DYNAMIC ATTRIBUTES HANDLING ON A GKS WORKSTATION

M.Rudalics
Johannes Kepler University, Linz, Austria

ABSTRACT

Attributes handling has been cited as one of the more distinctive features of the new graphical standard GKS. Beforemost dynamic attributes handling facilities have been introduced in the standard with the capabilities of intelligent workstations in mind. However, a final attempt of attributes unification in the transition from version 7.0 to version 7.2 defeated some of the earlier design decisions for GKS. In the consequence special problems emerged for two-stage pipelines, where the generation of the display image is subdivided into a transformation/clipping part followed by a subsequent display list interpretation part. This paper attempts a classification of GKS attributes according to their type, scope, binding and the moment of their evaluation. Further a technique for dynamic attributes handling on a multi-microprocessor based vector refresh device is described.

"KEYWORDS:" Attributes Classification, Intelligent Graphics Workstations, GKS.

1 Introduction

The presence of attributes in graphics systems has been dictated by two reasons:

(1) Attributes allow to address device specific functions, like the capability of a line generator to draw various linestyles, or the capability of a character generator to display text in different sizes and fonts.

(2) Transmission and intermediate storage of primitives is more efficient when attributes may be employed: Transmitting and storing text is less time and space consuming when text has been encoded as a character string accompanied by some attributes specifying writing direction, font, et.al., than as series of strokes.

The Graphical Kernel System (GKS) [3], considers attributes handling as the performance of particular manipulations, which define position, size, shape, appearance, and identification of an output primitive on the viewing surface of a workstation. Dynamic attributes handling is the retroactive execution of these manipulations, i.e., for primitives which already appear on the display surface.

Workstations with low capabilities for dynamic attributes handling, like raster or storage tube devices, require for each retroactive modification of an attribute (with the exception of colour on a raster device) a regeneration of the display image, in GKS also referred to as new frame action. Workstations with high dynamic attributes handling capabilities, like vector refresh displays, allow retroactive modifications of attributes to occur immediately, e.g., on-the-fly from one refresh cycle to the next.

In the following chapters we will attempt a classification of GKS output primitive attributes under the special aspect of their adaptability to dynamic modification, and describe our realization of attributes handling on a multi-microprocessor based vector refresh device.

2 Classification of GKS Output Primitive Attributes

GKS output primitive attributes have been classified by various sources [1], [3], [5], [7] according to their type, scope and binding (see also Table 1).

(1) Types of Attributes

Geometric attributes, like character height or text path, affect shape and position of an output

Attribute	Type	Scope	Binding	Evaluation
Aspect Source	Implicit	Global	Static	→ Bundle Index/Individual
Bundle Index	Implicit	Global	Static	→ Representation
Individual Attributes	Non-geometric	Global	Static	Display
Pick Identifier	Identification	Retained	Stat	Picking
Other Primitives' Attributes	Geometric	Global	Static	DC Transformation
Representations	Non-geometric	Local	Dynamic	Display
Normalization Transformation	Geometric	Global	Static	NDC Transformation
Normalization Transformation	Implicit	Global	Static	→ Clipping Rectangle et.al.
Clipping Indicator	Implicit	Global	Static	→ Clipping Rectangle
Clipping Rectangle	Geometric	Global	Static	DC Transformation
Clipping Rectangle	Geometric	Retained	Dynamic	DC Transformation
Workstation Transformation	Geometric	Local	Dynamic	DC Transformation
Segment Name	Identification	Retained	Dynamic	Picking
Insert Transformation	Geometric	Retained	Static	NDC Transformation
Segment Transformation 1)	Geometric	Retained	Static	NDC Transformation
Segment Transformation	Geometric	Retained	Dynamic	DC Transformation
Visibility	Non-geometric	Retained	Dynamic	Display
Highlighting	Non-geometric	Retained	Dynamic	Display
Priority	Identification	Retained	Dynamic	Picking
Priority	Geometric	Retained	Dynamic	Display
Detectability	Identification	Retained	Dynamic	Picking
Echo Type 2)	Non-geometric	Retained/Local	Dynamic	Picking
Echo Transformation 3)	Geometric	Retained/Local	Dynamic	DC Transformation

1) The segment transformation of the inserted segment frozen during an insertion.
2) Echo type for the pick device, defines how the appearance of a primitive has to be altered (blinking), for providing the appropriate feedback to the operator during a pick process.
3) The echo transformation may be applied to a segment when choice echo type five has been selected.

Table 1: GKS Attributes Classification

primitive.

Non-geometric attributes, like linetype or linewidth, affect merely the appearance of the output primitive.

Identification attributes, like the segment name or the pick identifier, serve for identifying the primitive in a pick process.

Implicit attributes define an attribute of one of these three types indirectly, e.g., normalization transformation and clipping indicator define the clipping rectangle of a primitive.

(2) Scopes of Attributes

Global attributes, like the normalization transformation or character height, are applied to an output primitive on all open output workstations.

Local attributes, like the workstation transformation or text representation, are applied to primitives which appear on a specified workstation only.

Retained primitives' attributes, like the segment transformation or the pick identifier, are only applied to primitives which have been put into a segment before.

In connection with input some attributes (echo transformation, echo type) are of retained and local scope.

(3) Bindings of Attributes

Static attributes, like the pick identifier or text direction, are bound to a primitive for its lifetime. When a static attribute is evaluated (i.e., applied to the primitive), the value of the attribute is taken from the context valid at the time of creation of the primitive. This value has to be stored until evaluation along with the primitive on any intermediate storage.

Dynamic attributes, like colour representation or highlighting, may be modified retroactively throughout the lifetime of the primitive. When a dynamic attribute is evaluated, its value is taken from the context valid at the time of evaluation.

In what may be considered a final tour de force, the GKS designers attempted a unification of these concepts by establishing the two

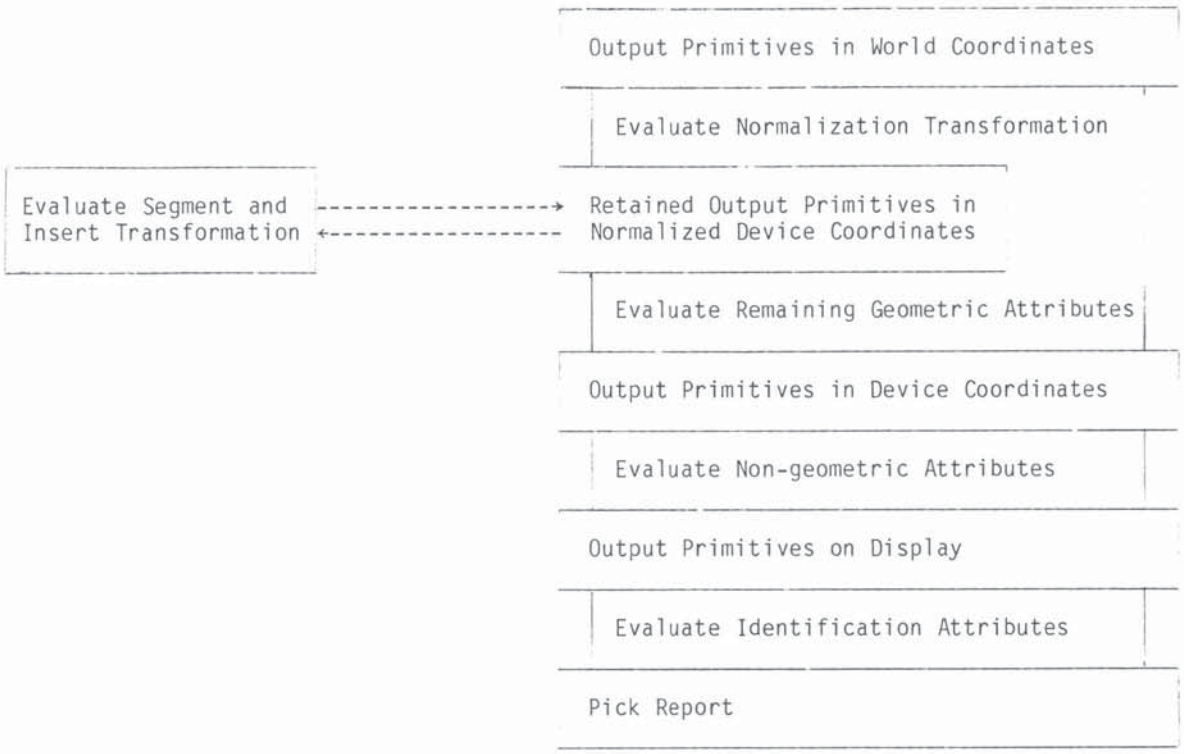

Figure 1: The GKS Viewing Pipeline

identities: geometric = static = global vs. non-geometric = dynamic = local attributes. Excluded from this unification were only segment attributes and the workstation transformation, due to their inherent dynamic character.

Unfortunately character spacing and the character expansion factor have been reclassified from geometric - GKS 7.0 - to non-geometric - GKS 7.2 - and thus dynamic attributes, consequently also subject to retroactive modification. This decision creates considerable difficulties for workstations, where the output pipeline is divided into a transformation/clipping part which creates a display list or refresh buffer in device coordinates notation, and a subsequent display part which interprets the display list for refreshing the display image: As in GKS 7.2 a text primitive may be modified dynamically by updating the character spacing and character expansion factor entries of the corresponding text bundle, the transformation/clipping process has to be initiated also for modifying the representation of text displayed in CHAR quality. This problem may not be circumvented, as the standard requires explicitely, that character spacing has to be evaluated exactly. Note, that a similar problem has been avoided with the marker size - another "non-geometric" attribute - by allowing an implementation to provide one marker size only.

3 Evaluation of Attributes - the GKS Viewing Pipeline

The purpose of a viewing pipeline is to describe where and when the evaluation, i.e., the application of an attribute to a primitive, has to be performed (Fig. 1).
- GKS requires the evaluation of certain attributes to occur before other actions may be taken: Before primitives may be stored on the Workstation Independent Segment Storage (WISS) or on a metafile they have to be transformed to normalized device coordinates (NDC) space.
- GKS requires the evaluation of other attributes to be postponed: The clipping rectangle must not be evaluated for retained primitives at the moment of NDC transformation, as a subsequent

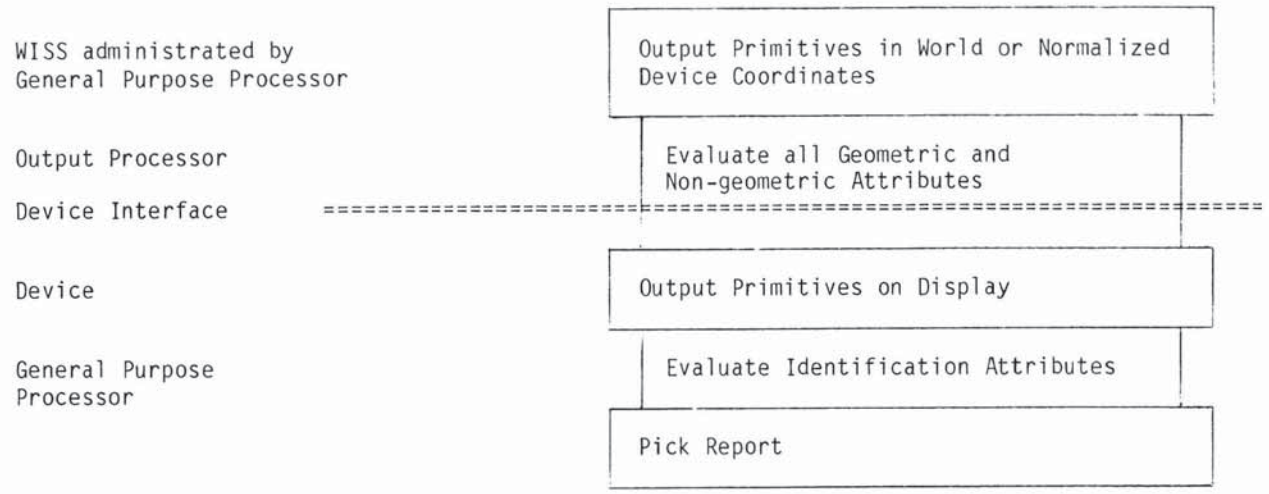

Figure 2: Encarnacao's VLSI Pipeline

insertion of the segment containing the primitives may require (due to a different normalization or clipping context) another clipping rectangle.

Generally, in a GKS implementation attributes evaluation may occur at one of the following moments:

- transformation to normalized device coordinates space
- transformation to device coordinates space
- displaying (i.e., mapping of abstract to device functions)
- picking.

Note, that in GKS the evaluation of attributes is independent from whether the attribute has been specified modally or as part of the primitive specification. The evaluation of dynamic attributes may be controlled seperately on each workstation which creates visible output with the help of the deferral state.

The evaluation of attributes may be combined as on Encarnacao's [2] one-stage VLSI pipeline (Fig. 2). The VLSI chip has been designed as an interface between a GKS implementation administrated by a general purpose processor, and an output device which is continuously refreshed by the VLSI processor. No intermediate storing of output primitives in device coordinates description is assumed. The VLSI pipeline requires some overhead, as:

- The segment storage has to accomodate non-retained primitives too.
- Due to restricted chip size, which requires polymarkers and text to be reduced to polylines, various capabilities of an output device, like character or marker generators, may not be addressed. Consequently the amount of text or polymarkers to be displayed may severely restrict the throughput of the device.
- Concatenating all transformations requires either to store normalization transformation and clipping indicator together with the primitive on the WISS, or to refrain from using the segment facility, as has been proposed for Minimal GKS [6].

The Dubna IGT [4] is a two-stage interpretation of the GKS pipeline (Fig. 3). The IGT accepts output primitive defined in NDC space only. All attributes which have been expressed in world coordinates (like the character height or the character up vector) have to be transformed to NDC space, and have to be retransformed whenever the valid normalization transformation is altered. When inserting a segment, the segment transformation of the inserted segment and the insertion transformation are bound to the primitives contained in the segment. Their evaluation is postponed until these transformations may be concatenated with the transformation of the segment open at the time of insertion and the workstation transformation. After the evaluation of all geometric attributes, the transformed and clipped primitives are stored on the display list from where the display image is refreshed.

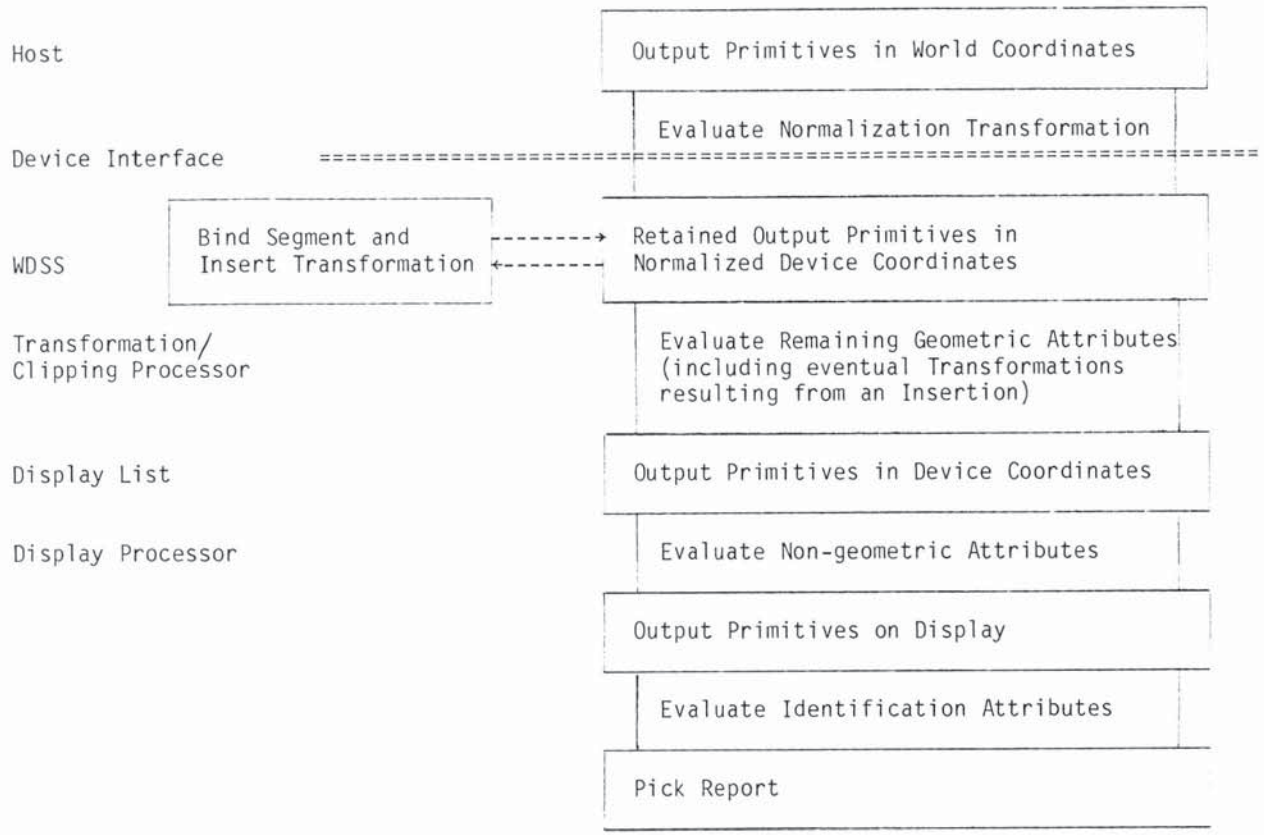

Figure 3: The Dubna IGT Pipeline

4 Dynamic Attributes Handling on the Workstation Level

The dynamic memory of the Dubna IGT has been subdivided into three logically distinct partitions:
- The Workstation Dependent Segment Storage (WDSS) contains the description of retained primitives in normalized device coordinates.
- The display list contains the description of retained and non-retained primitives in device coordinates.
- The bundles partition may be accounted in equal parts to: (1) the WDSS - as bundle indices are an integral part of the primitives defining a segment, (2) the display list - as the display process uses the information contained in bundles for displaying primitives (references to bundles are inserted accordingly in the display list), (3) the workstation state list. Latter is required by GKS.

Initially the bundles partition contains only predefined bundles. Like segment or display list elements, settable bundles may be created (allocated in the memory) dynamically without violating GKS rules. A new bundle has to be constructed when it is referenced for the first time in a SET...INDEX or a SET...REPRESENTATION function and may not be deleted until the workstation is closed.

On the Dubna IGT each bundle has two associated flags, which may be set every time the transformation process encounters an output primitive.

Flag_1 is set when a non-retained primitive which has to be displayed according to the information contained in the bundle is encountered (and may be clipped to non-existence) by the transformation process.

Flag_2 is set when a retained primitive which has to be displayed according to the information contained in the bundle is encountered (and may

be clipped to non-existence) by the transformation process.

If all entries of a bundle are shaded by the corresponding individual attributes (according to the current setting of the aspect source flags) neither flag_1 nor flag_2 are set.

Both flags are reset every time the display surface is cleared.

For displays operating in combined (store/refresh or write-thru) mode, flag_1 additionally reflects the usage of the bundle for displaying retained primitives in store mode. Flag_2 is set only when a retained primitive which is to be displayed in refresh mode uses this bundle.

Taking into account the current setting of these flags, three situations may arise when a dynamic attribute modification is about to be performed.
- When a SET...REPRESENTATION function modifies one of the entries of a bundle which has flag_1 set, a new frame action becomes necessary. This has the obvious consequence that all non-retained primitives will be thrown away with the next regeneration.
- When a SET...REPRESENTATION function modifies one of the entries of a bundle which has flag_2 set (but not flag_1), a reevaluation of all retained primitives becomes necessary (non-retained primitives are not affected by this process).
- When a SET...REPRESENTATION function modifies one of the entries of a bundle which has neither flag_1 nor flag_2 set, no action needs to be taken. This situation will usually occur during the initialization of a new bundle.

Both flags stand for the fact, that on the Dubna IGT it is impossible to modify dynamically the appearance of an output primitive, whenever a geometric entry of a bundle is involved. Modifying geometric aspects requires a reevaluation of the primitive for a correct application of the clipping algorithm. On the Dubna IGT the following entries of bundles may be modified dynamically: Linetype, polyline colour, marker type, polymarker colour, text font in STRING or CHAR quality (due to the fact that we use monosized fonts and clipping is performed on the character box), text colour, and fill area colour. A dynamic modification of other bundle entries, including character spacing and character expansion factor, may only be achieved through a reevaluation of all segments or a regeneration of the display image.

5 Conclusion

The implementation of dynamic attributes handling on a GKS workstation has been described. The realization reflects some of the difficulties encountered in the transition from GKS 7.0 to GKS 7.2. Nevertheless, the technique allows the full exploitation of the interactive facilities of the device, as a regeneration of the display image has only to be performed, when a visible change of the size or shape of an output primitive on the display surface is highly probable.

References:

[1] Arnold, D.B. The Importance of a Correct Approach to the Design of Metafiles Standards. In: Proc. of Eurographics '82, North-Holland Pub., 1982, pp.93-102.

[2] Encarnacao, J.L., Lindner, R., Mehl, M., Pfaff, G., and Strasser, W. A VLSI Implementation of the Graphics Standard GKS. Computer Graphics Forum 2, 2/3 (Aug. 1983), 115-121.

[3] Draft International Standard ISO/DIS 7942, Information Processing Graphical Kernel System (GKS), Functional Description, Version 7.2, NI-5.9/1-83, Nov. 1982.

[4] Leich, H., Levchanovsky, F.V., and Prikhodko, V.I. A Multi-Microprocessor Based Intelligent Graphics Terminal. Microprocessors and Microprogramming 12 (1983), 175-180.

[5] Rosenthal, D.S.H. Managing Graphical Resources. Computer Graphics 17, 1 (Jan. 1983), 38-45.

[6] Simons, R.W. Minimal GKS. In: Proc. of SIGGRAPH '83, Computer Graphics 17, 3 (July 1983), 183-189.

[7] Sutcliffe, D.C. Attribute Handling in GKS. In: Proc. of Eurographics '82, North-Holland Pub., 1982, pp.103-110.