        drawn with the primitives.

        At present, NAPLPS does not include interaction concepts like
        picture transformations or detectability, which are fundamental
        for attaining a shared graphical workspace. Neither does it allow
        key graphics input devices like mice, joysticks, stylus, rotating
        balls, or light pens, which are needed for simple and efficient
        editing of the shared workspace.

        We want to have user-to-user graphical communication that features
        the level of sophistication and ease of interaction provided by
        today's interactive graphics packages. Computer vector graphics
        can provide both because its paradigm includes an application
        program that keeps track of a very large number of possible
        changes of state of the displayed picture. In addition, the
        application drives a powerful graphics package, like GKS or ACM
        Core. In the videotex paradigm, the provider application only

allows limited changes to the displayed image, primarily database
retrieval requests. Also, the paradigm does not include a separate
graphics package. Both the graphics functionality and the data
format are collapsed into a coding specification, like NAPLPS.

In this paper we are interested primarily in business and
industrial applications where there is a two-way, or multi-way,
flow of vector graphics information among the users. The users
will have workstations with substantial processing and storage
capacities, and high-resolution monitors; moreover, the
communication will be on a distributed architecture not depending
on a central server host, like the provider application host of
videotex.

Currently, the videotex equipment at the consumer end consists of
inexpensive microprocessor-based decoders or personal computer
boards driving, in most cases, low-resolution standard TV sets and
personal computer displays. There is already affordable technology
to produce sophisticated decoders and high-resolution graphics
devices. The videotex standards need extensive revisions to take
advantage of these advances; in particular, they should consider
the receiving devices as capable of hosting a programmable
customer-application process. When this happens, videotex
protocols will be applicable to our intended problem areas [8].

The Computer Graphics Metafile [9] will become an international
and North American standard for graphics picture interchange in
the near future. However, the CGM, also referred as VDM, is a
picture-capture metafile that only records the final result of a
graphics session. It is not intended to record the
picture-creation process, which is fundamental for the interactive
applications that we are addressing. Moreover, the CGM is
presently aimed at a minimum support of GKS functionality. It will
be some time before the CGM will have some of the elements needed
for on-line interaction. If, after these additions, the CGM is
augmented for session capture, it would become a logical candidate
for a protocol format.

Another future standard is the Computer Graphics Interface, CGI
also referred as VDI [10]. The CGI is a standard functional and
syntactical specification of the control and data exchange between
device-independent graphics software and one or more
device-dependent graphics device drivers. A major use of the CGI
is for the communication between an application host and a
graphics device, but the asymmetry between its intended
communicating ends hinders the use of CGI for our purposes.

As previously stated, we want to take advantage of intelligence
and storage at the communicating ends in order to achieve powerful
information-conveying effects using narrow-bandwidth channels.
This requires that the format we seek must have items for
communication between two applications. In contrast, the CGI
streams are processed by device-dependent drivers, rather than by
applications. The CGI specification does include application data
elements, but only to be stored in a metafile. These application
data elements are not interpreted by the drivers, but by
applications that read the metafile, some time after metafile
creation.

Furthermore, the CGI has elements for obtaining graphical input,
as well as elements for inquiring graphics device capabilities,
characteristics, and states. Later, in Section III, we explain why
these two classes of elements are unnecessary for the
communication protocol we need. As the CGI evolves, it will
undergo significant changes, and, in the future, it may become a
very suitable kernel for the graphics protocol we seek. As a
matter of fact, the CGI will be the communication protocol between
graphical application hosts and graphics terminals. At SRI we are
tracking its evolution, and we are interested in defining a format
based on the CGI.

Finally, the Initial Graphics Exchange Specification [11] is not
aimed at our primary area of interest. The IGES defines standard
file and language formats for storing and transmitting
product-definition data that can be used, in part, to generate
engineering drawings and other graphical representations of
engineering products. Besides the CAD orientation of IGES, the
graphical output function may be secondary to other goals like
transmitting numerical-control machine instructions.

II. OPERATIONAL REQUIREMENTS AND USABILITY

The main goal of this paper is to lay the groundwork for the
development of a vector graphics format to be used as a basis for an
on-line graphical communication protocol. We call such a format an
"interactive graphical communication format," or IGCF. In this
section we describe some operational requirements and usable
characteristics for an IGCF.

A. Interoperation of Heterogeneous Systems

A first functional requirement is that an IGCF must permit
communication among heterogeneous graphical systems differing both
in the hardware used and in the software of their graphics

application interfaces. This is a fundamental for attaining
communication among similar graphical application programs running
on dissimilar hardware and using dissimilar graphics interface
packages. Some examples of such application programs are graphics
editors, CAD systems, and graphical database retrieval programs
communicating with other editors, CAD programs, and graphical
databases, respectively.

B. Picture Capture

A required characteristic of an IGCF is that it must be usable for
the exchange of static graphic pictures, i.e. for picture capture;
yet, it must not be restricted to final picture recording only.
There will be picture exchanges as part of the interactive
communication, and we anticipate the need to record the state of a
picture at some points during the on-line graphics engagement. We
foresee the creation of graphical IGCF libraries containing object
definitions and pictures for inclusion in new pictures. Since
metafiles have been used for a long time to capture pictures,
there is a strong motivation to base an IGCF on a metafile
standard in order to secure compatibility with a large number of
metafile sources and consumers.

C. Prompt Transmission

In some forms of interactive graphical communication, like
audiographics conferencing, it is critical to convey across users
the real-time nature of the interaction. This dictates that object
creations and manipulations be transmitted as they happen rather
than as a final result since a substantial part of the information
may be transmitted concurrently with the construction or operation
of an object, possibly through associated media like voice. Since
both construction and manipulation processes have to be
transmitted, there is a limit to the number of intermediate states
that can be economically transmitted.

A third requirement is, therefore, that the IGCF elements provide
fine "granularity" to convey the dynamics of the constructions and
manipulations. We believe that it is sufficient that the IGCF have
basic construction elements like polygons, markers, polylines, and
text strings and that it transmit them only when they are
completed; i.e., it is not necessary to transmit partial
constructions of such elements.

The problem for manipulations extends beyond an IGCF. Whereas we
know that an IGCF should include segment transformations, segment
highlighting and segment visibility on/off, the transmitter must

decide how often to sample an on-going transformation and transmit
its current state. The choice of a sampling frequency will depend
on the available transmission bandwidth.

D. Low Traffic Volume

In many of the applications we envision, coordinate graphics will
be transmitted over narrow bandwidth channels, and thus it is
essential to minimize traffic. Accordingly, several requirements
are imposed on an IGCF to take advantage of the characteristics of
the graphics communication intercourse and architecture in order
to minimize traffic.

An IGCF can help reduce traffic by including the basic geometric
objects from which so many other objects are built. Moreover, an
IGCF should permit the use of objects for the creation of more
complex objects; since reuse is very common, the result is a
reduction of traffic and storage cost.

E. Preservation of Application Semantic Units

A related requirement is that an IGCF must include elements to
represent graphical objects corresponding to real world entities
of the intended applications. For example, in a Navy application,
the entities of interest are carriers, submarines, planes, and the
like. We want to communicate such semantic units across systems
and to treat them as unitary objects because, in many
applications, communication is based on creating and operating
such units. If an IGCF has elements to represent such semantic
units, the communication traffic decreases because the entity
definitions can be transmitted only once and then reused, and
because the entities are manipulated as units rather than
separately manipulating their components.

It turns out that there is a small set of primary operations that
can be applied to a graphical object, and an IGCF must have
elements representing such operations. In contrast to dumb
graphics terminals receiving screen refresh information from a
host, we foresee graphical communication taking place among
intelligent workstations that can exchange encoded operations,
interpret them, and apply them to objects stored locally.

F. Transmission Batching

We previously indicated the desirability of conveying to the human
users the real-time tempo of interactive graphics exchanges.
However, it is possible to do so without having to transmit

immediately all IGCF elements. As a matter of fact, IGCF elements
should be divided into those causing a change on a displayed
picture and those that do not, although both classes may cause
changes to the stored graphical data structures.

It is only necessary to transmit immediately those elements
causing a visible change on a displayed picture because they are
the ones whose reception and interpretation delivers information
to a human user. The second class of elements can be batched and
queued for transmission until one element of the first class is
submitted. We call the first class update Group-1, and the second,
update Group-2.

The aforesaid division is quite important for packet
communications because each packet contains a hefty amount of
overhead control traffic. It is therefore mandatory to batch, into
a packet, as much client data as possible in order to reduce total
traffic. The batching units can be varied in size according to the
network traffic and response time of conference hosts. During
congested periods, the units may have to be increased, thus
lowering the number of messages, and then reduced when congestion
eases, thus increasing the number of messages.

G. Simple Translation Between IGCF and User Interface

According to the first requirement, an IGCF must permit the
interoperation of related heterogeneous graphics applications.
Such interoperation has, as an objective, the communication
between human users or between a human and a database.
Correspondingly, the interoperation involves a mapping between the
user interface commands and the IGCF elements. It is not advisable
to use the commands themselves as the IGCF elements; otherwise the
exchange would depend on the communicating systems, and every pair
of communicating systems would require an ad-hoc protocol.

An additional usability characteristic is that there must be a
simple mapping between IGCF elements and the actions represented
by the user interface commands employed for graphical
communications. This simplicity is a must because every
communicating graphical system must have a translator that ideally
should be very simple. It seems that the inclusion of command
sequence delimiters in the IGCF helps the simplicity since the
delimiters permit keeping a smaller amount of state information
for processing an IGCF stream.

We have verified the mapping from one set of commands for
audiographics conferencing to the IGCF proposed in this paper. The

         mapping from user interface commands to IGCF can be done in a
         direct and efficient manner; on the other hand, the reverse
         mapping, from IGCF to user interface commands, is a more difficult
         task. We anticipate that, in order to improve performance, we will
         have to map the IGCF elements to calls to lower level subroutines
         implementing the user interface actions. Whereas such mapping is
         conceptually no more complex than translating IGCF to the commands
         themselves, it will require considerably more programming.

III.   ELEMENTS OF AN IGCF

   IGCF Element Classes

         In this section we list the classes of elements that we believe an
         IGCF should have in order to exchange vector graphics under the
         requirements of the previous section. The classes correspond to
         the common function classes in computer graphics interfaces, and
         each contains elements corresponding to interface primitives and
         attributes. We do not list the elements for each class because
         they are exemplified by the elements in the proposed IGCF.

         In the following list, two categories of functions are missing:
         functions used to query the status of a graphics system, and input
         functions. As a matter of fact, an IGCF only needs to have
         elements representing actions that cause a change in the state of
         the communicating graphical systems, and the inquire functions
         obviously do not change their state. Even though an input function
         executed at the transmitting end causes a local change, it is not
         necessary to transmit the input command itself. The receivers only
         need to get the data input, in IGCF representation, and they can
         process the data in any manner, maybe simulating local input
         actions.

         Control

            Elements for workstation: initialization, control and
            transformation; and elements for normalization transformation.
            (The normalization and workstation transformations can be used
            to implement zooming.)

         Primitive attributes

            Elements for primitive, segment, and workstation attributes.

         Output primitives

            Elements for output primitives.

Segmentation

Elements for basic segmentation and workstation independent
segment storage.

Object manipulations can be implemented with segment
transformations. Object insertion can be implemented using
segment recall and segment visibility. Object deletion can be
implemented using segment deletion and segment visibility.
Object selection can use segment highlighting as feedback to
the user.

Dynamics

A considerable part of the graphical information exchanged
through an IGCF will be in the form of pointer movements over a
background picture. Pointer tracking is used to transmit points
sampled from a graphical pointer trace in order to reproduce,
at the receivers, the movement of the pointer at the sender
site. This can be done either by just moving the cursor or by
tracing its movement with a line. Rubber band echoes are used
to signal areas, routes, and scopes in a highly dynamic way.
These are indicated by an echo reference point and a feedback
point.

Hierarchical object definitions

The requirement for preserving application semantics dictated that
an IGCF include the means to represent objects that stand for
application entities, and to manipulate such entities as graphical
units. Furthermore, the low-traffic-volume requirement called for
the use of already existing objects for the creation of new ones.

One way to meet the aforesaid requirements is by including in an
IGCF the means to represent object hierarchies. In such a
hierarchy an object is a set of output primitives associated with
a set of attribute values or a set of lower-level objects, each
associated with a composition of transformations [12].

Graphics segments can be used to implement objects in the lowest
level of a hierarchy. The definition of a higher-level object can
be represented by sequences of IGCF elements describing the
definition process. Such a definition can be done by instantiating
lower-level objects with specific transformation parameters. Thus
an IGCF must incorporate brackets to mark the beginning and end of
object definitions, object instantiations, and object
redefinitions.

In order to complement the mechanism for object definition, an
IGCF must permit the use of a flexible alphabet for creating
object identifiers that ensure the uniqueness of an identifier in
a hierarchy. The construction of the object identifiers is not
part of an IGCF, an IGCF only has to represent the identifiers.
Further, an identifier has to be independent of a communication
session and a particular graphics system so that identifiers
created at a host during one session can be used, in other
sessions possibly involving other hosts, to recall the objects
they label.

We also leave to the communicating systems the implementation of
mechanisms to resolve duplicate identifiers when merging two
hierarchies, created in different sessions. In this paper we shall
limit ourselves to the warning that segment numbers do not qualify
as identifiers because they depend on the session and state of the
system in which they are created.

In addition to object definition and instantiation, an IGCF should
have elements representing operations on objects. The operations
so far identified are: transformation, deletion, display,
disappearance, expose, and hide. Expose is used to uncover objects
on a screen that are hidden by other objects; hide is used to
place an object behind others on a screen.

IV. A PROPOSED IGCF

  A. Using the GKSM as a Basis

An IGCF must be usable to transmit all graphical actions in a
conference session. This suggests to base an IGCF on a standard
session-capture graphics metafile, thus ensuring compatibility
with a large user population. We have based the proposed IGCF,
PIGCF, on the GKSM session-capture metafile specification because
GKSM contains many of the elements identified for an IGCF [14]. In
addition, the audit trail orientation of GKSM permits the
recording of interactive communication sessions for later play
out, and this is a feature that we anticipate will be frequently
used.

The GKSM is a proper subset of our PIGCF and thus any graphical
system developed to handle the PIGCF, can read a GKSM metafile.
Conversely, the applications using the PIGCF should have an option
for constraining session recording only to the GKSM part, possibly
suppressing some session events. By doing so, we will be able to
ship a GKSM metafile to any correspondent who has GKSM

interpretation software. Alternatively, an application with a
GKSM interpreter but without an PIGCF interpreter can read a PIGCF
file interpreting only the GKSM part and ignoring the rest.

Whereas the GKSM was specified for the GKS system, we believe that
the GKSM is a sound and general basis for all of our 2-D
applications. We feel that the GKSM specification is not parochial
to GKS systems but contains all the most useful items desired in a
metafile. In the future, we expect to tackle applications
requiring 3-D, like interactive repair and maintenance aids. When
GKS be augmented with 3-D capabilities [13], we will extend the
PIGCF with any necessary elements.

We are aware that the GKSM specification is not part of the GKS
standard itself but is an appendix recommending such a metafile
format. Nevertheless, all the GKS vendor implementations that we
know of, at the present time, support GKSM metafile output and
interpretation. If this trend continues, as we expect, we will be
able to exchange graphical files with a large base of GKS
installations. There will indeed be many of them since GKS will be
adopted as an standard by ISO and by many national standard bodies
in the near future.

B. Positional Information Coordinates

Following the GKSM convention, the PIGCF positional information is
in normalized device coordinates, NDC. Thus the originator of a
conference must indicate the workstation window for the
conference. This window is the sub-rectangle of the NDC space
enclosing the area of interest for the conference. In most cases,
the participating workstations will take this window as their own.
However, the graphical systems should provide for the possibility
of a workstation choosing a different workstation window, which
may contain the conference window or just overlap it. Except for
special cases, a conference originator should not state a
conference workstation viewport. In this manner, each workstation
can display its workstation viewport in the most convenient
portion of the screen.

There will be conferences where the participating workstations
will maintain the positional information in world coordinates, WC.
It might be necessary to reconstruct the world dimensions after
transmission because such dimensions have a relevant meaning for
the application, like sizes of components or distances. In this
case, a workstation will have to map from WC to NDC before
transmitting and from NDC to WC after receiving. At the outset,
the conference originator has to specify the world window and the

NDC viewport used in the conference in order for the conferencing
workstations to do such mappings. These mappings could be done by
the presentation layer, in terms of the ISO Open Systems
Interconnection Reference Model, in a manner that is transparent
to the communicating application programs.

Most often all workstations will have the same world windows and
NDC viewports. However, the graphical systems will provide for the
possibility of a workstation choosing a different window or
viewport, but such workstation will have to record the conference
ones for doing the aforesaid mappings. There are graphical
systems, like the ACM Core, that do not provide for a workstation
transformation. In such systems, the NDC viewport is considered to
be the workstation window for the aforesaid mappings.

C. Layers of the PIGCF

There are two levels in the PIGCF a lower level L and an upper one
U. The lower level L is just the GKSM metafile specification as
defined in Appendix E of the proposed GKS ANSI standard [14]. We
have excerpted most of Appendix E of [14] at the end of this RFC
as our Appendix A. All level L elements belong to the update
Group-1 except: SET DEFERRAL STATE, the output primitive attribute
elements, the workstation attribute elements, CLIPPING RECTANGLE,
CREATE SEGMENT, CLOSE SEGMENT, RENAME SEGMENT, SET SEGMENT
PRIORITY, and SET DETECTABILITY.

The upper level U is those elements that we believe complement the
GKSM for general on-line graphical exchanges. This layering
conforms to the graphics metafile level-structure described in
Enderle et. al [15]. Under such structuring, an application
oriented metafile can be based on graphical metafiles.

D. PIGCF Elements in the Level U

The level U items are encoded as GKSM user item elements so that a
PIGCF file will conform to the GKSM metafile specification.
Accordingly, a PIGCF file will be a GKSM metafile in its entirety.
We use the same formatting conventions as the GKSM specification.
Those unfamiliar with these conventions should read the beginning
of the appendix. The following items belong to the second update
group: the two items for object definition, the two items for
object redefinition, the two items for object instantiation, the
two items for normalization transformation, SELECT COMPONENT, and
RECALL LIBRARY. The remaining items belong to the first update
group.

Items for Object Definition

BEGIN DEFINITION

| 'GKSM 120' | L |

Indicates beginning of object definition sequence

END DEFINITION

| 'GKSM 121' | L | I |

Indicates end of object definition sequence. I(Nc): object
identifier (N preceding c, i, r means an arbitrary number
of characters, integers, or reals.) Objects defined
interactively are made visible on the screen; i.e. they are
automatically instantiated. If only the definition is to be
kept but not the image, a DISAPPEAR item must follow.

BEGIN REDEFINITION

| 'GKSM 122' | L | I |

Indicates beginning of object redefinition sequence
I(Nc): object identifier

END REDEFINITION

| 'GKSM 123' | L |

Indicates end of object redefinition sequence

Items for Object Instantiation

BEGIN INSTANTIATION

| 'GKSM 124' | L | I |

Indicates beginning of object instantiation sequence
I(Nc): Object identifier

END INSTANTIATION

| 'GKSM 125' | L |

Indicates end of object instantiation sequence

Items for Object Manipulation

TRANSFORM OBJECT

| 'GKSM 126' | L | C | I | M |

Apply transformation M to object I
C: number of characters in identifier
I(Nc): object id
M(6r): upper and center rows of a 3x3 matrix representing
        a 2D homogeneous transformation [12].
        M 11 M 12 M 13 M 21 M 22 M 23

DELETE OBJECT

| 'GKSM 127' | L | I |

I(Nc): object identifier

DISPLAY OBJECT

| 'GKSM 128' | L | I |

Turn on visibility of object I
I(Nc): object identifier

DISAPPEAR OBJECT

| 'GKSM 129' | L | I |

Turn off visibility of object I
I(Nc): object identifier

EXPOSE OBJECT

| 'GKSM 130' | L | I |

Redisplay object I on top of any overlapping objects
I(c): object identifier

HIDE OBJECT

| 'GKSM 131' | L | I |

Redisplay object I behind any overlapping objects
I(c): object identifier

SELECT COMPONENT

    | 'GKSM 132' | L | I | P |

    Select component P of object I
    I(c): object identifier
    P(i): pick id of component
    This is used to select a group of output primitives
    identified by P in a segment associated with I.

ERASE COMPONENT

    | 'GKSM 133' | L | I | P |

    Erase component P of object I
    I(c): object identifier
    P(i): pick id of component

    This erases a group of output primitives identified by P in
    a segment associated with I. This element can be used only
    within a REDEFINE OBJECT sequence.

Items for Normalization Transformation

SET WINDOW

    | 'GKSM 134' | L | W |

    Define boundaries of world window for normalization
    transformation.
    W(4r): limits of world window (XMIN, XMAX, YMIN, YMAX)

SET VIEWPORT

    | 'GKSM 135' | L | V |

    Define boundaries of NDC viewport for normalization
    transformation.
    V(4r): limits of NDC viewport (XMIN, XMAX, YMIN, YMAX)

Items for Other Operations

ABORT

| 'GKSM 136' | L |

Abort ongoing operation transmitted in PIGCF stream. This
provides the means to abort unwanted or erroneous
operations. Only the innermost operation of a nested
sequence is aborted; successive aborts can be used to get
out of several levels of operation nesting.

POINTER TRACKING

| 'GKSM 137' | L | T | P |

Update graphical pointer position to P
T(i): 0 causes only cursor to be moved
       1 causes cursor movement to be traced with
       a line
P(p): a point sampled from graphical pointer
       movement trace

RUBBER BAND

    | 'GKSM 138' | L | T | P |

    Echo a rubber band of type T with given reference and
    feedback points. The first occurrence of this item in a
    sequence carries the coordinates of the echo reference
    point. Subsequent occurrences carry updates to a pointer
    position indicating an echo feedback point.

    T(i): echo type
            (0 echo reference point;
            > 0 echo feedback:
              1 = line,
              2 = rectangle,
              3 = circle)
    P(r): echo reference point (T = 0),
           or echo feedback point (T > 0)

       The reference and feedback points are:
           T = 1 - reference is one end of line, feedback is
                   other end.
           T = 2 - reference is one corner of rectangle, feedback
                   is opposite corner.
           T = 3 - reference is center of circle, feedback is
                   perimeter point.

RECALL LIBRARY

    | 'GKSM 139' | L | F |

    Recall graphical library in file F
    F(i): name of file containing library

    The graphical pictures in F and all their components become
    available for use during the communication session. The
    pictures are assumed to be recorded with the PIGCF, and
    their components have to be displayed with DISPLAY OBJECT
    elements or similar actions so that the pictures become
    visible.

V. AN ARCHITECTURE FOR PIGCF PROCESSING

   This section presents an example software architecture for the
   generation and interpretation of PIGCF in a multimedia conferencing
   system using GKS as the underlying programmer's graphics interface.
   This section should not be interpreted as a definitive statement of
   such an architecture, but only as an exercise to illustrate how the
   format proposed in this paper fits within the overall framework of a
   conferencing system. Choosing GKS simplifies the example
   architecture; nevertheless, other graphics packages can be used by
   adding, to the architecture, the modules to interpret and generate
   the PIGCF level L items.

   Figure 1 shows the major software modules charged with graphics
   interaction and display at a conferencing workstation. This is a
   familiar programmer's view of the graphics pipeline. A conferencing
   application program updates data structures and uses
   device-independent graphics services through a language binding.
   These services, in turn, use device-dependent graphics services that
   call on device drivers to accept input and to present graphic
   pictures. The application performs numerous other functions for
   conference management and control of other media streams, but we need
   not consider them in this example.

   In Figure 2, the basic graphics pipeline has been augmented with the
   software modules involved in the generation, transmission, reception,
   and interpretation of PIGCF streams. The application has a module for
   interpreting the lower and higher levels of PIGCF and one for
   generating the upper level U. The device-independent graphics
   services include modules for generating and interpreting the lower
   level, L. This reflects the current practice of including the
   generation and interpretation functions in the graphics package.
   There is also a module that transmits the outgoing PIGCF streams to
   remote work stations. Similarly, there is a module that receives
   incoming streams from remote stations. In actual practice, the
   transmit and receive modules are decomposed into several processes
   implementing a layered protocol architecture. A process receives both
   levels of PIGCF and writes them into a conference record metafile for
   future use. A router process receives and forwards PIGCF traffic from
   and to the modules previously referred. This router is likely to be
   replaced by independent communication interfaces between pairs of
   modules exchanging PIGCF.

   The thick arrows show the flow of outgoing PIGCF, whereas the thin
   arrows show the incoming PIGCF flow. We first follow the outgoing
   path, starting at the application. The application processes local
   user actions which are transformed into data structure updates, level

U PIGCF elements, and executions of device independent graphics
subroutines that, among other things, generate level L PIGCF (GKSM)
elements.

The router merges both level streams according to generation order
and sends them to the local copy of the conference record and to the
transmission module. The latter batches Group-2 PIGCF items until it
receives a Group-1 item. It also timestamps the PIGCF stream to
synchronize its play-back, at the receiver, with the play-back of
other media information. The PIGCF may be separated into traffic
categories transmitted over diverse communication facilities
according to the transport services required by the categories, for
example, real-time service for pointer updates, highly reliable
transmission for new object definitions, or low-priority service for
graphical library transfers. Finally, the transmit module must
acknowledge the reception of incoming PIGCF, and of other media
traffic as well.

The receive module is the entry point for incoming PIGCF streams that
may come within diverse traffic categories requiring merging. It
checks the timestamps for synchronizing PIGCF items with related data
in other media, for example, voice. It is possible to include here a
high-level error-correction function that validates the received
streams using state and context information about PIGCF syntax and
semantics. The receive module passes the streams to the router which
forwards them to three processes: It sends level L items to the GKSM
interpreter which produces the corresponding changes on the displayed
picture; it sends level L and level U items to the conference record,
as well as to the PIGCF interpretation code in the application. The
level U items cause updates to both the data structures modeling
object hierarchies, and the pictorial representation of the
hierarchies, through the execution of graphics services. U items also
update graphics cursors and may recall new graphics libraries. The
application must process level L items because they could indicate
updates to the data structures; this happens if, for example, the
structures record attribute value information for the object
hierarchies. The application coordinates these actions with other
media effects according to the timestamps. Conference record
play-back is done in off-line mode. Record items are received by the
router and thereafter processed similarly to incoming PIGCF.

```
        +------------+              +------------+
        |APPLICATION |              |   OTHER    |
        |   DATA     |              |   MEDIA    |
        |STRUCTURES  |              |------------|
        +-----|------+              | CONFERENCE |
              |---------->          | APPLICATION|
                                    |  GRAPHICS  |
              |---------->          |            |
        +-----|------+              |            |
        | LANGUAGE   |              +------------+
        | BINDING    |
        +-----|------+              +------------+
              |---------->          |  DEVICE-   |
        +------------+              | INDEPENDENT|
        |  DEVICE    |              |  GRAPHICS  |
        | DEPENDENT  |   <--->      |  SERVICES  |
        | GRAPHICS   |              |            |
        | SERVICES   |              |            |
        +-----|------+              |            |
              |                     |            |
              v                     |            |
        +------------+              |            |
        |   DEVICE   |              |            |
        |  DRIVERS   |              |            |
        +------------+              +------------+
```
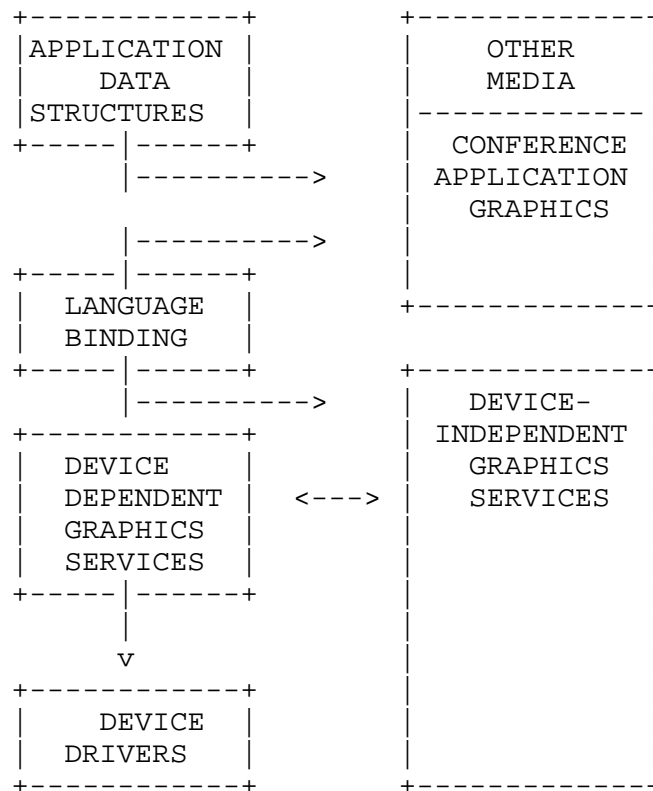
        FIGURE 1 - THE BASIC GRAPHICS PIPELINE
                IN A CONFERENCING SYSTEM

```
+-----------+     +-----------+                    +-----------------+
|APPLICATION|     |   OTHER   |                    |    TRANSMIT     |
|   DATA    |     |   MEDIA   |                    |      ACK        |=>
| STRUCTURES|     |-----------|      +-----+       | SEPARATE TRAFFIC|=>
+-----|-----+     |CONFERENCE |      |     |===>   |    BATCHING     |=>
      |---------->|APPLICATION|      |     |       |   TIMESTAMPING  |
      |           |  GRAPHICS |      |     |       +-----------------+
      |---------->|-----------|      |     |
      |           | PIGCF L, U| <---|      |       +-----------------+
+-----|-----+     |INTERPRETER|      |     |       |     RECEIVE     |
| LANGUAGE  |     +-----------+      | R   |       |  MERGE TRAFFIC  |<-
| BINDING   |     | PIGCF U   |===>  | O   | <---| CHECK TIMESTAMPS |<-
+-----|-----+     | GENERATOR |      | U   |       | ERROR CORRECTION|<-
      |           +-----------+      | T   |       |                 |
  -----------------|              | E   |       +-----------------+
+-----------+     +-----V-----+      | R   |
|  DEVICE   |     |  DEVICE   |      |     |       +-----------------+
| DEPENDENT |     |INDEPENDENT|      |     |====>|                 |
| GRAPHICS  |<-->|  GRAPHICS |      |     |---->|   CONFERENCE    |
| SERVICES  |     | SERVICES  |      |     |       |     RECORD      |
|           |     |           |      |     |       |                 |
+-----|-----+     |-----------|      |     |       +-----------------+
      |           |   GKSM    |      |     |
      v           |INTERPRETER|<---  |     |       <--- INCOMING PIGCF
+-----------+     +-----------+      |     |
|  DEVICE   |     |   GKSM    |      |     |       ===> OUTGOING PIGCF
|  DRIVERS  |     | GENERATOR |===>  |     |
+-----------+     +-----------+     +-----+
```
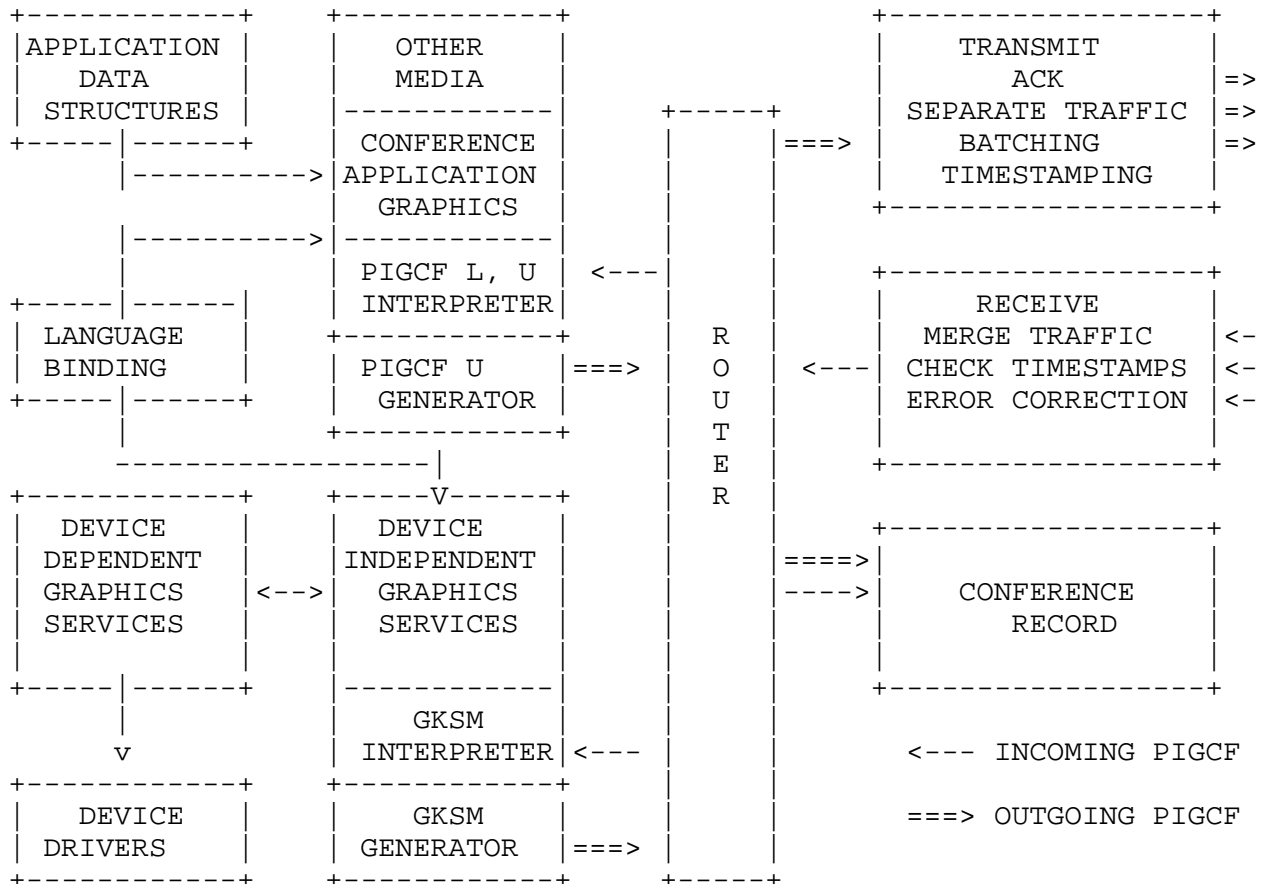
FIGURE 2 - A CONFERENCING SOFTWARE ARCHITECTURE FOR PROCESSING PIGCF

VI. CONCLUSIONS

     Teleconferencing and other multi-media applications will be part of
     the communication resources available to organizations in the near
     future. This will prompt computer graphics and computer communication
     practitioners to address the issue of application-to-application
     graphics communication. A key element of the issue is a protocol, and
     a key component of the protocol is a data format. We have presented
     the operational requirements for such a protocol and have proposed a
     format that fulfills these requirements.

     At present, none of the existing or emerging graphics standards can
     be used as the needed protocol or as a format for the protocol, but
     this may change as the standards evolve. We are monitoring the
     standards development and will study the use of some of them as a
     format basis, in particular the CGI. Nevertheless, the computer

communication community badly needs experience with multi-media
conferencing implementations. In order for these applications to
happen, one can base a graphics communication protocol on an official
or on a de-facto standard that is likely to gain wide use thus
assuring interoperability with a broad user base. We believe that,
by using the GKSM session metafile, we are moving in the proper
direction.

Planning the software architecture for generating and interpreting
the proposed PIGCF has brought up some problems we will confront as
we continue our work toward the development of a complete graphics
protocol. This is being done as part of the SRI on-going program in
multimedia communications. Within this program, we are implementing
a simple multi-media conferencing prototype and will design a more
complete one. The experience from both exercises will be a valuable
input to the protocol architecture design.

APPENDIX A

   Excerpt from "Draft Proposal: Graphical Kernel System" [14]

   E.2 Metafile Based on ISO DIS7942

      This metafile may be categorized as one which aims to provide a
      means of recording the exact sequence of function calls made to
      GKS. Its functional capability covers the entire range of GKS
      output functions, from level m to level 2. It is, therefore,
      suitable for applications where the individual graphics actions
      need to be 'played back', perhaps with selective graphical editing
      being done by the interpreter.

      Two encodings have been specified for this metafile. One encoding
      is inefficient for many applications. The second allows an
      unspecified binary format. The remainder of this IGCF appendix
      gives full details of these metafile structures and encodings.

      E.2.1 File Format and Data Format

         The GKS metafile is built up as a sequence of logical data
         items. The file starts with a file header in fixed format which
         describes the origin of the metafile (author, installation),
         the format of the following items, and the number
         representation. The file ends with an end item indicating the
         logical end of the file. In between these two items, the
         following information is recorded in the sense of an audit
         trail:

            a)   workstation control items and message items;

            b)   output primitive items, describing elementary
                    graphics objects;

            c)   attribute information, including output primitive
                    attributes; segment attributes, and workstation
                    attributes;

            d)   segment items, describing the segment structure and
                    dynamic segment manipulations;

            e)   user items.

The overall structure of the GKS metafile is as follows:

```
FILE:        |file  |item|---|item|---|end |
             |header| 1  |   | i  |   |item|

ITEM:        |item     |item data record|
             |header   |                |

ITEM         |'GKSM'  |identification|length of item data|

HEADER:  |optional|    number    |      in bytes      |
```

All data items except the file header have an item header
containing:

a)   the character string 'GKSM' (optional) which is
        present to improve legibility of the file and to
        provide an error control facility;

b)   the item type identification number which indicates
        the kind of information that is contained in the
        item;

c)   the length of the item data record.

The lengths of these fields of the item header are
implementation dependent and are specified in the file header.
The content of the item data record is fully described below
for each item type.

The metafile contains characters, integer numbers, and real
numbers marked (c), (i), (r) in the item description.
Characters in the metafile are represented according to ISO 646
and ISO 2022. Numbers will be represented according to ISO 6093
using format F1 for integers and format F2 for reals. (Remark:
Formats F1 and F2 can be written and read via FORTRAN formats I
and F respectively.)

Real numbers describing coordinates and length units are stored
as normalized device coordinates. The workstation
transformation, if specified in the application program for a
workstation writing a metafile of this format, is not performed
but WORKSTATION WINDOW and WORKSTATION VIEWPORT are stored in
data items for later usage. Real numbers may be stored as
integers. In this case transformation parameters are specified
in the file header to allow proper transformation of integers
into normalized device coordinates.

For reasons of economy, numbers can be stored using an internal
binary format. As no standard exists for binary number
representation, this format limits the portability of the
metafile. The specification of such a binary number
representation is outside the scope of this document.

When exchanging metafiles between different installations, the
physical structure of data sets on specific storage media
should be standardized. Such a definition is outside the scope
of this standard.

E.3 Generation of Metafiles

Table E1 contains a list, by class, of all GKS functions which
apply to workstations of category MO, and their effects on this
GKSM. In the table, GKSM-OUT is a workstation identifier
indicating a workstation writing a metafile of this format.

The concepts of clipping rectangle and clipping indicator are
encapsulated in one metafile item which specifies a clipping
rectangle. This item is written to the metafile on activate
workstation with the values (0, 1, 0, 1), if the clipping
indicator is OFF, or the viewport of the current normalization
transformation, if the clipping indicator is ON. If the viewport
of the current normalization transformation is redefined or a
different normalization transformation is selected when the
clipping indicator is ON, a further clipping rectangle item is
written. If the clipping indicator is changed to OFF, a clipping
rectangle item (0, 1, 0, 1) is written. If the clipping indicator
is changed to ON, an item containing the viewport of the current
normalization transformation is written. This is analogous to the
handling of clipping in segments (see 4.7.6 [14]).

GKS functions which apply to workstations   GKSM item created
of category MO   or effect
===

Control functions

OPEN WORKSTATION (GKSM-OUT,...)   - (file header)
                                                   1 (CONDITIONAL)
CLOSE WORKSTATION (GKSM-OUT)   0 (end item)
ACTIVATE WORKSTATION (GKSM-OUT)   (61, 21-44)
                                                   ensure attributes
                                                   current;
                                                   enable output

DEACTIVATE WORKSTATION (GKSM-OUT)   disable output
CLEAR WORKSTATION (GKSM-OUT,...)   1
                                                   2
REDRAW ALL SEGMENTS ON WORKSTATION (GKSM-OUT)
UPDATE WORKSTATION (GKSM-OUT,...)   3
SET DEFERRAL STATE (GKSM-OUT,...)   4
MESSAGE (GKSM-OUT,...)   5 (message)
ESCAPE   6
_____

Output Primitives

POLYLINE   11
POLYMARKER   12
TEXT   13
FILL AREA   14
CELL ARRAY   15
GENERALIZED DRAWING PRIMITIVE   16
_____

Output Attributes

SET POLYLINE INDEX   21
SET LINETYPE   22
SET LINEWIDTH SCALE FACTOR   23
SET POLYLINE COLOUR INDEX   24
SET POLYMARKER INDEX   25
SET MARKER TYPE   26
SET MARKER SIZE SCALE FACTOR   27
SET POLYMARKER COLOUR INDEX   28
SET TEXT INDEX   29
SET TEXT FONT AND PRECISION   30
SET CHARACTER EXPANSION FACTOR   31
SET CHARACTER SPACING   32
SET TEXT COLOUR INDEX   33
SET CHARACTER HEIGHT   34
SET CHARACTER UP VECTOR   34
SET TEXT PATH   35
SET TEXT ALIGNMENT   36
SET FILL AREA INDEX   37
SET FILL AREA INTERIOR STYLE   38
SET FILL AREA STYLE INDEX   39
SET FILL AREA COLOUR INDEX   40

SET PATTERN SIZE   41
SET PATTERN REFERENCE POINT   42

SET ASPECT SOURCE FLAGS   43
SET PICK IDENTIFIER   44

_____

Workstation Attributes

SET POLYLINE REPRESENTATION (GKSM-OUT,...)   51
SET POLYMARKER REPRESENTATION (GKSM-OUT,...)   52
SET TEXT REPRESENTATION (GKSM-OUT,...)   53
SET FILL AREA REPRESENTATION (GKSM-OUT,...)   54
SET PATTERN REPRESENTATION (GKSM-OUT,...)   55
SET COLOUR REPRESENTATION (GKSM-OUT,...)   56

_____

Transformation Functions

SET WINDOW of current normalization   34, 41, 42
transformation
SET VIEWPOINT of current normalization   61, 34, 41, 42
transformation
SELECT NORMALIZATION TRANSFORMATION   61, 34, 41, 42
SET CLIPPING INDICATOR   61
SET WORKSTATION WINDOW (GKSM-OUT,...)   71
SET WORKSTATION WINDOW VIEWPORT (GKSM-OUT,...)   72

Note: item 61 (CLIPPING RECTANGLE) is described more fully in E.2.2.

Note: When the current normalization transformation is altered, items
corresponding to attributes containing coordinate information are sent
(items 34, 41, and 42).

_____

Segment Functions

CREATE SEGMENT   81
CLOSE SEGMENT   82
RENAME SEGMENT   83
DELETE SEGMENT   84

DELETE SEGMENT FROM WORKSTATION (GKSM-OUT,...)   84
ASSOCIATE SEGMENT WITH WORKSTATION   81, (21-44), (11-16),
(GKSM-OUT,...)   (61), 82
COPY SEGMENT TO WORKSTATION (GKSM-OUT,...)   (21-44), (11-16), (61)

INSERT SEGMENT   (21-44), (11-16), (61)

_____

Segment Attributes

SET SEGMENT TRANSFORMATION   91

SET VISIBILITY   92
SET HIGHLIGHTING   93
SET SEGMENT PRIORITY   94
SET DETECTABILITY   95

_____

Metafile Functions

WRITE ITEM TO GKSM   > 100

_____

   E.4   Interpretation of Metafiles

     E.4.1   Introduction

        The interpretation of metafiles in GKS is described in 4.9
        [14]. The effects of INTERPRET ITEM for all types of metafile
        item are described in the following sections. Items are grouped
        by class of functionality.

     E.4.2   Control Items

        Interpretation of items in this class is described under the
        definitions of each item in E.5. ([14] reads "E.2.4" instead of
        "E.5" which we believe is an error).

     E.4.3   Output Primitives

        Interpretation of items in this class generates output
        corresponding to the primitive functions, except that
        coordinates of points are expressed in NDC. Primitive
        attributes bound to primitives are those which have originated
        from interpretation of primitive attribute items in this
        particular metafile (see E.4.4).

     E.4.4   Output Primative Attributes

        Interpretation of items in this class sets values for use in
        the display of primitives subsequently originating from this
        particular metafile (see E.4.3). No changes are made to entries
        in the GKS state list.

E.4.5 Workstation Attributes

Interpretation of items in this class has the same effect as
invocation of the corresponding GKS functions shown in Table
E1. The GKS functions are performed on all active workstations.

E.4.6 Transformations

Interpretation of a clipping rectangle item sets values for use
in clipping output primitives subsequently originating from
this particular metafile. No changes are made to entries in the
GKS state list. Interpretation of other items in this class
(WORKSTATION WINDOW and WORKSTATION VIEWPORT) causes the
invocation of the corresponding GKS functions on all active
workstations.

E.4.7   Segment Manipulation

Interpretation of items in this class has the same effect as
invocation of the corresponding GKS functions shown in Table
E1. (Item 84 causes an invocation of DELETE SEGMENT.)

E.4.8 Segment Attributes

Interpretation of items in this class has the same effect as
invocation of the corresponding GKS functions shown in Table
E1.

E.5 Control Items

FILE HEADER

| GKSM | N | D | V | H | T | L | I | R | F | RI | ZERO | ONE |

All fields in the file header item have fixed length. Numbers are
formated according to ISO 6093 - Format F1.

General Information:

GKSM   4 bytes   containing string 'GKSM'
N   40 bytes   containing name of author/installation
D   8 bytes   date (year/month/day, e.g., 79/12/31)
V   2 bytes   version number: the metafile described here has
                   version number 1
H   2 bytes   integer specifying how many bytes of the string 'GKSM'
                   are repeated at the beginning of each record.
                   Possible values: 0, 1, 2, 3, 4

```
T        2 bytes    length of item type indicator field
L        2 bytes    length of item data record length indicator field
I        2 bytes    length of field for each integer in the
                    item data record (applied to all data marked (i)
                    in the item description)
R        2 bytes    length of field for each real in the item data record
                    (applies to all data marked (r) in the item
                    description).
```

Specification of Number Representation:

```
F        2 bytes    Possible values:  1, 2.  This applies to all data
                    in the items marked (i) or (r) and to item type
                    and item data record length:
                    1:  all numbers are formatted according to ISO 6093
                    2:  all numbers (except in the file header) are
                    stored in an internal binary format
RI       2 bytes    Possible values:  1, 2.  This is the number
                    representation for data marked (r):
                    1 = real, 2 = integer
ZERO     11 bytes   integer equivalent to 0.0, if RI = 2
ONE      11 bytes   integer equivalent to 1.0, if RI = 2
```

After the file header, which is in fixed format, all values in
the following items are in the format defined by the file
header. For the following description, the setting:

                   H = 4; T = 3; F = 1

is assumed. In addition to formats (c), (i) and (r), which are
already described, (p) denotes a point represented by a pair of
real numbers (2r). The notation allows the single letter to be
preceded by an expression, indicating the number of values of
that type.

{Explanatory comments have been added to some item
specifications; these are not part of the GKS Appendix E and
they are enclosed in braces {}. A complete definition of the
generation and interpretation of the GKSM items is given by the
definition of the corresponding GKS functions [14].}

  END ITEM

    | 'GKSM 0' | L |

  Last item of every GKS Metafile. Sets condition for the error.

CLEAR WORKSTATION

    | 'GKSM 1' | L | C |

    Requests CLEAR WORKSTATION on all active workstations.

    C(i): clearing control flag
           (0 = CONDITIONAL, 1 = ALWAYS)

REDRAW ALL SEGMENTS ON WORKSTATION

    | 'GKSM 3' | L | R |

    Requests UPDATE WORKSTATION on all active workstations.

    R(i): regeneration flag
           (0 = PERFORM, 1 = SUSPEND)

DEFERRAL STATE

    | 'GKSM 4' | L | D | R |

    Requests SET DEFERRAL STATE on all active workstations.

    D(i): deferral mode
          (0 = ASAP, 1 = BNIG, 2 = BNIL, 3 = ASTI)

    R(i): implicit regeneration mode
           (0 = ALLOWED, 1 = SUPPRESSED)

    {This item provides control over the occurrence of the visual
    effect of GKS functions in order to optimize the use of
    workstation capabilities according to application needs.}

MESSAGE

    | 'GKSM 5' | L | N | T |

    Requests MESSAGE on all active workstations.
    N(i):   number of characters in string
    T(Nc): string with N characters.

    {The message is not part of a metafile output primitives; the
    message is only for interpretation by workstation operators.}

        ESCAPE

            | 'GKSM 6' | L | FI | L | M | I | R |

            Requests ESCAPE

            FI(i):   function identifier
            L(i):   length of integer data in data record
            M(i):   length of real data in data record
            I(Li):   integer data
            R(Mr):   real data.

            {This item permits the invocation of a specific non-standard
            escape function FI. The execution of the function with the
            given parameters must not alter the GKS state list nor produce
            geometrical output.}

    E.6 Items for Output Primitives

        POLYLINE

            | 'GKSM 11' | L | N | P |

            N(i):   number of points of the polyline
            P(Np):   list of points

        POLYMARKER

            | 'GKSM 12' | L | N | P |

            N(i):   number of points
            P(Np):   list of points.

        TEXT

            | 'GKSM 13' | L | P | N | T |

            P(p):   starting point of character string
            N(i):   number of characters in string T
            T(Nc):   string with N characters from the set of ISO 646

        FILL AREA

            | 'GKSM 14' | L | N | P |

            N(i):   number of points
            P(Np):   list of points.

        CELL ARRAY

           | 'GKSM 15' | L | P | Q | R | N | M | CT |

           P(p),Q(p),R(p):   coordinates of corner points of pixel array
                             (P and Q are the images of the points P and
                             Q specified in the function CELL ARRAY and
                             R is another corner)
           M(i):   number of rows in array
           N(i):   number of columns in array
           CT(MNi):   array of colour indices stored row by row

           {This item permits passing raster images to GKS. The raster
           image is defined by the colour index matrix CT, and its World
           Coordinate position given by points P and Q.}

        GENERALIZED DRAWING PRIMITIVE

           | 'GKSM 16' | L | GI | N | P | L | M | I | R |

           GI(i): GDP identifier
           N(i):   number of points
           P(Np): list of points
           L(i):   length of integer data in data record
           M(i):   length of real data in data record
           I(Li): integer data
           R(Mr): real data.

           {This item provides a standard way for drawing additional
           non-standard output primitives. The generalized drawing
           primitive GI is drawn according to the point list P and the
           data record in I and R.}

      E.7 Items for Output Primitive Attributes

        POLYLINE INDEX

           | 'GKSM 21' | L | LT |

           LT(i): linetype

LINEWIDTH SCALE FACTOR

    | 'GKSM 23' | L | LW |

    LW(r): linewidth scale factor

    {In GKS, the line width is not affected by GKS transformations.
    However, the effective line width is calculated as the product
    of the nominal line width times the line width scale factor in
    effect when a line is drawn.}

POLYLINE COLOUR INDEX

    | 'GKSM 24' | L | CI |

    CI(i): polyline colour index

POLYMARKER INDEX

    | 'GKSM 25' | L | I |

    I(i): polymarker index

MARKER TYPE

    | 'GKSM 26' | L | MT |

    MT(i): marker type

MARKER SIZE SCALE FACTOR

    | 'GKSM 27' | L | MS |

    MS(r): marker size scale factor

    {In GKS, the marker size is not affected by GKS
    transformations. However, the effective marker size is
    calculated as the product of the nominal marker size times the
    marker size scale factor in effect when a marker is drawn.}

POLYMARKER COLOUR INDEX

    | 'GKSM 28' | L | CI |

    CI(i): polymarker colour index

TEXT INDEX

    | 'GKSM 29' | L | I |

    I(i): text index

TEXT FONT AND PRECISION

    | 'GKSM 30' | L | F | P |

    F(i): text font
    P(i): text precision
    (0 = STRING, 1 = CHAR, 2 = STROKE)

CHARACTER EXPANSION FACTOR

    | 'GKSM 31' | L | CEF |

    CEF(r): character expansion factor

    {This item allows the manipulation of the width/height of the
    character body. The width of the character body is scaled by
    the CEF factor.}

CHARACTER SPACING

    | 'GKSM 32' | L | CS |

    CS(r): character spacing

TEXT COLOUR INDEX

    | 'GKSM 33' | L | CI |

    CI(i): text colour index

CHARACTER VECTORS

    | 'GKSM 34' | L | CH | CW |

    CH(2r): character height vector
    CW(2r): character width vector

    Note: These vectors are the height and width vectors described
    in 4.4.5 of [14].

    {The character height vector is parallel to the character up
    vector and has a length equal to character height. The
    character height specifies the height of a capital letter. The
    character width vector is perpendicular to the height vector,
    in the direction of the character baseline, and has the same
    length.}

TEXT PATH

    | 'GKSM 35' | L | P |

    P(i): text path
    (0 = LEFT, 1 = RIGHT, 2 = UP, 3 = DOWN)

TEXT ALIGNMENT

    | 'GKSM 36' | L | H | V |

    H(i): horizontal character alignment
          (0 = NORMAL, 1 = LEFT, 2 = CENTRE, 3 = RIGHT)
    V(i): vertical character alignment
          (0 = NORMAL, 1 = TOP, 2 = CAP, 3 = HALF, 4 = BASE,
           5 = BOTTOM)

FILL AREA INDEX

    | 'GKSM 37' | L | I |

    I(i): fill area index

FILL AREA INTERIOR STYLE

    | 'GKSM 38' | L | S |

    S(i): fill area interior style
          (0 = HOLLOW, 1 = SOLID, 2 = PATTERN, 3 = HATCH)

FILL AREA STYLE INDEX

| 'GKSM 39' | L | SI |

SI(i): fill area style index

FILL AREA COLOUR INDEX

| 'GKSM 40' | L | CI |

CI(i): fill area colour index

PATTERN SIZE

| 'GKSM 41' | L | PW | PH |

PW(2r): pattern width vector
PH(2r): pattern height vector

{One style for filling areas is with a pattern of color cells.
Such a pattern is defined by an array of color indices which is
mapped into a pattern rectangle with dimensions given by PW and
PH.}

PATTERN REFERENCE POINT

| 'GKSM 42' | L | P |

P(p): reference point

{One style for filling areas is with a pattern of color cells.
Such a pattern is defined by an array of color indices which is
mapped into a pattern rectangle whose lower left corner is
given by P.}

ASPECT SOURCE FLAGS

    | 'GKSM 43' | L | F |

    F(13i): aspect source flags
             (0 = BUNDLED, 1 = INDIVIDUAL)

    {An application can set an output primitive attribute to either
    bundled or individual. Bundled attributes are
    workstation-dependent, their binding is delayed, and their
    values can change dynamically. Individual attributes are global
    attributes, they are bound immediately, and their value is
    static and cannot be manipulated.}

PICK IDENTIFIER

    | 'GKSM 44' | L | P |

    P(i): pick identifier

  E.8 Items for Workstation Attributes

    POLYLINE REPRESENTATION

       | 'GKSM 51' | L | I | LT | LW | CI |

       I(i):   polyline index
       LT(i): linetype number
       LW(r): linewidth scale factor
       CI(i): polyline colour index

    POLYMARKER REPRESENTATION

       | 'GKSM 52' | L | I | MT | MS | CI |

       I(i):   polymarker index
       MT(i): marker type
       MS(r): marker size scale factor
       CI(i): polymarker colour index

TEXT REPRESENTATION

    | 'GKSM 53' | L | I | F | P | CEF | CS | CI |

    I(i):   text index
    F(i):   text font
    P(i):   text precision
    (0 = STRING, 1 = CHAR, 2 = STROKE)
    CEF(r): character expansion factor
    CS(r):   character spacing
    CI(i):   text colour index

FILL AREA REPRESENTATION

    | 'GKSM 54' | L | I | S | SI | CI |

    I(i):   fill area index
    S(i):   fill area interior style
    (0 = HOLLOW, 1 = SOLID, 2 = PATTERN, 3 = HATCH) SI(i): fill
    area style index
    CI(i): fill area colour index

PATTERN REPRESENTATION

    | 'GKSM 55' | L | I | N | M | CT |

    I(i):   pattern index
    N(i):   number of columns in array*
    M(i):   number of rows in array
    CT(MNi): table of colour indices stores row by row

       {* The ANSI document reads "area" instead of "array".}

    {One style for filling areas is with a pattern of color cells.
    Such a pattern is defined by a pattern representation.}

COLOUR REPRESENTATION

    | 'GKSM 56' | L | CI | RGB |

    CI(i):   colour index
    RGB(3r): red, green, blue intensities

   E.9 Items for Transformations

      CLIPPING RECTANGLE

         | 'GKSM 61' | L | C |

         C(4r): limits of clipping rectangle (XMIN, XMAX, YMIN, YMAX)

      WORKSTATION WINDOW

         | 'GKSM 71' | L | W |

         W(4r): limits of workstation window (XMIN, XMAX, YMIN, YMAX)

         {GKS includes a workstation transformation that maps a
         rectangle of the NDC space (a workstation window) into a
         rectangle of the device coordinate space (a workstation
         viewport).}

      WORKSTATION VIEWPORT

         | 'GKSM 72' | L | V |

         V(4r): limits of workstation viewport (XMIN, XMAX, YMIN, YMAX)

   E.10 Items for Segment Manipulation

      CREATE SEGMENT

         | 'GKSM 81' | L | S |

         S(i): segment name

      CLOSE SEGMENT

         | 'GKSM 82' | L |

         indicates end of segment

      RENAME SEGMENT

         | 'GKSM 83' | L | SO | SN |

         SO(i): old segment name
         SN(i): new segment name

         DELETE SEGMENT

              | 'GKSM 84' | L | S |

              S(i):   segment name

      E.11 Items for Segment Attributes

         SET SEGMENT TRANSFORMATION

              | 'GKSM 91' | L | S | M |

              S(i):   segment name
              M(6r):   transformation matrix
                       upper and center rows of a 3x3 matrix representing
                       a 2D homogeneous transformation [9]
                       M 11 M 12 M 13 M 21 M 22 M 23

              {This differs from the ANSI X3.124 Jan. 5 1984 document, in the
              matrix elements indicated. We believe there is an error in such
              document.}

         SET VISIBILITY

              | 'GKSM 92' | L | S | V |

              S(i):   segment name
              V(i):   visibility
                      (0 = VISIBLE, 1 = INVISIBLE)

         SET HIGHLIGHTING

              | 'GKSM 93' | L | S | H |

              S(i):   segment name
              H(i):   highlighting
                      (0 = NORMAL, 1 = HIGHLIGHTED)

         SET SEGMENT PRIORITY

              | 'GKSM 94' | L | S | P |

              S(i):   segment name
              P(r):   segment priority

     SET DETECTABILITY

        | 'GKSM 95' | L | S | D |

        S(i):   segment name
        D(i):   detectability
                (0 = UNDETECTABLE, 1 = DETECTABLE)

   E.12 User Items

     USER ITEM

        | 'GKSMXXX' | L | D |

        XXX   > 100
        D: user data (L bytes)

        {The PIGCF level U items are encoded as GKSM USER ITEM elements
        so that a PIGCF file will conform to the GKSM metafile
        specification.}

   APPENDIX B

   Example of PIGCF Use in Conferencing

   This section presents an example illustrating the proposed PIGCF
   graphical component in an audio-graphics conference exchange. We
   present only the graphical part of the conference exchange, which
   actually would be complemented with speech. For the sake of briefness
   the example does not contain all the parameter negotiation that a
   conference set-up would require.

   The example is about an on-line audio-graphics conference between a
   Navy command and control center and a Navy task force. The PIGCF
   items shown do not belong to a single transmission stream. The stream
   they belong to is determined by the station that transmits them, and
   the identification of the transmitter belongs to lower level
   communication protocols. We use the character encoding, rather than
   the binary one, for this PIGCF example. We illustrate just a few of
   the possible groups of items that could be batched in this example.
   The plot of the example is as follows.

   The command center (center) establishes a conference with some ships
   in a task force (platforms) to coordinate the interception of an
   unidentified ship that has been sighted in a conflict area. After
   recalling graphical libraries, all conference sites can see in their
   screens a map of the sighting area as well as iconic representations
   of the task force ships. Then the center interactively draws an
   iconic representation of the unidentified vessel, scales it, and
   places it in the sighting location.

   The platforms explain possible courses of action using graphical
   pointers. The center draws the expected trajectory of the
   unidentified ship and the platforms situate the task force icons at
   the expected points of interception. Then the center zooms into the
   interception area and the platforms use rubber bands to discuss
   interception maneuvers.

   Now we proceed to list the PIGCF items exchanged. The center
   initiates the conference graphical set-up with the FILE HEADER item
   to set basic representation parameters for the graphical
   information to be exchanged.   This item can be interpreted
   according to its definition in E.5 [14]. The most important
   parameter selections for this example are:

      i)   The items contain 0 characters of the "GKSM" string in the
            identification field of the item header.
      ii)   The item type indicator field containing the PIGCF

            item number is three bytes long in each item.
    iii) The integers are 4 bytes long, and the reals 6 bytes long.
    iv) The item data record length indicator is 2 bytes long.

We will obey the PIGCF specification field lengths and the aforesaid
field length settings. However, we will add one space before and
after the "|" separator to improve legibility. Also, every item will
be preceded with its name to help identification.

FILE HEADER:

    | GKSM | center | 84/11/10 | 1 | 0 | 3 | 2 | 4 | 6 | 1 | 1
    | | |

The center states the boundaries of the work station window for the
conference.

WORKSTATION WINDOW: | 71 | 24 | 0.0 0.5 0.0 0.375 |

In this example, we assume that the conferencing work stations use
world coordinates for the internal representation of positional
information. Accordingly, the center states the boundaries of the
world window for the normalization transformation used in the
conference.

SET WINDOW: | 134 | 28 | 0.0 320.0 0.0 240.0 |

The center informs the location of its local NDC viewport, however,
other conferees can choose different NDC viewports for the same
transformation, but their work station window should include the
conference's. All systems record the conference: world window, NDC
viewport, and work station widow.

SET VIEWPORT: | 135 | 28 | 0.0 0.5 0.0 0.375 |

The center recalls graphical libraries containing geographical maps
of the crisis area and icons of the task forces in the area. It
also displays a graphical object that provides a background picture.

RECALL LIBRARY: | 139 | 9 | caribbean |
DISPLAY OBJECT: | 128 | 11 | coast_lines |
RECALL LIBRARY: | 139 | 10 | task_units |

The center proceeds to instantiate one of the task forces in the
task_units library. This is done by recalling some of the library
objects and applying transformations to the objects, later. Since set
window, set viewport, and recall library belong to the update

Group-2, they can be batched until display object, from update
Group-1, is entered. The second recall library can be batched
together with the following begin instantiation until display object
is produced. The rest of the example contains more cases of item
sequences which can be batched; however, for briefness we do not
indicate any more of them.

```
BEGIN INSTANTIATION:    | 124 | 15 | US_CONSTITUTION |
DISPLAY OBJECT:         | 128 | 15 | US_CONSTITUTION |
TRANSFORM OBJECT:       | 126 | 55 |    15 | US_CONSTITUTION |
                          0.1   0.0   0.0   0.0   0.1   0.0 |
TRANSFORM OBJECT:       | 126 | 55 |    15 | US_CONSTITUTION |
                          0.1   0.0  0.312  0.0   0.1  0.078 |
END INSTANTIATION:      | 125 |  0 |

BEGIN INSTANTIATION:    | 124 | 13 | US_NEW_JERSEY |
DISPLAY OBJECT:         | 128 | 13 | US_NEW_JERSEY |
TRANSFORM OBJECT:       | 126 | 53 |    13 | US_NEW_JERSEY |
                          0.1   0.0   0.0   0.0   0.1   0.0 |
TRANSFORM OBJECT:       | 126 | 53 |    13 | US_NEW_JERSEY |
                          0.1   0.0  0.312  0.0   0.1  0.093 |
END INSTANTIATION:      | 125 |  0 |
```

Next the center sets values for two output primitive attributes in
preparation for drawing a new icon on the screens. We assume that all
the other attributes have been assigned default values as a result of
the conference set-up.

```
POLYLINE INDEX:         | 21 | 4 |  20 |
POLYLINE COLOUR INDEX:  | 24 | 4 | 200 |
```

The following items correspond to the interactive definition of the
unidentified vessel. Since the definition is done interactively, the
vessel image remains visible on the screens after definition.

```
BEGIN DEFINITION:  | 120 |  0 |
POLYLINE:          | 11 | 64 |    5 |
0.047  0.063  0.063  0.047  0.125  0.047  0.14  0.063  0.047  0.047 |
POLYLINE:          | 11 | 52 |    3 |
            0.078 0.063  0.078  0.078  0.109  0.078  0.109  0.063 |
END DEFINITION:    | 121 |  8 | sighting |
```

Then the unidentified vessel "sighting" is scaled and placed at the
sighting site.

```
BEGIN INSTANTIATION:    | 124 |  8 | sighting |
TRANSFORM OBJECT:       | 126 | 48 |    8 | sighting |
                          0.2   0.0   0.0
                          0.0   0.2   0.0 |
TRANSFORM OBJECT:       | 126 | 48 |    8 | sighting |
                          0.1   0.0 0.156
                          0.0   0.1  0.016 |
END INSTANTIATION:      | 125 |  0 |
```

The center and the platforms use graphical pointer movements to
discuss possible routes the unidentified vessel might follow. We only
show a few pointer updates. In practice, there would typically be a
large number of points transmitted to convey the movement of the
pointers over the screens.

from the center:

```
POINTER TRACKING:  | 137 | 16 |    0 |  0.39  0.032 |
POINTER TRACKING:  | 137 | 16 |    0 |  0.388 0.035 |
POINTER TRACKING:  | 137 | 16 |    0 |  0.388 0.039 |
POINTER TRACKING:  | 137 | 16 |    0 |  0.386 0.04  |
```

from one of the platforms:

```
POINTER TRACKING:  | 137 | 16 |    0 |  0.22  0.016 |
POINTER TRACKING:  | 137 | 16 |    0 |  0.222 0.159 |
POINTER TRACKING:  | 137 | 16 |    0 |  0.233 0.157 |
POINTER TRACKING:  | 137 | 16 |    0 |  0.24  0.155 |
```

The center now draws the expected route to be followed by the
unidentified ship. This time the pointer trace is recorded on the
screen by drawing a line.

```
POINTER TRACKING:  | 137 | 16 |    1 |  0.388 0.038 |
POINTER TRACKING:  | 137 | 16 |    1 |  0.386 0.038 |
POINTER TRACKING:  | 137 | 16 |    1 |  0.386 0.052 |
POINTER TRACKING:  | 137 | 16 |    1 |  0.375 0.078 |
POINTER TRACKING:  | 137 | 16 |    1 |  0.369 0.105 |
POINTER TRACKING:  | 137 | 16 |    1 |  0.361 0.125 |
POINTER TRACKING:  | 137 | 16 |    1 |  0.352 0.144 |
POINTER TRACKING:  | 137 | 16 |    1 |  0.351 0.156 |
POINTER TRACKING:  | 137 | 16 |    1 |  0.35  0.16  |
```

A platform moves the two US ship icons to interception positions.

```
   TRANSFORM OBJECT:   | 126 | 55 |    15 | US_CONSTITUTION |
                         1.0   0.0 0.16
                         0.0   1.0 -0.046 |
   TRANSFORM OBJECT:   | 126 | 53 |    13 | US_NEW_JERSEY |
                         1.0   0.0 0.113
                         0.0   1.0 -0.034 |
```

The center zooms into the interception area in order to obtain a
larger view for further discussion.

```
   WORKSTATION WINDOW:  |  71 | 24 | 0.286 0.403 0.077 0.177 |
```

The two platforms indicate their striking ranges using circular
rubber bands centered at each ship. For each platform, we show first
the echo reference point and then two echo feedback points. Typically
there will be a large number of feedback points.

```
   RUBBER BAND:  | 138 | 10 |    0 | 0.335 0.125 |
   RUBBER BAND:  | 138 | 10 |    3 | 0.35  0.128 |
   RUBBER BAND:  | 138 | 10 |    3 | 0.37  0.128 |

   RUBBER BAND:  | 138 | 10 |    0 | 0.384 0.13  |
   RUBBER BAND:  | 138 | 10 |    3 | 0.367 0.128 |
   RUBBER BAND:  | 138 | 10 |    3 | 0.346 0.129 |
```

Once the interception strategy has been agreed upon, the center zooms
out to the original, larger picture.

```
   WORKSTATION WINDOW:  |  71 | 24 |   0.0   0.5   0.0 0.375 |
```

The center terminates the conference

```
   END ITEM:  |   0 |  0 |
```

At the end of a conference, the final pictures remain visible on the
screens. In addition, the PIGCF items will be recorded in its
entirety in order to play back the conference session if necessary.
The conference record could also be sent to other locations as part
of a multi-media message.

REFERENCES

    [1]   J. D. Day and H. Zimmermann, "The OSI Reference Model",
           Proceedings of the IEEE, V 71, N 12; Dec. 1983, pp 1334-1340.

    [2]   W. Pferd, L. A. Peralta and F. X. Prendergast, "Interactive
           Graphics Teleconferencing", IEEE Computer, V 12, N 11; Nov.
           1979, pp 62-72.

    [3]   K. S. Sarin, "Interactive On-Line Conferences", Ph.D. Diss.
           MIT, Dept. of EE and CS, 1984.

    [4]   S. Randall, "The Shared Graphic Workspace: Interactive Data
           Sharing in a Teleconference Environment", Proceedings CompCon
           82 Fall, IEEE Computer Society, pp 535-542.

    [5]   G. Heffron, "Teleconferencing Comes of Age", IEEE Spectrum,
           Oct. 1984, pp 61-66, pp 61-66.

    [6]   R. W. Hough and R. R. Panko, "Teleconferencing Systems: A
           State-of-the-Art Survey and Preliminary Analysis", SRI
           International, Menlo Park California, SRI project 3735, April
           1977.

    [7]   C. W. Kelly III, "An Enhanced Presence Video Teleconferencing
           System" Proc. CompCon 1982, Sept. 20-23 Washington D.C., pp
           544-551.

    [8]   J. Vanglian, "Private Communication", Comments on the
           suitability of videotex for on-line graphical communication.

    [9]   ANSI Technical Committee X3H, "Draft Proposal: Virtual Device
           Metafile", X3.122, X3 Secretariat, CBEMA, Washington, D.C.

    [10]   American National Standards Committee X3H3, "Virtual Device
           Interface", X3 - Information Processing Systems, Working
           Document, Jan. 2, 1985 Available from Computer and Business
           Equipment Manufacturers Association, Washington D.C.

    [11]   E. Van Deusen, "Graphics Standards Handbook", CC Exchange 1984,
           P.O. Box 1251, Laguna Beach, CA 92652.

    [12]   J. D. Foley and A. Van Dam, "Fundamentals of Interactive
           Computer Graphics", Addison-Wesley, 1982.

    [13] American National Standards Committee X3H3, "GKS -- 3D
          Extensions", X3 - Information Processing Systems, Working
          Document, Nov. 16 1984 Available from Computer and Business
          Equipment Manufacturers Association, Washington D.C.

    [14] ANSI Technical Committee X3H3, "Draft Proposal: Graphical
          Kernel System", X3.124, X3 Secretariat, CBEMA, Washington, D.C.

    [15] G. Enderle, K. Kansy, and G. Pfaff, "Computer Graphics
          Programming", Springer-Verlag, 1984.

    [16] International Organization for Standardization "Information
          processing - Representation of numerical values in character
          strings for information interchange", ISO/DIS 6093.2, ISO/TC
          97, 1984-01-19; available from ANSI, New York, N.Y.