Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

P.J.W. ten Hagen, M.M. de Ruiter

Segment grouping, an extension to the graphical kernel system

Computer Science/Department of Interactive Systems Report CS-R8623 August

---

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

Segment Grouping,

an Extension to the Graphical Kernel System

P.J.W. ten Hagen, M.M. de Ruiter
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

This paper introduces an extension to GKS, the ISO standard for 2D graphics software. Segment Grouping provides a device independent window manager functionality for application programmers. Segment Grouping allows for more efficient (given certain hardware restrictions) clipping and transformation of segments on both low and high function workstations.

1980 Math. Subject Classification : 69K32, 69K34, 69K30
Key Words & Phrases : Computer graphics, graphics systems, standardization, window management

1. Introduction

After several years of experience with GKS[1] implementations and use it is worth noticing that very few changes to GKS or extensions have been proposed. This is evidence for the fact that the generality of GKS surpasses most, if not all, other 2D packages. However two areas can be distinguished where applications fairly consistently try to add functionality which ought to be part of todays general purpose 2D graphics packages.

The first is *screen window management.*[2] This is the ability to divide the screen in rectangular areas and associate different data sets (e.g. text, pictures) with each of these windows. For alpha numeric screens, especially when they are equipped with so-called rasterop hardware facilities these window managers have been a tremendous success. For graphics devices this success has been relatively modest, mainly because of the low performance of raster devices to (re)draw pictures.

The second area is the possibilities for so-called *selective updates.* This is the possibility to minimize the number of graphical elements that need to be redrawn in order to visualize a picture change. (The not redrawn elements remain on the screen unchanged). This need is felt equally strong for high- and low performance devices, because in both cases unnecessary updates degrade performance well below expectations.

Screen window management can, among other things, provide a solution to the selective update problem. The functionality proposed in this report is termed *segment grouping.* It provides a window manager which quite naturally combines with GKS and yet supports all the facilities expected from window managers in as far as they are application program controlled (as opposed to operator controlled). Segment grouping also provides the basis for selective updates. This possibility can cause a dramatic performance improvement, especially for low cost hardware such as PC-graphics. In the next two sections the window management support and selective updates support will be explained.

Once segment grouping is introduced it is not difficult to discover many other useful functions that can be more easily supported. These however, will be discussed in a subsequent paper. This report only introduces the new concepts. The format is such that it can be seen as (another) appendix to GKS. This makes it easier to understand both in terms of added functionality and in terms of how to implement it. It may however be difficult to be understood for people who are not familiar with GKS.

Segment grouping can be implemented on any kind of workstation. It fully complies with the concept of device independence of GKS. It extends device independence by providing a device independent window manager which can be implemented using special window manager hardware support if available. The selective update facility is provided by allowing the GKS run-time system to restrict the updates to the windows that contain altered pictures.

Report CS-R8623
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

1.1. Window management

A window manager deals with screen windows on three different levels. Two of these levels are by definition outside the scope of GKS. They are the virtual terminal level and the virtual workstation level. Terminal properties and workstation properties other than workstation dependent attributes cannot be controlled by GKS. As a result all window management functions which allow an operator to open a "new screen window" and start an application program (e.g. a GKS application) in this window are transparent to GKS. If GKS will be opened it can open one or several virtual workstations each of which will be confined to this operator selected virtual terminal screen. The window manager will have to map the virtual workstation on the appropriate portion of the actual screen, even if this portion is changed in size, position or priorities through operator manipulations. The operator has the possibility to temporarily make a virtual workstation have (almost) full screen size.
The third level of window management, being the control of the virtual workstation, is given completely to the segment grouping extension. The basic mechanism for window control is obtained by extending the set of normalization transformations by a set of so-called group normalization transformations which have a fixed viewport (in NDC). These viewports are mapped onto the screen (by the workstation transformation, as usual) to determine the lowest level of screen windows, which are not under operator control (other than by manipulating the enclosing virtual workstation viewport).
By distributing the segments over those windows (i.e. by putting them in the corresponding "group") it is possible to assure the disjointness of such segment groups. Disjointness of groups results from either group windows being non-overlapping or by a shielding capacity given to each group. In order to ensure that group segments will be confined to this window, the segments share the group normalization transformation and clip (i.e. it is global to all segments in the group and cannot be changed for those segments). However individual segment transformations remain without restrictions. In addition there is a group transformation for all the group segments which is predefined to all group segment transformations. This group transformation can be controlled by both application program and operator. In this way the group window can actually be viewed as a "window" on the group segments.

1.2. The update function

The semantics of the update function are refined for grouping by maintaining an update status per group. This means that as soon as an update is necessary only those groups that need to be updated actually will be.
There is no separate workstation independent group store. The WISS is sufficient for supporting groups as well. The collection of segments in a WISS is available for including them individually in a group. Segments contained in a group at the WISS cannot be taken out of the group through the function COPY SEGMENT TO WORKSTATION or ASSOCIATE SEGMENT WITH WORKSTATION. Instead of those there now exists COPY GROUP TO WORKSTATION and ASSOCIATE GROUP WITH WORKSTATION. However, INSERT SEGMENT can use segments from groups as well as segments outside groups.
Segment groups do not introduce any overhead for the workstation independent part of the system. They allow for a much more efficient implementation of segment handling on a per workstation basis. The updates necessary during input (e.g. providing real-time feedback) can be executed much faster when groups are used. The response time improvements are so rewarding that it seems to be appropriate to introduce a whole new class of prompt/echo typed based on group functions. This will also be further explored in the future report.

1.3. Groups for GKS-3D

The concept of segment grouping can also be added to GKS-3D. This will be discussed in a separate document.

2. Concepts

2.1. Group attributes

A group has the following attributes:

1) Static :
- Group name, which is synonymous with its normalization transformation number.
- clipping window, which is identical to the viewport of the associated normalization transformation.

2) Dynamic :

- group transformation:	matrix
- priority:	if parts of groups overlap, the group with the higher priority will be prefered when the groups are displayed
- visibility:	VISIBLE/INVISIBLE
- detectability:	UNDETECTABLE/ DETECTABLE
- shielding:	NOSHIELDING/SHIELDING
- shielding colour:	a colourindex indicating the shield colour, which is painted as background.
- highlighting:	NORMAL/HIGHLIGHTED
- panning enabled:	DISABLED/ENABLED
- zooming enabled:	DISABLED/ENABLED
- rotating enabled:	DISABLED/ENABLED

Associated with these attributes are a number of attribute setting functions. In order to describe their effects, first the extensions of the GKS- and workstation statelists will be explained.

GKS statelist:
- holds the current values of static group attributes.
- when a group is open, the clipping window, being an entry in the global statelist, cannot be changed. (this can be enforced by forbidding normalization transformation set functions for the current normalization transformation or a more liberal approach would allow for all functions concerning transformations to be used as long as they don't have the effect of changing the clipping window in the GKS statelist)

Workstation statelist:
- has a list of all groups associated with this workstation
- may use the knowledge about grouping, in particular whether they overlap, for minimizing redrawing segments after dynamic changes.

Group statelist:
- when a group is created it gets default values for the dynamic attributes, which can be adjusted immediately following an CREATE/OPEN GROUP call, for this particular group.
- a group name is created for the group, dependent on the name of the current normalization transformation
- an empty group is made visible by its shield painted over the group window area on the screen, coloured with the background colour. depending on the setting of the shielding attribute either the viewport is solid filled or only the boundary is drawn.
- the treatment of groups is very much like the treatment of segments, but one level higher in the hierarchy.
- their is no need for an insert group function. however, an associate group function is provided as an extension of WISS for groups.

Group shielding affects the visibility of primitives laying in the clipping window area associated with a group. If the shielding attribute of a group is set to SHIELDING then a solid rectangle, equal to the group viewport, is drawn before drawing the group primitives. Thus it is possible to shield off all

primitives laying in the viewport of a specific group, and not belonging to the group. The shielding attribute can be set by the function SET GROUP SHIELDING. If shielding is set to NOSHIELDING only the boundaries of the rectangle are drawn. The colour to be used for this shielding action is indicated by the 'current shielding colour' entry in the group statelist, and can be changed by SET GROUP SHIELDING COLOUR.
Panning, zooming and rotation of a group can be realized by changing the group transformation. The pan, zoom and rotating ALLOWANCE fields in the group statelist indicates whether panning, zooming or rotating is either enabled or disabled.

2.2. Groups and segments
A group can be open when there is at least one workstation active. There can only be one open group at a time. A group cannot be opened when there is a segment open.

Primitives are not directly stored in a group, but indirectly via the segments contained in a group. A group itself does not contain any primitive. This includes that primitives outside segments will not go into a group.

Upon creating a group the set of active workstations is stored in the group statelist, thus being the set of workstations associated with the group. A workstation can be deactivated when a group is open. However it is an error to deactivate the last active workstation. By allowing a workstation to be temporarily deactivated it can skip certain segments of the group. An activate/deactivate workstation action does not affect the list of associated workstations in the group statelist. However the 'set of associated workstation' of a group can be extended by an ASSOCIATE GROUP WITH WORKSTATION call.

A close workstation action causes the workstation identifier to be deleted from the set of associated workstations in the group statelist of each group associated with the workstation. If the set of associated workstations of a group becomes empty, the group is deleted.

Workstations activated after the group has been opened are not added to the set of associated workstations, and are thus not affected by any function concerning the currently open group.

Closing a group when a segment is open is an error condition.

A closed group may be reopened. Thus it is possible to selectively store segments in the group. Segments created outside groups are not contained in a group, and cannot be affected or manipulated by any function concerning segment grouping.

On workstations not supporting grouping, a segment inside a group is treated as if there was no group open.

2.3. Group and segment attributes
The segments of a group are constrained by the group attributes. The segment priority/detectability/visibility is only defined relative to the other segments of the group. With respect to segments outside the group or other groups, the segment attributes are overruled by the group's attributes. E.g. segment detectability is valid only when the group is detectable. Likewise, segment visibility is only valid if the group is visible, etc. Segment priority is only effective in relation to other segments in the same group. The priority of segments in different groups is defined by the priority of the groups to which they belong. Segments not belonging to any group have lower priority than all group segments.

2.4. Group input priorities
Each group is associated with a normalization transformation. The connection is made via the group name which is identical to the number of one of the available transformations. During the time a group is open the associated normalization transformation must be the current transformation. To prevent unexpected visual effects on a workstation it is not allowed to change window or viewport of a normalization transformation associated with any of the existing groups.
To enforce that the creation of a locator input event inside a group viewport will return the normalization transformation belonging to the group, viewports associated with groups automatically get higher viewport input priority then viewports not associated with any group.

2.5. Deferring picture changes

GKS allows a workstation to delay for some time the actions requested by the application program. During this period the state of the display is undefined. The concept of deferral pictures refers only to a limited set of functions, namely those generating output. This set of functions must be extended with the following functions.

ASSOCIATE GROUP WITH WORKSTATION
COPY GROUP TO WORKSTATION

Depending on the type of workstation, some GKS functions can be performed immediately, while other functions lead to an implicit regeneration. Some new entries in the workstation description table indicate which changes caused by group functions require an implicit regeneration to update the workstation.
In general an implicit regeneration is equivalent to an invocation of the functions REDRAW ALL SEGMENTS ON WORKSTATION, and an invocation REDRAW GROUP ON WORKSTATION for each group on this workstation.

An implicit regeneration may be either allowed or suppressed, depending on the GKS deferral state. Besides these cases an implicit regeneration is also made necessary if any of the following occur:

1) If the 'dynamic modification accepted' entry in the workstation description table is IRG for group priority and this workstation supports group priority:

   i) if a segment containing primitives is added to a group with a higher group priority.

   ii) if the complete execution of one of the following actions would be affected by group priority.

   DELETE GROUP
   DELETE GROUP FROM WORKSTATION
   ASSOCIATE GROUP WITH WORKSTATION
   SET GROUP TRANSFORMATION
   SET GROUP VISIBILITY
   SET GROUP PRIORITY

2) if the dynamic modification entry in the workstation description table is IRG for group transformation:

   SET GROUP TRANSFORMATION

3) if the dynamic modification entry in the workstation description table is IRG for group visibility (visible -> invisible):

   SET GROUP VISIBILITY(INVISIBLE)

4) if the dynamic modification entry in the workstation description table is IRG for group visibility (invisible -> visible):

   SET GROUP VISIBILITY(VISIBLE)

5) if the dynamic modification entry in the workstation description table is IRG for highlighting:

   SET HIGHLIGHTING

6) if the dynamic modification entry in the workstation description table is IRG for delete group:

   DELETE GROUP
   DELETE GROUP FROM WORKSTATION

Deferred actions can be made visible by the use of UPDATE WORKSTATION.

If possible the implicit regeneration can be restricted to a subset of the execution of the REDRAW ALL SEGMENTS ON WORKSTATION function and all REDRAW GROUP ON WORKSTATION functions. In the ultimate case it can be reduced to one invocation of the function REDRAW GROUP ON WORKSTATION.

2.6. Clipping
Clipping takes place after the normalization transformation, segment transformation and group transformation have been applied.

At opening a group a static clipping window is bound to the group, being the clipping rectangle from the GKS statelist. During the time a group is open this clipping rectangle may not be changed. This is enforced by forbidding to change the clipping window of the GKS statelist via a change of the current normalization transformation viewport.
At reopening an already existing group it will be an error if the current clipping window of GKS is not the same as the clipping window of the reopened group.

Clipping rectangles are not changed by a group transformation.

GKS states that a clipping rectangle is associated with each primitive in a segment, being the clipping rectangle stored in the GKS statelist. Due to the current approach each primitive in a segment of a group, inserted by a GKS output function during the time the group was open, will have the same clipping window bound to it as the clipping window of the group. There is no way to insert primitives in an open segment with a different clipping window than the current GKS clipping window. This is enforced among others by not allowing a group to be open when calling ASSOCIATE SEGMENT WITH WORKSTATION.

2.7. Workstation Independent Segment and Group Storage
In GKS one workstation Independent Segment Storage (WISS) is defined. Besides segments outside groups this workstation may also contain groups. Groups stored on this workstation can be used for the COPY GROUP TO WORKSTATION or the ASSOCIATE GROUP WITH WORKSTATION functions. None of these functions modify the contents of the groups to which they are applied.

2.8. GKS operating states
GKS always exists in one of its predefined operating states. To enable the control over the allowance of the new grouping functions two new states are introduces.
After creation or opening a group GKS will be in the state GROP. Closing the open group brings the system back into the state WSAC. Opening a segment during the state GROP brings GKS in the state GOSO. Closing the segment moves GKS back to the state GROP.

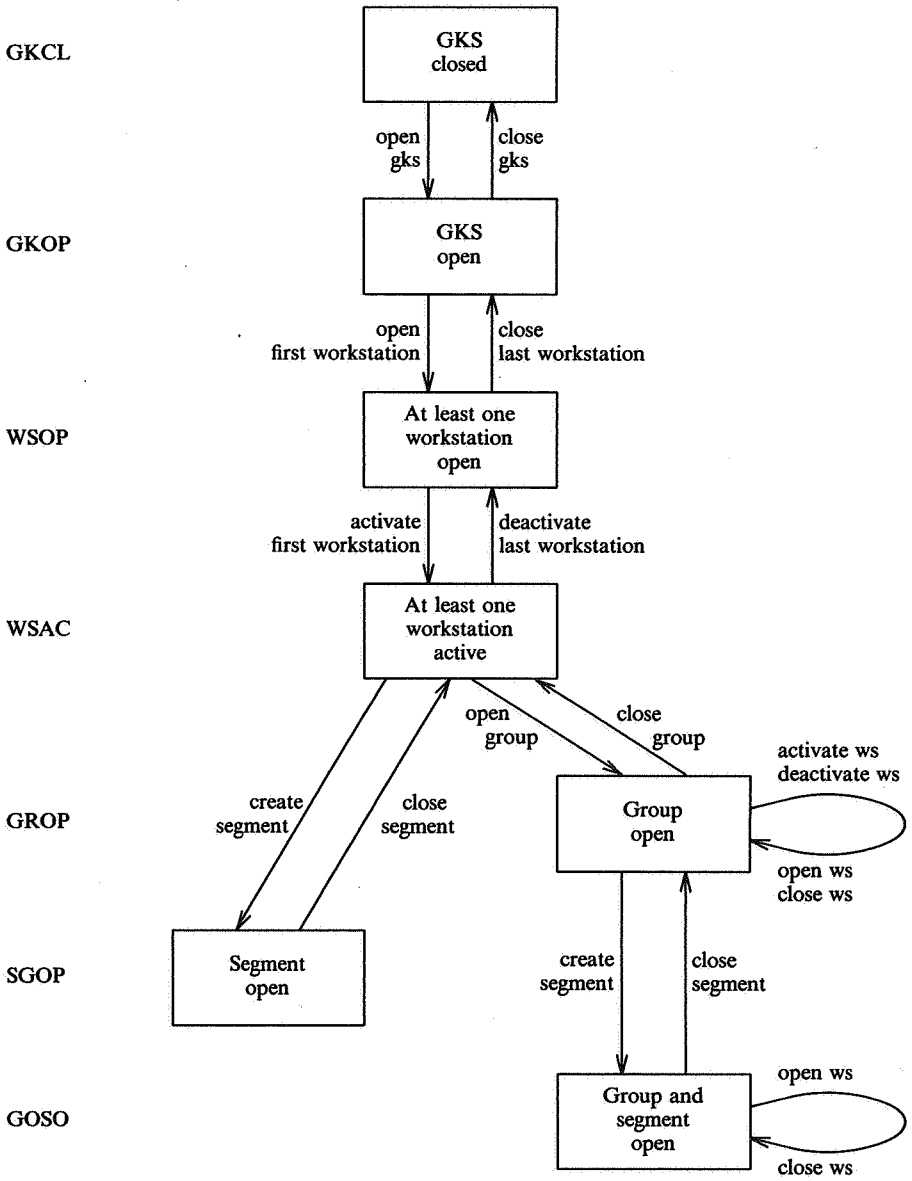

Figure 1. Some transitions between operating states

2.9. GKS 7.4 and Groups

As many standard GKS functions are also allowed in the state GROP and/or GOSO, the set of states in which each GKS function can be called must be redefined.

The functions ASSOCIATE SEGMENT WITH WORKSTATION and COPY SEGMENT TO WORKSTATION are restricted to segments outside groups.

As it is not allowed to change a normalization transformation connected to an existing group it is an error to call SET WINDOW or SET VIEWPORT for these transformations. The function SELECT CURRENT NORMALIZATION TRANSFORMATION may not be called when a group is open.

Additional information about restricted states and errorconditions is to be supplied for GKS 7.4 functions.

3. FUNCTIONS

3.1. Grouping functions

3.1.1. Grouping manipulation functions

CREATE GROUP WSAC L1a

Parameters:

Out group name N

Effect:
GKS is set into the state GROP if the current state is WSAC. The workstation statelist groupstatus entry of each active workstation, capable of handling groups, is set to GROUPOPEN. The group statelist is set up and initialized as follows:
The current normalization viewport is assigned to the viewport entry. The other entries are filled according to the default values:

visibility:	VISIBLE
shielding:	SHIELDING
shielding colour:	BACKGROUND COLOUR INDEX
highlighting:	NORMAL
detectability:	UNDETECTABLE
transformation:	IDENTITY

All subsequent output will be constrained (by clipping) to the group viewport.
A group name is created dependent on the name of the current normalization transformation.
The group name is recorded as 'the name of the open group' in the GKS state list. All subsequent segments until the next CLOSE GROUP will be collected into this group. The group name is entered into the 'set of stored groups for this workstation' in the workstation statelist of every active workstation able to handle groups. All active workstations able to handle groups are included in the 'set of associated workstations' of the group statelist of the opened group.
The group name is entered into the 'set of group names in use' in the GKS statelist. Primitive attributes are not effected.
Window and viewport of the associated normalization transformation are locked during the existence of the created group.
For further references to the created group the group name is returned to the caller of ths function.

Errors:

3	*GKS not in proper state: GKS shall be in state WSAC*
-831	*There exists already a group with group name equal to the current normalization transformation number.*
-836	*None of the active workstations able to support grouping*

OPEN GROUP WSAC L1a

Parameters:

In group name N

Effect:
GKS is set into the state GROP if the current state is WSAC. The workstation statelist entry of all active workstations, associated with the group, for the groupstatus is set to GROUPOPEN.
All subsequent output will be constrained (by clipping) to the group viewport.
The group name is recorded as 'the name of the open group' in the GKS state list. All subsequent output primitives until the next CLOSE GROUP will be collected into this group.
Primitive attributes are not effected.

Errors:

3	*GKS not in proper state: GKS shall be in state WSAC*
-830	*Specified group name is invalid*
-832	*Specified group name does not exist*
-837	*The normalization transformation associated with the specified group is not selected.*

CLOSE GROUP GROP L1a

Parameters:

none

Effect:
GKS is put into the operating state WSAC . Segments may no longer be added to the previously open group. The 'name of the open group' in the GKS statelist becomes unavailable for inquiry. The workstation statelist entry of all active workstations, associated with the group, for the groupstatus is set to NOGROUPOPEN.

Errors:
-800 *GKS not in proper state: GKS shall be in the state GROP*

DELETE GROUP WSOP, WSAC, SGOP, GROP, GOSO L1a

Parameters:

In group name		N

Effect:
The group is deleted. The group name is removed from each 'set of stored groups for this workstation' (in the workstation state lists) which contains it, and from the 'set of group names in use' in the GKS statelist. All segments in the group are deleted, i.e. an action equivalent to DELETE SEGMENT is performed for each segment in the group. The groups statelist is cancelled.

Errors:
-801 *GKS not in proper state: GKS shall be in state WSOP, WSAC, SGOP, GROP or GOSO*
-830 *Specified group name is invalid*
-832 *Specified group name does not exist*
-844 *Specified group is open*

DELETE GROUP FROM WORKSTATION WSOP, WSAC, SGOP, GROP, GOSO L1a

Parameters:

In workstation identifier	(1..n)	N
In group name		N

Effect:
The group is deleted from the specified workstation. The group name is removed from the 'set of stored groups for this workstation' in the workstation statelist. The workstation identifier is removed from the 'set of associated workstations' in the group statelist. If the 'set of associated workstations' becomes empty, the group is deleted, i.e. the DELETE GROUP function is performed.
All segments in the group are deleted at the workstation, that is, an action equivalent to DELETE SEGMENT FROM WORKSTATION is performed for each segment in the group.

Errors:
-801 *GKS not in proper state: GKS shall be in state WSOP, WSAC, SGOP, GROP or GOSO*
20 *Specified workstation is invalid*
25 *Specified workstation is not open*
33 *Specified workstation is of category MI*
35 *Specified workstation is of category INPUT*
-813 *Specified workstation does not support grouping*
-830 *Specified group name is invalid*
-833 *Specified group name does not exist on specified workstation*
-844 *Specified group is open*

CLEAR GROUP WSOP, WSAC, GROP L1a

Parameters:

In group name	N

Effect:
All segments in the group are deleted. The 'set of stored segments for this group' is set to empty.

Errors:

-806	*GKS not in proper state: GKS shall be in one of the states WSOP, WSAC or GROP*
-830	*Specified group name is invalid*
-832	*Specified group name does not exist*

REDRAW GROUP ON WORKSTATION WSOP, WSAC, GROP, SGOP, GOSO L1a

Parameters:

In workstation identifier	N
In group name	N

Effect:
The following actions are executed in the given sequence:

a) If the workstation transformation update state entry in the ws statelist is PENDING the current workstation window and current workstation viewport entries in the workstation statelist are assigned the values of the requested workstation window and requested workstation viewport entries. The workstation transformation update state entry is set to NOTPENDING.

b) If the specified group is visible the group is redisplayed. This action causes all visible segments stored in the group to be redisplayed.

Errors:

-801	*GKS not in proper state: GKS shall be in state WSOP, WSAC, SGOP, GROP or GOSO*
20	*Specified workstation identifier is invalid*
25	*Specified workstation is not open*
33	*Specified workstation is of category MI*
35	*Specified workstation is of category INPUT*
36	*Specified workstation is WISS*
-830	*Specified group name is invalid*
-832	*Specified group name does not exist*
-833	*Specified group does not exist on specified workstation.*

ASSOCIATE GROUP WITH WORKSTATION WSOP, WSAC L2a

Parameters:

In workstation identifier	N
In group name	N

Effect:
The group is sent to the specified workstation in the same way as if the workstation were active when the group was created. Clipping rectangles are copied unchanged. The group name is added to the 'set of stored groups for this workstation' in the workstation statelist. The workstation identifier is included in the 'set of associated workstations' in the group statelist, and in the 'set of associated workstations' in the segment statelists of all segments contained in the group.

Note: If the group is already associated with the specified workstation the function has no effect.

Errors:

6	*GKS not in proper state: GKS shall be in state WSOP, WSAC*
25	*Specified workstation is not open*
33	*Specified workstation is of category MI*
35	*Specified workstation is of category INPUT*

27 *Workstation Independent Segment Storage is not open*
-813 *Specified workstation does not support grouping*
-830 *Specified group name is invalid*
-834 *Specified group does not exist on Workstation Independent Segment Storage*

COPY GROUP TO WORKSTATION WSOP, WSAC L2a

Parameters:

In workstation identifier	N
In group name	N

Effect:
The primitives in the segments in the group are sent to the specified workstation after group transformation and clipping at the clipping rectangle stored with each primitive. The primitives are not stored in a segment or a group.

Errors:
6 *GKS not in proper state: GKS shall be in state WSOP, WSAC*
25 *Specified workstation is not open*
33 *Specified workstation is of category MI*
35 *Specified workstation is of category INPUT*
27 *Workstation Independent Segment Storage is not open*
36 *Specified workstation is Workstation Independent Segment Storage*
-830 *Specified group name is invalid*
-834 *Specified group does not exist on Workstation Independent Segment Storage*

3.1.2. Grouping Attributes

SET GROUP TRANSFORMATION WSOP, WSAC, SGOP, GROP, GOSO L1a

Parameters:

In group name	N
In transformation matrix	2×3×R

Effect:
The 'group transformation matrix' entry in the group statelist of the named group is set to the value specified by the parameter. When a group is displayed, the coordinates of the primitives in its segments are, before being transformed by the segment transformation, transformed by applying the following matrix multiplication to them:

$$\begin{Bmatrix} x' \\ y' \end{Bmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

This function can be used to transform a group stored on a workstation. The transformation applies to all workstations where the specified group is stored, even if they are not all active.
The group transformation (conceptually) takes place in NDC space. The group transformation will be stored in the group statelist and will not affect the contents of the group. The group transformation is not cumulative.

Errors:
-801 *GKS not in proper state: GKS shall be in state WSOP, WSAC, SGOP, GROP or GOSO*
-830 *Specified group name is invalid*
-832 *Specified group name does not exist*

SET GROUP VISIBILITY WSOP, WSAC, SGOP, GROP, GOSO L1a

Parameters:

In group name		N
In visibility	(VISIBLE, INVISIBLE)	E

Effect:
The 'visibility' entry in the group statelist is set to the value specified by the parameter.

Errors:
-801 *GKS not in proper state: GKS shall be in state WSOP, WSAC, SGOP, GROP or GOSO*
-830 *Specified group name is invalid*
-832 *Specified group name does not exist*

SET GROUP SHIELDING WSOP, WSAC, SGOP, GROP, GOSO L1a

Parameters:

In group name		N
In shielding	(NOSHIELDING, SHIELDING)	E

Effect:
The 'shielding' entry in the group statelist is set to the value specified by the parameter, thus enabling or disabling the shielding of primitives outside the specified group.

Errors:
-801 *GKS not in proper state: GKS shall be in state WSOP, WSAC, SGOP, GROP or GOSO*
-830 *Specified group name is invalid*
-832 *Specified group name does not exist*

SET GROUP SHIELDING COLOUR WSOP, WSAC, SGOP, GROP, GOSO L1a

Parameters:

In group name		N
In colour index	(0,n)	N

Effect:
The shielding colour index in the group statelist is set to the value specified by the parameter. If shielding is enabled this colour will be used as background colour.

The colour index is a pointer into the colour tables of the workstations. If the specified colour index is not present in a workstation colour table, a workstation dependent colour index is used on that workstation.

Errors:
-801 *GKS not in proper state: GKS shall be in state WSOP, WSAC, SGOP, GROP or GOSO*
92 *Colour index is less than zero*
-830 *Specified group name is invalid*
-832 *Specified group name does not exist*

SET GROUP HIGHLIGHTING WSOP, WSAC, SGOP, GROP, GOSO L1a

Parameters:

In group name		N
In highlighting	(NORMAL, HIGHLIGHTED)	E

Effect:
The 'highlighting' entry in the group statelist is set to the value specified. If the group is marked as HIGHLIGHTED and VISIBLE, the group is highlighted in an implementation dependent manner.

Errors:
-801 *GKS not in proper state: GKS shall be in state WSOP, WSAC, SGOP, GROP or GOSO*
-830 *Specified group name is invalid*
-832 *Specified group name does not exist*

SET GROUP PRIORITY WSOP, WSAC, GROP, SGOP, GOSO L1a

Parameters:

In group name		N
In group priority	[0,1]	N

Effect:
The group priority entry in the group statelist of the named group is set to the value specified. Group prioirty affects the display of groups and pick input (if groups can overlap).

Errors:
-801 *GKS not in proper state: GKS shall be in state WSOP, WSAC, SGOP, GROP or GOSO*
-830 *Specified group name is invalid*
-832 *Specified group name does not exist*
-835 *Specified group priority is outside range [0, 1]*

SET GROUP DETECTABILITY WSOP, WSAC, GROP, SGOP, GOSO L1a

Parameters:

In group name		N
In detectability	(UNDETECTABLE, DETECTABLE)	E

Effect:
The 'detectability' entry in the group statelist of the named group is set to the value specified. If the group is marked as DETECTABLE and VISIBLE the primitives in it are available for pick input. DETECTABLE but INVISIBLE groups cannot be picked.

Errors:
-801 *GKS not in proper state: GKS shall be in state WSOP, WSAC, SGOP, GROP or GOSO*
-830 *Specified group name is invalid*
-832 *Specified group name does not exist*

SET PAN, ZOOM AND ROTATE ALLOWANCE WSOP, WSAC, GROP, SGOP, GOSO L1a

Parameters:

In group name		N
In pan allowance	(DISABLED, ENABLED)	E
In zoom allowance	(DISABLED, ENABLED)	E
In rotate allowance	(DISABLED, ENABLED)	E

Effect:
The entries for pan, zoom, and rotate allowance in the group statelist of the named group are set accordingto the value specified. Panning, zooming or rotating of the primitives in the group is either enabled or disabled, depending on the setting of the flags. The primitives in the group may be panned , zoomed or rotated by the group transformation.

Errors:
-801 *GKS not in proper state: GKS shall be in state WSOP, WSAC, SGOP, GROP or GOSO*
-830 *Specified group name is invalid*
-832 *Specified group name does not exist*

3.1.3. Grouping inquiry functions

3.1.3.1. Inquiry function for GKS description table

INQUIRE WORKST. MAX. GROUP NUMBER GKOP, WSOP, WSAC, GROP, SGOP, GOSO L1a

Parameters:

Out error indicator		I
Out max nr of workstations associated with group	(1..n)	I

Effect:

Errors:

-808 *GKS not in proper state: GKS shall be in of of the states GKOP, WSOP, WSAC, GROP, SGOP, GOSO*

3.1.3.2. Inquiry functions for GKS state list

INQUIRE NAME OF OPEN GROUP GROP, GOSO L1a

Parameters:

Out error indicator		I
Out name of open group		N

Effect:

If the inquired information is available, the error indicator is returned as 0 and values are returned in the output parameters.

Errors:

-807 *GKS not in proper state: GKS shall be in state GROP or GOSO.*

INQUIRE SET OF GROUP NAMES IN USE WSOP, WSAC, GROP, SGOP, GOSO L1a

Parameters:

Out error indicator		I
Out number of segment names	(0..n)	I
Out set of group names in use		n×N

Effect:

If the information is available, the error indicator is returned as 0 and values are returned in the output parameters.

Errors:

-801 *GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, SGOP, GROP, GOSO.*

3.1.3.3. Inquiry functions for workstation statelist

INQUIRE SET OF GROUP NAMES ON WORKST. WSOP, WSAC, GROP, SGOP, GOSO L1a

Parameters:

In workstation identifier		N
Out error indicator		I
Out number of group names	(0..n)	I
Out set of stored groups for this ws		n×N

Effect:

If the information is available, the error indicator is returned as 0 and values are returned in the output parameters.

Errors:

-801 *GKS not in proper state: GKS shall be in one of states WSOP, WSAC, GROP, SGOP, GOSO*

20 *Specified workstation identifier is invalid*

25 *Specified workstation is not open*
33 *Specified workstation is of category MI*
35 *Specified workstation is of category INPUT*
-813 *Specified workstation does not support grouping*

INQUIRE WORKSTATION GROUP STATE WSOP, WSAC, GROP, SGOP, GOSO L1a

Parameters:

Out error indicator		I
Out workstation group state	(NOGROUPOPEN, GROUPOPEN)	E

Effect:
If the information is available, the error indicator is returned as 0 and values are returned in the output parameters.

Errors:
-801 *GKS not in proper state: GKS shall be in one of states WSOP, WSAC, GROP, SGOP, GOSO*
20 *Specified workstation identifier is invalid*
25 *Specified workstation is not open*
33 *Specified workstation is of category MI*
35 *Specified workstation is of category INPUT*
-813 *Specified workstation does not support grouping*

3.1.3.4. Inquiry functions for workstation description table

INQUIRE NUMBER OF GROUP PRIORITIES SUPPORTED GKOP, WSOP, WSAC, GROP, SGOP, GOSO L1a

Parameters:

In workstation type		N
Out error indicator		I
Out number of group priorities supported	(0..n)	I

Effect:
If the inquired information is available, the error indicator is returned as 0, and values are returned in the output parameters.

Errors:
-808 *GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC, GROP, SGOP, GOSO*
22 *Specified workstation type is invalid*
23 *Specified workstation type does not exist*
39 *Specified workstation is neither of category OUPUT nor of category OUTIN*
-813 *Specified workstation does not support grouping*

INQUIRE DYNAMIC MODIFICATION OF GROUP ATTRIBUTES WSOP, WSAC, GROP, SGOP, GOSO L1a

Parameters:

In workstation type		N
Out error indicator		I
Out group transformation changeable	(IRG, IMM)	E
Out visibility changeable from 'visible' to 'invisible'	(IRG, IMM)	E
Out visibility changeable from 'invisible' to 'visible'	(IRG, IMM)	E
Out shielding changeable	(IRG, IMM)	E
Out back ground colour changeable	(IRG, IMM)	E
Out highlighting changeable	(IRG, IMM)	E
Out group priority changeable	(IRG, IMM)	E

Out adding primitives to open segm in open group	(IRG, IMM)	E
Out group deletion immediately visible	(IRG, IMM)	E

Effect:
If the inquired information is available, the error indicator is returned as 0 and values are returned in the output parameters.
IRG means that implicit regeneration is necessary; IMM means that the action is performed immediately.

Errors:

-808	*GKS not in proper state; GKS shall be in one of the states GKOP, WSOP, WSAC, SGOP, GROP, GOSO*
22	*Specified workstation type is invalid*
23	*Specified workstation type does not exist*
39	*Specified workstation is neither of category OUTPUT nor of category OUTIN*
-813	*Specified workstation type does not support grouping*

INQUIRE GROUP SUPPORT ON WORKSTATION TYPE GKOP, WSOP, WSAC, GROP, SGOP, GOSO L1a

Parameters:

In workstation type	N
Out error indicator	I
Out group support available	E

Effect:
If the inquired information is available, the error indicator is returned as 0 and a value is returned in the output parameter.

Errors:

-808	*GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC, GROP, GOSO, SGOP*
22	*Specified workstation type is invalid*
23	*Specified workstation type does not exist*
39	*Specified workstation is neither of category OUTPUT nor of category OUTIN.*

3.1.3.5. Inquiry functions for segment statelist

INQUIRE NAME OF GROUP CONTAINING SEGMENT WSOP, WSAC, GROP, SGOP, GOSO L1a

Parameters:

In segment name	N
Out error indicator	I
Out name of group containing segment	N

Effect:
If the inquired information is available, the error indicator is returned as 0 and the values are returned in the output parameters.

Errors:

-801	*GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, GROP, SGOP, GOSO*
120	*Specified segment name is invalid*
122	*Specified segment name does not exist*
-845	*Specified segment is not contained in any group*

3.1.3.6. Inquiry functions for group statelist

INQUIRE SET OF SEGMENTS ASSOCIATED WITH A GROUP WSOP, WSAC, GROP, SGOP, GOSO L1a

Parameters:

In group name		N
Out error indicator		I
Out number of associated segments	(0..n)	I
Out set of associated segments		n×N

Effect:
If the inquired information is available, the error indicator is returned as 0 and the values are returned in the output parameters.

Errors:
-801 *GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, GROP, SGOP, GOSO*
-830 *Specified group name is invalid*
-832 *Specified group name does not exist*

INQUIRE GROUP ATTRIBUTES WSOP, WSAC, GROP, SGOP, GOSO L1a

Parameters:

In group name		N
Out error indicator		I
Out group clipping window		4×R
Out group transformation matrix		2×3×R
Out group priority	[0,1]	R
Out visibility	(VISIBLE, INVISIBLE)	E
Out detectability	(UNDETECTABLE, DETECTABLE)	E
Out shielding	(NOSHIELDING, SHIELDING)	E
Out shielding colour	(0..n)	I
Out highlighting	(NORMAL, HIGHLIGHTED)	E
Out panning enabled	(DISABLED, ENABLED)	E
Out zooming enabled	(DISABLED, ENABLED)	E
Out rotating enabled	(DISABLED, ENABLED)	E

Effect:
If the information is available, the error indicator is returned as 0 and values are returned in the ouptut parameters. Among these values the current group transformation matrix, as is set and stored in the group statelist, is returned.

Errors:
-801 *GKS not in proper state: GKS shall be in state WSOP, WSAC, GROP, GOSO or SGOP*
-830 *Specified group name is invalid*
-832 *Specified group name does not exist*

INQUIRE DYNAMIC GROUP TRANSFORMATION WSOP, WSAC, GROP, SGOP, GOSO L1a

Parameters:

In workstation name	N
In group name	N
Out error indicator	I
Out transformation matrix	2×3×R

Effect:
The current group transformation matrix, as is realized on the given workstation, is returned.

Errors:

-801 *GKS not in proper state.*
20 *Specifed workstation identifier is invalid*
25 *Specified workstation is not open*
33 *Specified workstation is of category MI*
35 *Specified workstation is of category INPUT*
36 *Specifed group does not exist*
-813 *Specified workstation is of category WISS*
-830 *Specified group name is invalid*
-832 *Specified workstation does not support grouping*
-839 *Specified workstation not associated with specified group*

INQUIRE DYNAMIC GROUP WINDOW WSOP, WSAC, GROP, SGOP, GOSO L1a

Parameters:

In workstation name		N
In group name		N
Out error indicator		I
Out window	WC	2×P

Effect:
if the inquired information is available, the current rectangle in worldcoordinates that maps the viewport of the given group if the current normalization transformation and the current dynamic group transformation are applied, is returned.

Errors:

-801 *GKS not in proper state:GKS shall be in of one of the states WSOP, WSAC, SGOP, GROP, GOSO*
20 *Specifed workstation identifier is invalid*
25 *Specified workstation is not open*
33 *Specified workstation is of category MI*
35 *Specified workstation is of category INPUT*
36 *Specified workstation is of category WISS*
-813 *Specified workstation does not support grouping*
-830 *Specified group name is invalid*
-832 *Specifed group does not exist*
-839 *Specified workstation not associated with specified group*

4. GKS DATA STRUCTURES

4.1. Statelist extensions

Besides a new data structure for the Group statelist some new fields and changes are to be added to the existing GKS data structures.

4.2. Group state list

group name		N	
set of associated workstations		n×N	
(list contains all workstations active at opening the group. associated workstions may be activated or deactivated while the group is open. this action however does not affect the list. activated workstations not contained in this list are not 'seen' by this group.)			
set of stored segments		n×N	empty
clipping window	NDC	4×R	0,1,0,1
group transformation matrix		2×3×R	1,0,0 0,1,0
(the elements M_{13} , M_{23} of the transformation matrix are in NDC coordinates, the other elements are unitless)			
visibility	(VISIBLE, INVISIBLE)	E	VISIBLE
shielding	(NOSHIELDING, SHIELDING)	E	SHIELDING
shielding colour		(0, n)	0
highlighting	(NORMAL, HIGHLIGHTED)	E	NORMAL
group priority	[0, 1]	R	0
detectability	(UNDETECTABLE, DETECTABLE)	E	UNDET.
panning	(DISABLED, ENABLED)	E	ENABLED
zooming	(DISABLED, ENABLED)	E	ENABLED
rotating	(DISABLED, ENABLED)	E	ENABLED

4.3. Operating state

change

Operating state	(GKCL, GKOP, WSOP, WSAC, SGOP)	E	GKCL

into

Operating state	(GKCL, GKOP, WSOP, WSAC, SGOP, GROP, GOSO)	E	GKCL

4.4. GKS description table

The following must be added to the GKS description table.

max nr of workstations associated with a group	(1..n)	I	i.d.

4.5. Global statelist

The following must be added to the global statelist.

name of open group	N	undef
set of group names in use	n×N	empty
set of group state lists (one for every group)		empty

4.6. Workstation description table

The following must be added to the workstation description table:

Entries in this group do not exist for ws of type INPUT, WISS, MO and MI

workstation is able to support grouping	(GROUPING, NOGROUPING)	E	NOGROUPING
number of group priorities supported	(0..n)	I	i.d.
if ws is able to support grouping then dynamic modification accepted for:			
group transformation	(IRG, IMM)	E	i.d.
visibility(visible->invisible)	(IRG, IMM)	E	i.d.
visibility(invisible->visible)	(IRG, IMM)	E	i.d.
shielding	(IRG, IMM)	E	i.d.
back ground colour	(IRG, IMM)	E	i.d.
highlighting	(IRG, IMM)	E	i.d.
group priority	(IRG, IMM)	E	i.d.
adding primitives to open segment in open group overlapping segment in group of higher priority	(IRG, IMM)	E	i.d.
delete group	(IRG, IMM)	E	i.d.

4.7. Workstation statelist

The following must be changed in / added to the workstation statelist.

Entries in this group only exist for workstations able to support grouping

workstation group state	(NOGROUPOPEN, GROUPOPEN)		NOGROUPOPEN
set of stored groups for this ws		n×N	empty

change

set of stored segments for this ws

into

set of stored segments outside groups for this ws

4.8. Segment statelist

The following must be added to the segment statelist.

name of group containing segment	N	undef

5. METAFILES

5.1. Generation of metafiles

The list of 'GKS functions and their effect on GKSM output workstations' (table 4 in annex E) is to be extended with the following entries:

GKS functions which apply to workstations of category MO	**GKS item created or effect**
Control functions	
REDRAW GROUP ON WORKSTATION	201
Group functions	
CREATE GROUP	211
OPEN GROUP	212
CLOSE GROUP	213
DELETE GROUP	214
DELETE GROUP FROM WORKSTATION	214
CLEAR GROUP	215
ASSOCIATE GROUP WITH WS	
COPY GROUP TO WORKSTATION	
Group attributes	
SET GROUP TRANSFORMATION	221
SET GROUP VISIBILITY	222
SET GROUP SHIELDING	223
SET GROUP SHIELDING COLOUR	224
SET GROUP HIGHLIGHTING	225
SET GROUP PRIORITY	226
SET GROUP DETECTABILITY	227
SET GROUP PAN/ZOOM/ROTATE ALLOWANCE	228

5.2. Interpretation of metafiles

E.4.9 Group manipulation

interpretation of the CREATE GROUP item causes the invocation of GKS functions SET CLIPPING INDICATOR with parameter CLIP, SET VIEWPORT with a clipping window and normalization transformation in accordance to the information stored in the item, SELECT NORMALIZATION TRANSFORMATION with the group name as parameter, and CREATE GROUP. The function SET VIEWPORT is not called if the group name is equal 0.
Item 214 causes an invocation of DELETE GROUP.
Interpretation of the other items in this class has the same effect as invocation of the corresponding GKS functions.

E.4.10 Group attributes

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

5.3. Control items

The following items are to be added to the list of metafile items.

E.5 Control items

REDRAW GROUP ON WORKSTATION

'GKSM 201'	L	G

G(i): group name

requests REDRAW GROUP ON WORKSTATION on all active workstations

E.13 Items for group manipulation

CREATE GROUP

'GKSM 211'	L	G

G(i): group name

OPEN GROUP

'GKSM 212'	L	G

G(i): group name

CLOSE GROUP

'GKSM 213'	L	G

G(i): group name

DELETE GROUP

'GKSM 214'	L	G

G(i): group name

CLEAR GROUP

'GKSM 215'	L	G

G(i): group name

E.14 Items for group attributes

SET GROUP TRANSFORMATION

'GKSM 221'	L	G	M

G(i): group name
M(6r): transformation matrix:

$$M_{11}, M_{12}, M_{13}, M_{21}, M_{22}, M_{23}$$

SET GROUP VISIBILITY

'GKSM 222'	L	G	S

G(i): group name
S(i): visibility (1 = VISIBLE, 0 = INVISIBLE)

SET GROUP SHIELDING

'GKSM 223'	L	G	S

G(i): group name
S(i): shielding (0 = NOSHIELDING, 1 = SHIELDING)

SET GROUP SHIELDING COLOUR

'GKSM 224'	L	G	C

G(i): group name
C(i): colour index

SET GROUP HIGHLIGHTING

'GKSM 225'	L	G	H

G(i): group name
H(i): highlighting
(0 = NORMAL, 1 = HIGHLIGHTED)

SET GROUP PRIORITY

'GKSM 226'	L	G	P

G(i): group name
P(r): group priority

SET GROUP DETECTABILITY

'GKSM 227'	L	G	D

G(i): group name
D(i): detectability
(0 = UNDETECTABLE, 1 = DETECTABLE)

SET GROUP PAN, ZOOM AND ROTATE ALLOWANCE

'GKSM 228'	L	G	D	D	D

G(i): group name
D(i): panning
D(i): zooming
D(i): rotating
(0 = DISABLED, 1 = ENABLED)

6. Error list

The following errors are added to the error list.

B.2 States

-800 GKS not in proper state: GKS shall be in the state GROP
-801 GKS not in proper state: GKS shall be in one of the states WSOP, WSAC, SGOP, GROP or GOSO
-802 GKS not in proper state: GKS shall be in one of the states WSAC or GROP
-803 GKS not in proper state: GKS shall be in one of the states SGOP or GOSO
-804 GKS not in proper state: GKS shall be in one of the states SGOP, GROP or GOSO
-805 GKS not in proper state: GKS shall be in one of the states WSAC, SGOP, GROP or GOSO
-806 GKS not in proper state: GKS shall be in one of the states WSOP, WSAC or GROP
-807 GKS not in proper state: GKS shall be in one of the states GROP, GOSO
-808 GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC, SGOP, GROP or GOSO

B.3 Workstations

-811 Specified workstation not known in group containing specified segment
-813 Specified workstation does not support grouping

B.14 Grouping

-830 Specified group name is invalid
-831 Specified group name is already in use
-832 Specified group name does not exist
-833 Specified group name does not exist on specified workstation
-834 Specified group does not exist on Workstation Independent Segment Storage
-835 Specified group priority is outside range [0, 1]
-836 None of the active workstations is able to support grouping
-837 The normalization transformation associated with the specified group is not selected.
-838 Tried to close the last ws associated with the currently open group
-839 Specified workstation not known on specified group
-840 Specified group name not equal to current normalization transformation number
-841 Specified segment not contained in currently open group
-842 Group containing specified segment must be open
-843 Tried to deactivate the last active ws associated with the currently open group
-844 Specified group is open
-845 Specified segment not contained in any group
-846 There exists already a group with group name equal to the current normalization transformation number

B.4 Transformations

-861 Contents of specified normalization transformation may not be changed, as transformation is connected to a group.
-862 Not allowed to select another normalization transformation during states GROP and GOSO.

TEST IMPLEMENTATION

The described grouping functionality is implemented and tested as an extension to C-GKS, being the GKS implementation in C, developed at the Centre for Mathematics and Computer Science.[3] An addition to our C binding of GKS[4] is defined for the grouping functions and data types. To enable C-GKS programmers to test and use the grouping extension two appendices are added. The first appendix contains the C binding for both the type definitions and the functions.
Appendix B contains the extension that is made to the di/dd interface, as stated by the structured data type 'Screen'. Via these grouping entries group control can be performed by the workstation driver.

ACKNOWLEDGEMENT

Discussions with Miente Bakker, Kees Blom, Fons Kuyk and Bill Sass markedly improved earlier versions of the grouping specification.

REFERENCES

1. ISO (1985). *Information Processing - Graphical Kernel System - Functional description,* International Standard DIS 7942.
2. F.R.A. HOPGOOD, D.A. DUCE, E.V.C. FIELDING, K. ROBINSON, A.S.WILLIAMS ED. (1986). *Methodology of Window Management*, Springer-Verlag, Berlin.
3. M.M.DE RUITER. *C-GKS user guide*, CWI, Amsterdam, to be published.
4. D.S.H. ROSENTHAL & P.J.W. TEN HAGEN (1982). GKS in C, *Proceedings Eurographics '82*, p. 359-369.

APPENDIX A: C BINDING

Group type definitions
The following new C-GKS typedefinitions are relevant for the GKS groups.

```
typedef Int Grpname;
typedef Real Grppri;
typedef struct {
    Grpname     gr_name;   /* group name */
    Wss         **gr_onws; /* pointer to list of associated ws, followed by NULL*/
    Nrect       gr_clip;   /* group clipping window */
    Tmat        gr_mat;    /* group transformation matrix */
    Bool        gr_vis;    /* group visibility */
    Bool        gr_shld;   /* shielding enabled or disabled */
    Cindex      gr_shcol;  /* colour of shield */
    Bool        gr_hil;    /* group highlighting */
    Grppri      gr_pri;    /* group priority */
    Bool        gr_det;    /* group detectability */
    Bool        gr_pan;    /* panning enabled */
    Bool        gr_zoom;   /* zooming enabled */
    Bool        gr_rotate; /* rotating enabled */
    Seginst     *gr_segs;  /* pointer to segments associated with group */
    }Group;
```

Several existing typedefs contain some new, grouping special, fields.
They are listed at the end of this section.

Grouping manipulation functions

CREATE GROUP — **Group *newgroup()**

Diagnostics:
For further reference to the newly created group a pointer to a structure of type Group is returned.

OPEN GROUP — **opengroup(gr)**
GROUP NAME — Group *gr;

CLOSE GROUP — **closgroup()**

DELETE GROUP — **zapgroup(grp)**
IN GROUP NAME — Group *grp;

DELETE GROUP FROM WORKST. — **delgroup(ws, grp)**
IN WORKSTATION IDENTIFIER — Wss *ws;
IN GROUP NAME — Group *grp;

CLEAR GROUP — **clr_group(grp)**
IN GROUP NAME — Group *grp;

REDRAW GROUP ON WORKSTATION — **redr_group(ws, grp)**
IN WS IDENTIFIER — Wss *ws;
IN GROUP NAME — Group *grp;

ASSOCIATE GRP. WITH W.S. — **sendgroup(ws, grp)**
IN WORKSTATION IDENTIFIER — Wss *ws;
IN GROUP NAME — Group *grp;

COPY GROUP TO WORKSTATION	**copygroup(ws, grp)**
IN WORKSTATION IDENTIFIER	Wss *ws;
IN GROUP NAME	Group *grp;

Grouping Attributes

TRANSFORM GROUP	**transgroup(grp, mat)**
IN GROUP NAME	Group *grp;
IN TRANSFORMATION MATRIX	Tmat *mat;

SET GROUP VISIBILITY	**grp_vis(grp, vis)**
IN GROUP NAME	Group *grp;
IN VISIBILITY	Bool vis;

SET HIGHLIGHTING	**grp_hil(grp, hlt)**
IN GROUP NAME	Group *grp;
IN HIGHLIGHTING	Bool hlt;

SET GROUP PRIORITY	**grp_prio(grp, pri)**
IN GROUP NAME	Group *grp;
IN GROUP PRIORITY	Grppri pri;

SET GROUP DETECTABILITY	**grp_det(grp, det)**
IN GROUP NAME	Group *grp;
IN DETECTABILITY	Bool det;

Diagnostics:
A 'det' value being TRUE means detectability enabled.

SET GROUP SHIELDING	**grp_shield(gr, shield)**
GROUPNAME	Group *gr;
SHIELDING	Bool shield;

Diagnostics:
A 'shield' value being TRUE means shielding enabled.

SET GROUP SHIELDING COLOUR	**grp_shcol(gr, col)**
GROUP NAME	Group *gr;
SHIELD COLOUR	Cindex col;

SET GROUP PAN, ZOOM, ROTATE	**grp_panzoom(gr, pan, zoom, rotate)**
GROUP NAME	Group *gr;
PAN ALLOWANCE	Bool pan;
ZOOM ALLOWANCE	Bool zoom;
ROTATE ALLOWANCE	Bool rotate;

C inquiry functions.
In addition to the GKS inquiry functions, new inquiry functions are defined to inquire values of the various statelists concerning grouping. If the inquiry functions are available via functions the following binding is supported.

INQUIRE WS MAX GROUP NUMBER	**Ercode i_max_nrgrp(max)**
OUT MAX GRP NUMBER	Int *max;

Diagnostics:
On return **max** is filled with the maximum number of groups on a workstation.

INQUIRE NAME OF OPEN GROUP	**Ercode i_opengrpname(name)**
OUT GROUP NAME	Group **name;

Diagnostics:
On return **name** is filled with a pointer to the statelist of currently open group.

INQUIRE SET OF GROUP NAMES IN USE

Ercode i_set_grps(nrnames, grpnames, sn_sz)
Int *nrnames;
Grpname *grpnames;
Int sn_sz;

Diagnostics:
On return the user supplied integer nrnames will contain the number of group names in use, while the user supplied array **grpnames**, being of size **sn_sz**, will contain the group names. If *nrnames is larger than sn_sz, only the first sn_sz entries are filled, and error -18 is returned.

INQUIRE SET OF GROUP NAMES ON WS

Ercode i_grps_ws(ws, nrnames, sname, sn_sz)
Wss *ws;
Int *nrnames;
Grpname *sname;
Int sn_sz;

Diagnostics:
On return the user supplied parameter **nrnames** is filled with the number of group names on workstation. The user supplied array **sname** is filled with the group names. If the array is not large enough to hold all the names, only the first sn_sz names are stored and an error is returned.

INQUIRE WORKSTATION GROUP STATE
IN WORKSTATION
OUT WS GROUP STATE

Ercode i_wsgrpstate(ws, gr_open)
Wss *ws;
Bool *gr_open;

Diagnostics:
On return **gr_open** is filled with the workstation group state. If there is a group open on the workstation gr_open is TRUE, else FALSE.

INQUIRE NR OF GROUP PRIO'S SUPP.
IN WS TYPE
OUT NR OF GROUP PRIO'S SUPP

Ercode i_nr_grpprio(wsd, nr_grpprio)
Wsd *wsd;
Int *nr_grpprio;

Diagnostics:
On return the user supplied parameter **nr_grpprio** is filled with the number of group priorities supported. on a workstation.

INQ DYN MOD OF GROUP ATTS
IN WS TYPE
OUT IMPLICIT REGENERATION FLAGS

Ercode i_dm_grp_atts(wsd, irgflags)
Wsd *wsd;
Int *irgflags;

Diagnostics:
On return the user supplied parameter **irgflags** contains a mask for the dynamic modification settings for group attributes, where:

group transformation	if (*irgflags & 01)	then IMM else IRG
visible -> invisible	if (*irgflags & 02)	then IMM else IRG
invisible -> visible	if (*irgflags & 04)	then IMM else IRG
shielding	if (*irgflags & 010)	then IMM else IRG
back ground col change	if (*irgflags & 020)	then IMM else IRG
hilighting	if (*irgflags & 040)	then IMM else IRG
group priority	if (*irgflags & 0100)	then IMM else IRG
adding prims to open seg in grp	if (*irgflags & 0200)	then IMM else IRG
grp deletation immediate visible	if (*irgflags & 0400)	then IMM else IRG
grp clearance immediate visible	if (*irgflags & 01000)	then IMM else IRG

INQUIRE GROUP SUPPORT ON WS
IN WS TYPE
OUT GROUP SUPPORT

Ercode i_grp_support(wsd, support)
Wsd *wsd;
Bool *support;

Diagnostics:
On return **support** is filled with TRUE if the workstation type supports grouping, else FALSE.

INQUIRE NAME OF GROUP CONT. SEGM.
IN SEGMENT NAME
OUT GROUP NAME

Ercode i_grpname_seg(segname, sg_grp)
Segname segname;
Group **sg_grp;

Diagnostics:
On return **sg_grp** is filled with the group pointer of the group containing the segment with segmentname **segname**.

INQUIRE SET OF SEGMENTS IN GROUP
IN GROUP NAME
OUT NUMBER OF SEGMENTS IN GROUP
OUT SEGMENTS IN GROUP
IN SIZE OF ARRAY segs

Ercode i_segs_in_grp(grp, nr_segs, segs, sn_sz)
Group *grp;
Int *nr_segs;
Seg **segs;
Int sn_sz;

Diagnostics:
On return **nr_segs** is filled with the number of segments in the group. The user supplied array **segs**, being of size **sn_sz**, is filled with the segment pointers of the segments available in the group. If *nr_segs is larger than the sn_sz, only the first sn_sz entries are filled, and error -18 is returned.

INQUIRE GROUP ATTRIBUTES
IN GROUP NAME
OUT GROUP ATTRIBUTES

Ercode i_grpatt(grpname, group)
Grpname grpname;
Group **group;

Diagnostics:
On return the group pointer **group** will point to the group structure belonging to the group with name **grpname**. The Group structure will contain among others the information asked for. The Group type looks as follows:

```
typedef   struct {
          Grpname   gr_name;     /*group name */
          Wss       **gr_onws;   /*pointer to list of associated ws, followed by NULL*/
          Nrect     gr_clip;     /* group clip rectangle */
          Tmat      gr_mat;      /* group transformation matrix */
          Bool      gr_vis;      /* group visibility */
          Bool      gr_shld;     /* shielding enabled or disabled */
          Cindex    gr_shcol;    /* colour of shield */
          Bool      gr_hil;      /* group highlighting */
          Grppri    gr_pri;      /* group priority */
          Bool      gr_det;      /* group detectability */
          Bool      gr_pan;      /* panning enabled */
          Bool      gr_zoom;     /* zooming enabled */
          Bool      gr_rotate;   /* rotating enabled */
          Seginst   *gr_segs;    /* pointer to segments associated with group */
          } Group;
```

INQUIRE DYNAMIC GROUP TRAFO
IN WORKSTATION NAME
IN GROUP NAME
OUT GROUP TRANSFORMATION

Ercode i_grdyntrafo(ws, gr, tmat)
Wss *ws;
Group *gr;
Tmat *tmat;

Diagnostics:
On return **tmat** is filled with the group transformation matrix as dynamically set on the workstation.

INQUIRE DYNAMIC GROUP WINDOW
IN WORKSTATION NAME
IN GROUP NAME
OUT WINDOW

Ercode i_grwindow(ws, gr, wc)
Wss *ws;
Group *gr;
Wc *wc;

Diagnostics:
On return the 4 user supplied world coordinates pointed to by **wc**, are filled with the group window as dynamically set on the workstation.

C inquiry macro's
Nearly all the inquire functions can be made available via macros, producing in-line code. They all inquire a field of one of the available statelists, and return the value of that particular field. The argument 'fn' is a field name of the structure.
To enable the use of the inquiry functions via macro's producing in-line macro's, the following macro binding is defined.

INQ WS MAX GROUP NUMBER	call:	return val of type:
Out max nr of ws ass with grp	**inq_gkd(gd_nwsag)**	Int
INQ NAME OF OPEN GROUP	call:	return val of type:
Out name of open group	**inq_gkss(gk_opgrp)**	Group *
INQ SET OF GROUP NAMES IN USE	call:	return val of type:
Out set of group names in use	**inq_gkss(gk_grp)**	Grpinst *
INQ SET OF GROUP NAMES ON WS	call:	return val of type:
Out set of stored groups for this ws	**inq_wss(ws, ws_grp)**	Grpinst *
INQ WORKSTATION GROUP STATE	call:	return val of type:
Out workstation group state	**inq_wss(ws, ws_gropen)**	Bool
INQ NR OF GROUP PRIO'S SUPP	call:	return val of type:
Out nr of group prio's supported	**inq_wsd(wsd, wd_ngprio)**	Int
INQ DYN MOD OF GROUP ATTR	call:	return val of type:
	inq_wsd(wsd, wd_dmag)	Int

The field wd_dmag is an integer. If the bits of this integer are numbered from RIGHT to LEFT then the bits stand for:
A bit being 0 means IRG, a bit being 1 means IMM.

group transf. changeable:	1th bit
visib. changeable from vis to invis:	2nd bit
visib. changeable from invis to vis:	3nd bit
shielding changeable:	4th bit
back ground colour changeable:	5th bit
highlighting changeable:	6th bit
group priority changeable:	7th bit
adding prim's to seg in open grp:	8th bit
group deletion imm. visible:	9th bit
group clearance imm. visible:	10th bit

INQ GROUP SUPP ON WS	call:	return val of type:
Out group support available	**inq_wsd(wsd, wd_grp)**	Bool
INQ SET OF SEGMENTS ASS. WITH GROUP	call:	return val of type:
Out set of segments	**inq_grp(grp, gr_segs)**	Seginst *
INQ GROUP ATTRIBUTES	call:	return val of type:
Out group clipping window	**inq_grp(grp, gr_clip)**	Nrect
Out group transformation matrix	**inq_grp(grp, gr_mat)**	Tmat
Out group priority	**inq_grp(grp, gr_pri)**	Grppri
Out visibility	**inq_grp(grp, gr_vis)**	Bool
Out detectability	**inq_grp(grp, gr_det)**	Bool
Out shielding	**inq_grp(grp, gr_shld)**	Bool
Out shielding colour	**inq_grp(grp, gr_shcol)**	Cindex
Out highlighting	**inq_grp(grp, gr_hil)**	Bool
Out panning enabled	**inq_grp(grp, gr_pan)**	Bool
Out zooming enabled	**inq_grp(grp, gr_zoom)**	Bool
Out rotating enabled	**inq_grp(grp, gr_rotate)**	Bool
INQ NAME OF GROUP CONTAINING SEGMENT	call:	return val of type:
Out name of group containing segm.	**inq_seg(seg, sg_grp)**	Grpinst *

The inquiry functions INQUIRE DYNAMIC GROUP TRANSFORMATION and INQUIRE DYNAMIC GROUP WINDOW couldn't be made availabel via inline macros. They are only available as functions.

New fields in existing C types.

The following C-GKS typedefs contain some new fields. The newly added entries are printed bold.

```
/*
 * segment statelist
 */
typedef     struct {
        Segname     sg_sn;
        Wss         *sg_onws[NWSAS+1];
        Tmat        sg_mat;
        Bool        sg_vis;
        Bool        sg_hil;
        Segpri      sg_pri;
        Bool        sg_det;
        Bhead       *sg_firstbh;
        Int         *sg_local;
        Group       *sg_grp; /* pointer to group containing segment */
        }   Seg;

/*
 * global description table
 */
typedef struct {
        GksLevl     gd_lev;
        Int         gd_nrwt;
        String      *gd_avwt;
        Int         gd_nopws;
        Int         gd_nactv;
        Int         gd_nwsas;
        Int         gd_nntran;
        Int         gd_nwsag; /* maximum number of groups on a workstation */
        }   Gksd;

/*
 * workstation description table
 */
typedef     struct {
        Wscat       wd_cat;
        Wsrov       wd_rov;
        Screen      *wd_screen;
        .
        .
        .
        Int         wd_nprio;
        Int         wd_ndevs[6];
        Iddescr     *wd_ddesc[6];
        Bool        wd_grp; /*  workstation type supports grouping */
        Int         wd_ngprio;/* number of group priorities supported */
        Int         wd_dmag; /* dynamic modification accepted flags*/
        }   Wsd;
```

```
/*
 * workstation statelist
 */
typedef struct {
        Wsd         *ws_wsd;
        Wsis        *ws_wsis;
        .
        .
        .
        Int          ws_ncolr;
        Colour      *ws_colr;
        Idevice     *ws_devs[6];
        Bool        ws_gropen;  /* currently open group on workstation */
        Grpinst     *ws_grp;        /* set of groups on workstation */
        } Wss;

/*
 * GKS global statelist
 */
typedef struct {
        Wss         **gk_opws;
        Wss         **gk_actv;
        Asflags     gk_asf;
        Bindex      gk_line;
        .
        .
        .
        Seg         *gk_opsg;
        Seginst     *gk_segs;
        Quevent     *gk_queue;
        Bool        gk_more;

        Group       *gk_opgrp; /* currently open group */
        Grpinst     *gk_grp;      /* set of existing groups */
        } Gks;
```

APPENDIX B

DI/DD interface
For intelligent devices, able to handle the grouping locally, the grouping extension makes it necessary to add new entries to the di/dd interface. New entries are added at the end of the current Screen structure.

```
typedef struct {
        Ic          s_chsz;
        Int         s_chht;
        Bool        s_clipflg;
        Bool        s_hattflg;
        Bool        s_segmflg;
        Bool        s_sgtrflg;
        Bool        s_hinpflg;

        Bool        (*s_open)();
        Bool        (*s_clos)();
        .
        .
        .
        Bool        (*s_irep)();
        Bool        (*s_iidv)();
        Bool        (*s_itex)();
        Bool        (*s_res4)();
        Bool        (*s_res5)();
        Bool        (*s_gcix)();

/***GKS GROUP extension entries*****************************************/

        Bool        s_grpflg;    /* TRUE if ws can handle grouping.
                                  * This implies that the following
                                  * entries are unequal NULL
                                  */

        Bool        (*s_rdgr)(); /* redraw all groups on workstation */
        Bool        (*s_grcr)(); /* new group */
        Bool        (*s_grop)(); /* open group */
        Bool        (*s_grcl)(); /* close group */
        Bool        (*s_grdl)(); /* delete group */
        Bool        (*s_grem)(); /* clear/empty group */
        Bool        (*s_grat)(); /* set group attributes */
        Bool        (*s_igrt)(); /* inquire group transformation */
} Screen;
```

The parameters of the functions that can be called via the new entries are:

```
Bool
dev_grcr(ws, grp) /* new group */
Wss     *ws;
Group *grp;

Bool
dev_grcl(ws) /* close group */
Wss         *ws;

Bool
dev_grop(ws, gr) /* open group */
Wss     *ws;
Group   *gr;

Bool
dev_grdl(ws, grpname) /* delete group */
Wss     *ws;
Group       *grpname;

Bool
dev_grem(ws, grpname) /* clear group */
Wss     *ws;
Group   *grpname;
```

```
Bool
dev_grat(ws, grpname, attsel, mat, bool, prio, shield, pan)/* set grp attributes*/
Wss       *ws;
Group     *grpname;
Int       attsel; /* which of the attributes to be set
                  * attsel = S_GR_TR: set group transform
                  * attsel = S_GR_SH: set group shielding
                  * attsel = S_GR_SC: set group shielding colour
                  * attsel = S_GR_VI: set group visibility
                  * attsel = S_GR_HI: set group highlighting
                  * attsel = S_GR_PR: set group priority
                  * attsel = S_GR_DT: set group detectability
                  * attsel = S_GR_PA: set grp pan, zoom, rotate
                  * the contents of only one of the following fields will be interpreted
                  * according to the value of 'attsel'. the other parameters can be set to 0.
                  */
Real      *mat;   /* pointer to 1th element of matrix for 'set group transformation' */
Bool      bool;   /* indicates either group shielding, visibibilty, highlighting, or detectability*/
Grppri    prio;   /* group priority */
Cindex    shield; /* shielding colour */
Bool      *pan;   /* pointer to three booleans indicating pan, zoom and rotate allowance */


Bool
dev_rdgr(ws) /* redraw a group */
Wss       *ws;


Bool
dev_igrt(ws, gr,  gr_mat)      /* inquire dynamic group transformation */
Wss       *ws;
Group     *gr;
Tmat      *gr_mat; /* contains on return the transformation matrix as it is
                      *dynamically  set on the workstation
                      */
```