stichting
mathematisch
centrum

MC

AFDELING INFORMATICA
(DEPARTMENT OF COMPUTER SCIENCE)

IW 164/81 APRIL

P.J.W. TEN HAGEN

THE GKS REVIEWING PROCESS

Preprint

kruislaan 413 1098 SJ amsterdam

Printed at the Mathematical Centre, 413 Kruislaan, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

1980 Mathematics subject classification: 68B99

ACM Computer Review Categories: 8.2, 1.2

The GKS Reviewing Process*)

by

P.J.W. ten Hagen

ABSTRACT

This paper describes the technical debate within ISO on GKS. The specific ISO-frame for the organisation technical work is outlined. The major technical results of the debate are described with emphasis on newly developed concepts.

KEY WORDS & PHRASES: *Computer graphics, graphics standard, portability, device independence*

---

*) This report will be submitted for publication elsewhere.

1. INTRODUCTION

This paper discusses the technical debate within ISO (International Standards Organisation) about GKS (Graphical Kernel System). The purpose of such a debate is to improve and change successive versions of GKS until, finally, a version is obtained which satisfies the international community (including the originators).

The aspects of the debate that will be discussed in this paper all concern criteria for international acceptance of the standard. The discussion on GKS takes place within a framework of rules and procedures which gradually become more tight as GKS processes through the various stages defined for an ISO-project. ISO works with the principle that a standard must be acceptable to *every* member. It allows very little deviation from this rule. Most of the formal framework exists to ensure that every ISO member interested is indeed satisfied with the result.

The international group of technical experts responsible for the review of GKS is called ISO/TC97/SC5/WG2 Graphics. This name reveals the ISO hierarchy. Each level in the hierarchy may give rules, tasks and limits to lower level bodies. WG2 (working group 2) is responsible for all aspects of computer graphics within the scope of SC5 (subcommittee 5). SC 5 on behalf of TC 97 (technical committee 97), covers all aspects of programming languages. TC 97 is the ISO committee for the whole area of Data Processing. As is often seen in hierarchical organizations, the hard technical work is done at the lowest level.

The ISO directives give outlines for the organisation of the technical work. Basically they define how a particular standardization project may start, how progress is to be measured and in what way a project may finish (either successful or unsuccessful). The GKS reviewing process is to be understood against this background of ISO and lower level rules.

A reason for further structuring the debate originates within the working group itself. Procedures are fixed to ensure steady progress. The fact that such procedures are established is in itself a sign of progress.

They prove that the working group has found an effective way of producing a proposal as well as a way to convince the ISO hierarchy. From a technical point of view these procedures have some relevance, which is why they are mentioned here.

Apart from the primary goal of the reviewing process, namely the GKS standard, a document containing *principles* and *concepts* is produced, which provides the motivation for each feature of GKS.

The ISO format for technical work identifies a number of stages called *milestones* as follows:

0. exploration: A certain area (e.g., graphics) has been identified as a promising area for standards (i.e., there is a need and there are opportunities).

1. work item: within the area a subject for standardization has been accepted (e.g., GKS),

2. working draft: a proposal has been accepted as a basis for a standard. A member body has accepted the responsibility for further development.

3. draft proposal: a working draft is sufficiently complete to be a candidate for a standard.

4. draft international standard: all technical objections against a draft proposal have been resolved or removed. The draft standard is compatible with already existing ISO standards.

5. international standard: the draft is officially registered as an ISO standard and is available for distribution (in English, French and (sometimes) Russian).

For each milestone, agreement within the body concerned is required. Until milestone 3 the work is only technical. Thereafter, formalities play their part. These later stages, for the working group, are not so important. In practice, stage 3 means that the international community already starts using the standard.

At this moment (early 1981) GKS has passed milestone 1, and is about to move past milestone 2. The remainder of this paper describes what technical problems had to be solved in order to reach this stage. In addition, the methodology that was developed is demonstrated. The strategy for reaching the next milestones is an extrapolation of this method. Methodology here means first of all that a list of principles is adopted which ensure that the standard will fit the choosen scope and purpose. Secondly, it means that a list of concepts is defined which ensure that realization of the standard according to the principles is feasible.

Before further describing the technical work, a brief overview of the make-up of WG2 Graphics will be given.
The working group consists of about 35 technical experts appointed by 12 member countries. The member countries are:

Austria	Germany F.R.	Norway
Canada	Hungary	Switzerland
Finland	Italy	UK
France	The Netherlands	USA

The working group works by correspondence. It has an annual meeting, on which the results of that year are summarized and reported to the parent subcommittee. The latter waits for the working group to ask to be moved on to the next milestone. Currently the technical work is carried out in three subgroups as follows:

- Draft Standards Subgroup:
  charged with the review of the technical contents of GKS
- Editorial Subgroup:
  charged with the review of the non-technical aspects of GKS (e.g., English wording, ISO guide for format of a standard).
  Charged with the preparation of all formal decisions and liason with parent committee.

Charged with planning of future standardisation projects

- Reference Model Subgroup:
  continue exploring the graphics area for new standard projects. Further develop principles and concepts.
  Liaise with other ISO bodies.

2. HISTORY OF THE REVIEWING PROCESS

The exploratory stage of WG2 with respect to GKS can be divided into three periods:

1. Consensus about scope and principles
2. compatibility among candidates within the consensus frame
3. formulation of programme of work

The exploratory phase did not start from scratch. The establishment of WG2 was justified by already existing opinions that the need for a basic graphics package was urgent and that a starting point for such a package in terms of goal and basic concepts was present.

In Fig. 1, the first reference model for a basic system is depicted. It identifies two interfaces, namely, between package and application program and between package and graphical devices. This model, which was developed during a study within IFIP WG5.2 ([9]), states that the interface with the application program coincides with the interface between the model of the problem and a system that can produce a visible representation of the model. Internal to the package, the viewing functions map the model on a (logical) geometrical space. The second interface maps this logical picture onto a physical device.

The impact of these two concepts is that they define a scope for the functions of the package. Also they divide viewing functions into groups according to basic properties of a virtual device.

For input no conceptual analogue to viewing was adopted. However the virtual device concept was accepted. A virtual graphics device is, by definition, a device which can accept two seperate sets of functions. One

set consists of abstract graphical entities, called primitives, which will be mapped onto *actions* of a physical device. The second set only specifies the appearance of the primitives. They are called *attributes*. The virtual device level ensures that this division into primitives and appearance is the same irrespective of which physical device is mapped to. Since, for input, no appearance is to be controlled as long as echoing functions are omitted, input devices are logical devices. However, the decision was taken to choose the simplest possible data type to represent a logical input device. It was also assumed that interaction was the responsibility of the application program. This made it possible to have an output stream and an input stream which are mutually independent.

Next, the working group encouraged members to formulate proposals within this framework. At the same time consensus was developed on how to organize the proposal into major functional groups, giving a global characterization to each of those so-called modules. The purpose of this work was mainly to further develop expertise within the working group. As a side effect national bodies actually working on a proposal were given an idea how their proposal might be received.

The major functional groups that have since been used are the following:

Picture modules:

- output
- input
- attributes
- segments
- segment attributes

Environmental modules:

- control
- input mode
- metafile
- inquire

These groupings have proven very useful in structuring the reviewing process. The main reason is that a full evaluation of all functions in a proposal with respect to all concepts and principles (e.g., a matrix with functions and principles) is impossible to use because such an overview is absolutely

incomprehensible. A strongly reduced matrix of groups and *major* principles for each group is more realistic. In practice, however, even such a reduced matrix has only been produced partially, e.g., for areas of disagreement.

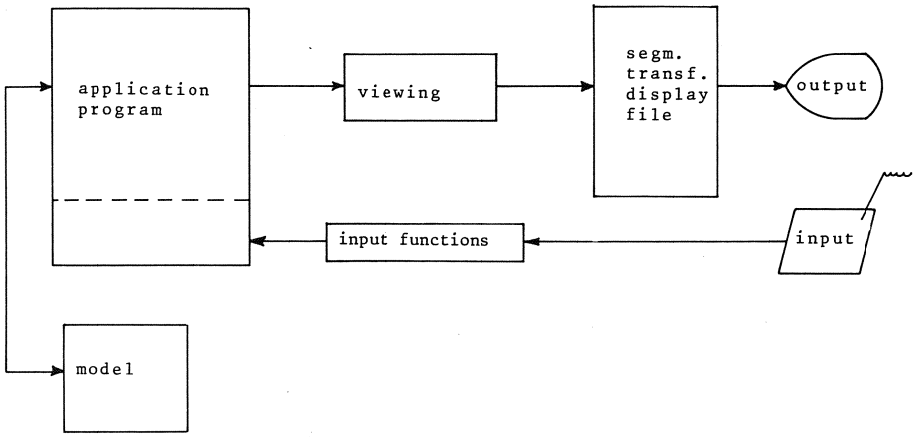
In appendix 1 an overview of the principles is given. It has turned out that a relatively small number of principles are difficult to follow. All others seem not to cause problems.

The two main principles concern application portability and device independence. In section 3, examples of how principles are developed and applied to functional groups will be given.

In the fall of '78 WG2 decided to offer the results of its study in a constructive way to the two groups working on a proposal (ACM SIGGRAPH GSPC on the CORE and DIN AK 5.9 on GKS). In the next six months, a compatibility study was performed which compared the two proposals by functional groups. Next, a small subgroup, called the editorial board, proposed changes to both documents which would make them at least compatible (i.e., the functions of one proposal could be expressed in terms of the other and vice versa) [2]. The impact of this study was considerable. It caused major changes in both proposals. Through this work WG2 was recognized as an important source for improvements.

In Fig. 2, the second reference model of both GKS'79 and GSPC-CORE '79 are given. It shows a fair degree of compatibility at least in structure. It also shows a more elaborate model compared to the first reference model (fig.1).

FIRST REFERENCE MODEL



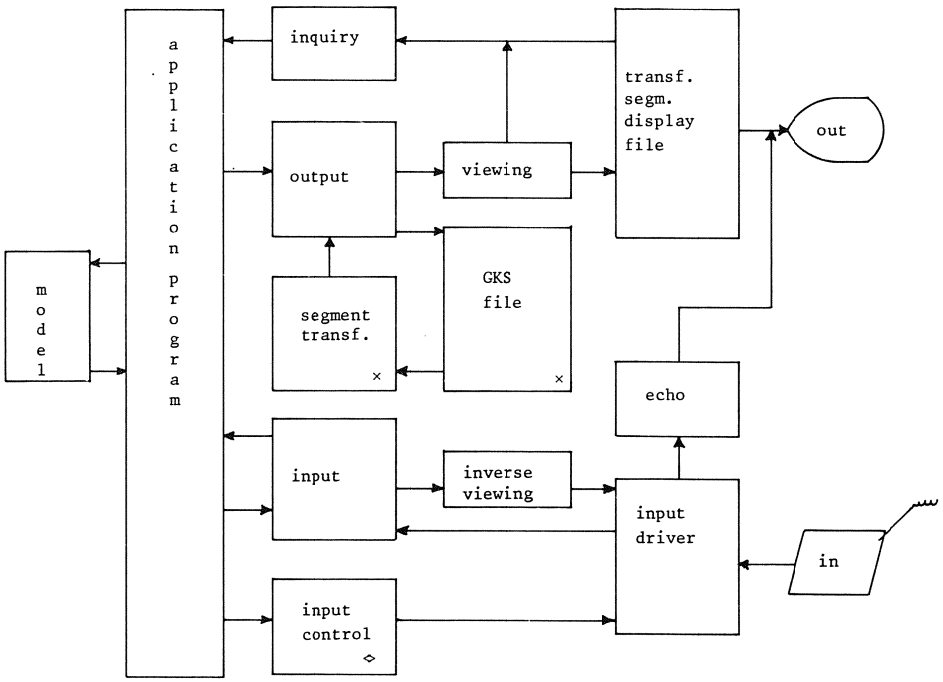
+ viewing
+ logical:
- interaction:
- feedback:

‖ - modeling
input/output
application dependent
device dependent

Figure 1.

SECOND REFERENCE MODEL

function modules and information flow



◇ CORE only

× GKS only

Figure 2

In addition it shows that both groups independently had started to fill the functional gaps with respect to low level feedback and interaction. The relative big differences are due to the fact that these parts of the proposals were not yet specified at the time of the compatibility study.

In the fall of '79, WG2 decided to propose GKS to ISO as the first work item in the area of Graphics. The compatibility study allowed basing a work item on either GKS or CORE. At that point in time DIN was the only member body prepared for sponsoring such a work item. It must be said, however, that the expertise developed within SIGGRAPH has not been lost. It has come back to WG2 via the newly formed ANSI X3H3 technical committee. In the fall of '80, WG2 managed to pass the formal barrier. GKS now is the first work item of WG2. This means that the reviewing process of GKS no longer is an internal affair of WG2. It has to present the result of reviewing the improved versions and an overview of items of disagreement to SC 5. For that purpose a *review format* has been designed (based on current practice within ANSI X3H3) which can be used to demonstrate progress and, moreover, ensures that progress cannot be eroded.

In addition the review format structures the discussions within, WG2 itself. The four basic documents in each review round are the following:

1. The *current version* of GKS being reviewed. (i.e., the working draft or draft proposal).
2. The *active issues list* containing all technical issues on which there (still) is disagreement.
3. The *resolved issues list* containing the issues which have satisfactorily been dealt with.
4. The *change list* containing changes of the current version as a result of resolving issues.

When the change list has grown substantially or when no additional changes are expected due to an empty active issues list, the changes are merged with the current version into a new current version.

The amount of progress made in the technical discussion can be demonstrated by comparing the active and resolved issues list. Especially the latter never shrinks. At any moment during the review process resolved issues as well as reinforcements of resolved issues may be added to it.

Disagreements are very carefully documented in the active issues list. Agreements are documented in the resolved issues list. The technical reviewers of WG2 work with the understanding that issues can be added to the active issues list only when they are really new issues or when completely new arguments are provided which justify resolving an issue in a different way. The resolved issues list also provides a basis for dealing with comments in later stages of the standardization process.

The format for both issues lists, not surprisingly, is the same. It lists the following information about each issue:

- title, i.e., a brief description
- subject area, e.g., input, text, segments, etc.
- category, e.g., disagreement, omission, ambiguity, inconsistency, question
- explanation
- alternatives for resolving the issue. One of the alternatives must be the "no change" alternative
- originator and date
- (resolution)

Care is taken that in description, subject area, explanations and arguments the established terminology is used. The list of concepts and principles is considered part of the terminology. It is mainly through the developments of arguments that the principles are made explicit. Allthough further experience must be gained, it appears that every argument, after being restated many times, directly refers to a principle.

Application of concepts and principles to major functional groups, as mentioned before, gives rise to so-called *meta-issues* (ef. [3] section 1 and [7]). Meta-issues are ad-hoc issues, which are defined and discussed in order to resolve a set of inter-related issues in a consistent way. The effect of resolving meta-issues is the provision of arguments for active issues. Meta-issues are raised in difficult subject areas. Difficult means that active issues in that area tend to become permanent. Formulating meta-issues is an excellent preparation for a discussion. However, meta-issues easily grow beyond their initial scope. Therefore they need to be raised with a certain reserve. A meta-issue entering the debate must be coupled with the intended resolutions in terms of alternatives and arguments.

During the compatibility period the major difficulties were overcome by resolving meta-issues on 2D-3D relationships, segment handling versus filing and level-structure.

Recently meta-issues on viewing transformations and attribute bundling have been very effective (cf. [7],[8]).

3. TECHNICAL RESULTS

The major goal of the graphics standardization effort is (cf. [1],[2]) to reduce the cost of accumulating hardware and software, converting software to get an application running. To achieve this goal the following principles must be considered. One cannot say: followed because many principles ask for opposed decisions i.e., a trade off must be found. These principles are (cf. [3]):

- A standard is *device independent* if any application program written to that standard can, without modification, offer equivalent facilities on a wide range of dissimilar graphics devices.
- *Compactness* is the degree to which the amount of data required by the standard primitives to represent a picture is reasonably small.
- *Size* is the space requirement needed to run the system.
- *Device richness* is the ability to exploit the capabilities of the full range of displays by using the standard facilities.
- *System portability* is the ability to transport implementations of the standard facilities.

The principles of device independence and application portability are by far the most important. They have influenced the resolution of most issues. This is not surprising if one looks at the goals stated for the standardization project.

System portability is left mainly to the implementors. Portable implementation of GKS functions should be possible but is not required for every implementation. Problems concerning compactness, size and richness strongly influence the (upwards compatible) level structure of GKS. Size and compactness accumulate from detailed design decisions and are therefore hard to trace. The demand for richness could be satisfied after the

introduction of levels. It is however interesting to see how the opinion on what is rich and what is basic has evolved.
As mentioned in the previous section GKS functions are organized in modules. This grouping was influenced by the portability and device independence principles. To characterize this influence the reference model of fig. 1 can be used further. The model depicts two separate information flows:

1. The output stream, driven by so-called viewing functions and
2. the input stream, driven by activation of abstract input devices.

The viewing functions visualize a model of the application without knowing anything about that particular model. The kernel therefore does not contain any functions for manipulating or structuring the model itself. Manipulations of the model can be visualized by regenerating the picture on the screen.

The input functions supply abstract data to the application program. Relationships between input and output are not explicit in the kernel. (Implicit relation exists for pick input only). Hence, the application program cannot delegate interaction functions to the kernel.

The virtual device concept used in this model is the vehicle for device independence. The actions a device can perform (e.g. draw a line, print a string, input a position) and the appearance of that action (e.g. a red line, a bold string, a tracking cross at the position) are controlled (if they are!) by separate functions. For both input and output a two-stage execution is assumed. For output, in the first stage a device independent representation of the primitives is generated, in which all device independent (geometrical) attributes are already applied. In the second stage these primitives are translated to hardware action controlled by the device dependent "current" attributes.

For input, in the first stage, i.e. at the physical device, the input primitive value is extracted from the incoming data from that device. The "attributes" controlling the appearance are purged. In the second stage the input primitives are synchronized with other input primitives and delivered to the application program. As a result the effect of input on the screen is not controlled by kernel functions.

The segmentation functions of GKS are also affected by the requirements of device independence. A segment can be redrawn on a different device.

Segments provide a basis for picture change. A segment is the unit of modification. Hence, application-programs have, via segmentation functions, control over the process of modification. The semantics of the segment functions guarantee application portability.

The two-stage output process models the segment storage functions. Storage of primitives and attributes can be device independent. Segments contain the result of the execution of the first-stage functions. Picture change may be caused by two sets of functions:

1. Segment manipulation e.g., delete a segment, create a new segment, transform a segment.
2. Change of attribute values which control the second-stage functions followed by a redraw of segments.

The second possibility of picture change requires the possibility of late binding between an attribute and its actual value. This concept has been under discussion until recently a satisfactory solution has been found.

Apart from storing pictures for the purpose of changing them, there is also long term filing. During the compatibility study it was resolved that these two forms of storage had to be kept independent.
Both the GSPC 77 proposal and the DIN 78 proposal were based on this model. The consensus period established that WG2 also wanted to work on this basis.

The major issues in the compatibility study were 2D-3D relationship, *current position* and escape functions. They will be discussed with respect to the above principles.

Application program portability requires that a 2D application could run on a 3D system. Hence, 2D is defined as a special case of 3D, i.e., take a default z-coordinate if needed. The only question that remains is what is the default mechanism? Two alternatives have been considered: z=0 and $z=cp_z$ (z-coordinate of current position). The first solution clearly is independent of both devices and application. For the second alternative we will first discuss CP.

Current position is a point in absolute world coordinates defining were the last output primitive ended. It can be used to define where the next output primitive may start or, the next primitive may use a coordinate value which is an off set from CP. CP and the possibility to use relative

coordinates are rich features. Therefore they might not be present in lower levels and they may not introduce device dependence. The latter can easily be the case because the standard cannot enforce the way actual devices produce primitives, and hence do not know were primitives end. The resolution adopted again uses the two-stage execution. The device independent description of output primitives after the first stage must be in absolute normalized device coordinates without any reference to CP. All primitives which cannot produce a "new CP" at this intermediate level (like text generated by hardware) are not allowed to change CP. Returning to the question of 2D-3D relation, taking the CP value as the default z-coordinate means adding a 0 value for z in the relative case. It means adding a relative value in the absolute case. In the current version of GKS no CP concept is used.

A generalized drawing primitive is a method for directly accessing special hardware facilities, like generating arcs. The purpose is to find a uniform scheme to address such facilities. Then either hardware can be used or the function can be simulated in software. In this way device dependencies can be isolated without rearranging application programs. Problems which were not solved during the compatibility study deal with two basic questions:

- is the GDP to be restricted to output primitives?
- at which stage is the result of a GDP delivered?
  (e.g. are transformation by passed?)

Two further important results from the compatibility study must be mentioned. The definition of an interface from GKS to a filing system which was completely transparent with respect to all GKS-structures like segments and attributes tables. This was achieved by separating the two storages functions (for filing and the segment storage for modification and reuse).

Last but not least a sequence of upwards compatible levels for GKS was defined without introducing new functions. The level structure turned out to be very helpful in judgeing realisation of principles concerning richness, compactness and size. The most important level in this respect is level 0 which indeed proves that the bulk of simple applications can be handled by small subset of GKS functions.

The compatibility study signalled differences with respect to input, but did not resolve them. For one reason, because both CORE and GKS were subject to considerable change at that point in time. The new definition of GKS and CORE which were influenced by [3] can be characterized by the reference model of fig. 2. It shows a more elaborate system. It also shows that on input the structure and the difference in approach were going to take considerable attention in the next review period.

For the review which started by the end of '78 only GKS (version 5.2) was taken into consideration. Issues inspired by differences with GSPC 79 were still submitted but they completely adhered to the (preliminary) issues format, so they were selfcontained. In the sequel we will trace issues concerning attributes and input in order to see how device independence and program portability were further supported.

For attributes GKS introduced the concept of a so-called bundle (in the document version 6.0 the terms bundle and table have the same meaning in the sequel they have not!). An attribute bundle contains a set of attribute values of different attributes. A bundle has a name. By selecting a certain bundle, the values of the bundle are invoked.

An attribute table contains a series of values for the same attribute. The values are invoked by defining a table index. In this way attribute value ranges can be defined through a table.

Bundles and tables are independent concepts, they can be freely compined. In GKS a fixed number of bundles and tables is envisaged. (e.g. pen bundle, colour table and a window viewport table).

The binding between bundle- and table index and actual bundles and tables takes place in the second stage of the executing of the viewing functions. Before the redraw of a segment table and bundle contents may be changed by the application program. This results in a picture change controlled by attributes.

The concept of bundles and tables also provide useful flexibility with respect to portability of programs. The contents of bundles and tables can be arranged (either statically or dynamically) in such a way that for a given hardware setting the optimal attribute selection takes place for a given class of pictures, without changing the application program (which only uses indexes). This possibility requires a division of attributes

in device independent attributes (like perhaps transformations) and attributes for which the mapping of a value to appearance control functions depends on the available hardware (like colour). The former group should (conceptually) be applied in the first stage of the execution of the viewing functions. The resolution of these problems require a careful definition of what tables and bundles are permitted. The associated flexibility and the fact that dynamic modification of bundles and tables can be optional (richness!) seems to promise a satisfactory solution.

The treatment of attributes in GKS is a typical example of the situation that, in order to solve problems connected with device independence and portability, the concept and functions had to be developed beyond current practice.

With respect to input the reviewing process is still going strong. Two main problems concerning input have been identified. They both can be viewed upon as attempts to have more control over input attributes.

The first set of issues concerns the possibility to return locator data in world coordinates. Based on the synthetic camera analogy, the user may look at different views of the application world which have accumulated on his screen. He therefore, quite naturally may wish to enter one of these "viewports". This means that for input a set of reverse window-viewports must be available, one for each view on the screen. The locator position on this screen selects the right mapping. Conflicts are resolved by priorities assigned to each viewport. The mapping can be considered a reverse viewing attribute for locators. The alternatives could be: either the current window/viewport for output is used or the application tries to find the intended view by searching its own history. In the latter case locators would return NDC. In the former case "old views" on the screen cannot be accessed by locators. In a long debate were the additional complexity was traded off against the gain in functionality a resolution has been found which accepts this concept in its most simple form. Other functions, like shielding, which could be introduced via this mechanism remain omitted. The multiple window/viewport concept can be structured as a table driven attribute. The reason for adding this function (also beyond current practice) is that more and more it is felt that locator input is the most fundamental graphical input device. The attribute as given turns the locator into a device

independent and application independent concept.

The second set of issues concerning input are associated with attributes that control the appearance of echos for logical input devices. For device independence the application program should not know how a logical device is realized on the existing hardware. However, certain echo modes can only be selected if a logical device is realized in a certain way. For instance the echoing for a choice device in the case of a realization via menues is different from the echoing in case of function buttons. A solution which has only become possible after introducing the attribute bundle concept and which is currently being studied is to define a table-like attribute, where the index is associated with a certain realization.

The situation is considerably complicated due to the fact that all logical input devices have to be realized with only a few physical devices. In that case conversions are required between devices. They can take place between logical devices, which bring them under application program control, or between a physical and a logical device, which may require attributes to control the context which defines which conversion will take place.

Allthough the right functionality has not yet been defined, the scope of the input functions is almost fixed. This means that in the near future a concentrated effort can take place on these problems. Also in this case a solution cannot easily be found within the area of common practice. There is some concern not to go too far beyond it.

REFERENCES

[1] DIN, *Graphical Kernel System*, Version 6.2, Functional Description, DIN 0066252 (July 1980).

[2] ACM-SIGGRAPH-GSPC, Status Report of the Graphics Standard Planning Committee in Computer Graphics, Vol. 13 (3), (August 1979).

[3] TEN HAGEN, P.J.W. & F.R.A. HOPGOOD, *Towards Compatible Graphics Standards*, Mathematical Centre Report IN 17, 1979.

[4] ISO/TC97/SC5/WG2 N54, *Concepts and Principles for Graphics Languages*, by P.R. BONO & K. WILLET.

[5] ISO/TC97/SC5/WG2 N83, *Procedure for Technical Comments on GKS*, by D.S.H. ROSENTHAL.

[6] ISO/TC97/SC5/WG2/N92, *GKS-Review*, Active Issues List.

[7] ISO/TC97/SC5/WG2 N93, *GKS-Review, Meta-Issues*, by BSI.

[8] ISO/TC97/SC5/WG2 N100, Report of the Draft Standard Subgroup, Meeting in Melbourne, Florida, 2-5/2/1981.

[9] GUEDJ, R.A. & H.J. TUCKER (eds), *Methodology in Computer Graphics*, Proc. IFIP WG5/2 workshop, North Holland, 1979.

APPENDIX 1

Principles Grouped by Category

System Costs

Access Depth

Extensibility

Minimality

Orthogonality

Symmetry

Upwards Compatibility

Application Cost

Clarity

Error Handling

Language Independence

Program Readability

Range of Applications

Richness

Routines versus Arguments

User Friendliness

Configuration Costs

Application Portability

Compactness

Device Independence

Device Richness

Size

System Portability

Operational Costs

Efficiency

Robustness

Global Considerations

Certifiability

Compatibility

Completeness

Consistency

Documentation

Implementability

State-of-the-Art