объединенный
институт
ядерных
исследований
дубна

757/83

E11-82-782

M.Rudalies

SOFTWARE ISSUES AS RELATED TO THE ARITHMETIC PROCESSING CAPABILITIES OF AN INTELLIGENT GRAPHICS TERMINAL

Submitted to the Conference on Computer Graphics´83 (ČSSR June 1983)

1982

1. INTRODUCTION

The two major proposals for graphic standardization, the Graphic Standard Planning Committee's CORE [CORE] and the Graphical Kernel System - GKS [GKS] have been designed with the concept of device independence in mind. Consequently they include only suggestions as far as the capabilities of intelligent output devices or workstations are concerned.

A clever implementation of the INSERT function may be one aspect of how to optimally use the rather restricted main memory of such a device, in this case - a multiprocessor based Intelligent Graphics Terminal - IGT [IGT], which has been designed at the JINR in Dubna. Minimal space requirements however generally introduce some kind of overhead, in particular for the so-called arithmetics module [ARITHMOD], a highly specialized component of the IGT, capable of fast performing transformations of graphic objects.

We assume the reader familiar with the terminology of CORE and GKS, respectively, where both proposals diverge this will be either indicated or an own terminology will be used.

2. THE FUNCTION OF THE INSERT FUNCTION, OR MODELLING ASPECTS IN VIEWING SYSTEMS

CORE and GKS unanimously abolished modelling facilities in favour of a consistent viewing concept. It has become a general accepted philosophy to build modelling systems "on top" of viewing systems. Thus CORE and GKS agree that:

- No output functions are provided which retransfer the description of graphic objects into world coordinates space. Intermediate storing of objects on a Meta-file is accepted in normalized device coordinates (NDC) space.
- No extend function is provided, i.e., once a segment has been closed it may not be reopened. This implies that no more functions may be added to it, no more segments may be inserted into it.

However CORE and GKS disagree about introducing in their standards an INSERT function, i.e., a function which allows a source-object to become part of a target-object with the consequence that all subsequent operations on the target-object apply to the source-object too:

- In CORE no function is provided to insert (copy) a segment into the currently open segment, as this would imply an extend function or a symbol system - thus introducing some kind of modelling flavour - and require segment interpretation for resolving different view surface conflicts. For a discussion of this argument see issue 3.14 in [CORE].
- GKS provides an INSERT function with the following restrictions: A segment may only be inserted into the open segment and must not be the open segment itself. Thus recursive insertion is avoided while arbitrary hierarchies of insertions may be achieved.

However GKS pays the introduction of the INSERT function with a certain design inconsistency. Before a segment may reenter the viewing pipeline three actions have to occur:

(1) The segment transformation and an insert transformation have to be performed. Note that this is the only moment in GKS where a transformation of coordinates from NDC space to NDC space without subsequent transformation to (physical) device coordinates (DC) space takes place.

(2) Settings of attributes have to remain unaffected by the insertion process. This requires some overhead for saving and restoring their respective values before and after invocation of the INSERT function.

(3) All clipping rectangles contained in the inserted segment have to be ignored. Although this allows the user to replace possibly mismatched clipping rectangles (note that in GKS clipping rectangles are bound to their primitives for lifetime), it requires the system to interpret the inserted segment for the only purpose of eliminating enclosed clipping rectangles. It is interesting to remark that CORE would not have run this problem as it forbids the viewing point to change while a segment is open, i.e., one and only one "clipping rectangle" is valid for a CORE segment.

GKS leaves it up to the implementor how and where the Device Independent Segment Storage - which has to contain a segment to be inserted into the open segment - is organized, as long as all GKS functions are performed correctly. The following three approaches should contain all necessary informations for proper functioning of the INSERT function:

Объединенный институт ядерных исследований БИБЛИОТЕКА

(1) Copying Approach

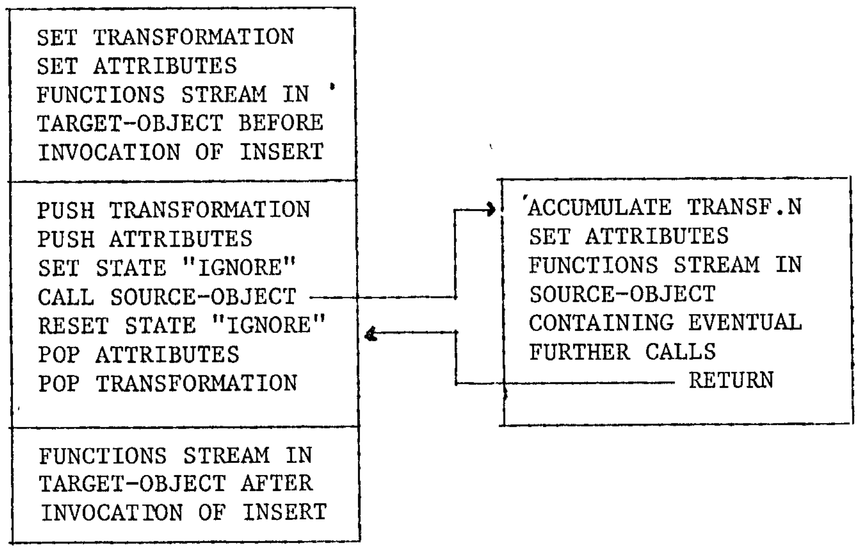
All functions contained in the source-object are simply copied into the target-object:

```
+------------------------------+
| SET TRANSFORMATION           |     ... actions contained in frames
| SET ATTRIBUTES               |         are performed with every
| FUNCTIONS STREAM IN          |         evaluation of the object
| TARGET-OBJECT BEFORE         |
| INVOCATION OF INSERT         |
+------------------------------+
 SAVE ATTRIBUTES
 SET TRANSFORMATION
 SET STATE "IGNORE CLIPPING RECTANGLES"

+------------------------------+
| SET ATTRIBUTES               |
| FUNCTIONS STREAM IN          |
| SOURCE-OBJECT                |
+------------------------------+
 RESTORE ATTRIBUTES                  ... actions out of frame are
 RESET TRANSFORMATION                    performed once only
 RESET STATE "IGNORE..."

+------------------------------+
| FUNCTIONS STREAM IN          |
| TARGET-OBJECT AFTER          |
| INVOCATION OF INSERT         |
+------------------------------+
```

With this approach the entire information contained in the source-object (with the exception of the clipping rectangles) is duplicated with each insertion command. Coordinates are transformed once during the copying operation. System overhead remains small but for saving and restoring the primitives attributes.

(2) Contextual Call

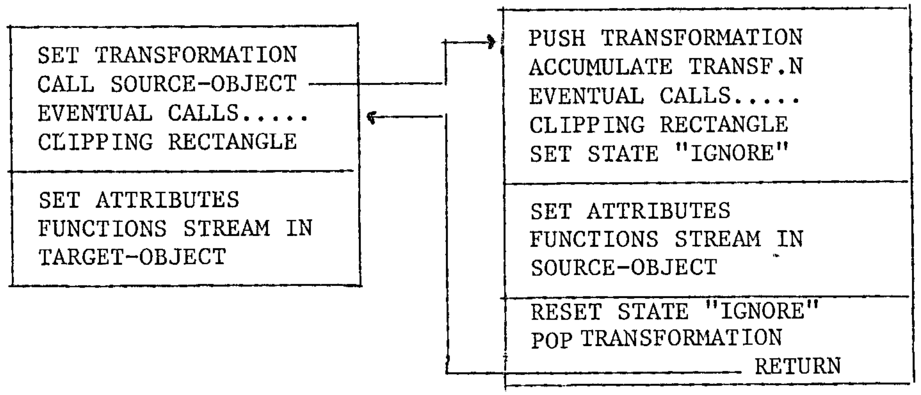
The source-object is called "within the context" of the target-object´s function stream:



This approach has minimal space requirements, however it introduces a considerable overhead, as with every call the source-object´s transformation and attributes settings have to be saved on push-down stacks and the elimination of clipping rectangles has to occur not only once but with every reevaluation of the target-object. Additional problems are caused by the fact that the target-object has to be interpreted when it has to be deleted, as some kind of reference (e.g., reference counter, pointer) has to be updated in the source-object.

(3) Call Out of Context

The call of the source-object does not reflect the point of invocation of the INSERT function:



This method requires slightly more overhead for storing some header information for each insertion of the source-object. On the other hand all calls may be processed in one block before (or after) processing the functions stream of the target-object. This eliminates the need for repeated save and restore operations for attributes, however requires the setting of the clipping rectangle at the time of invocation of the INSERT function to be stored along with the source-object. Deleting the target-object causes a simple pointer operation in the header of the source-object.

3. AN ARITHMETICS MODULE´S VIEW OF THE DATA STRUCTURE

The arithmetics module of the IGT [ARITHMOD] is equipped with some special hardware for fast performing transformations of graphic objects. However when attempting to offload the control module of the IGT (which is occupied with I/O and storage management) and the display module (in refresh mode continuously busy with the display process) it additionally has to understand the data structures of graphic objects. Thus the arithmetics module has to

- proceed all functions contained in a graphical object like settings of attributes, clipping rectangles, etc.,
- understand the storage representation of compound objects, i.e., objects with references to other objects as described in approach (2) and (3) of the previous chapter,
- perform push and pop operations of transformations preferable on a stack in its own memory (e.g., when accumulating GKS´ transformation matrices),
- interact with the control module for reclaiming and returning variable portions of the main memory,
- provide immediate visibility (deferral state "ASAP" in GKS) for the display process.

REFERENCES

[ARITHMOD] Leich H. The Architecture of an Arithmetics Module for the Intelligent Graphics Terminal, to appear in: Almanac of Conference on Computer Graphics´83, Bratislava.

[CORE] Status Report of the Graphic Planning Committee. ACM/SIGGRAPH, Computer Graphics, vol.13, 3, Aug.1979.

[GKS] Information Processing Graphic Kernel System (GKS). Functional Description, Version 7.0, Jan. 1982.

[IGT] Leich H., Levchanovsky V., Prikhodko V. Multimikroprozessorsystem zur Steuerung eines Intelligenten Graphischen Terminals (IGT), submitted to Nachrichtentechnik Elektronik.

Received by Publishing Department
on November 17th 1982.

Рудалич М. E11-82-782
Проблемы программирования,
связанные с арифметическими возможностями
интеллектуального графического терминала

Функция "INSERT" и ее реализация являются одним из главных расхождений между двумя предложениями для графических стандартов CORE и GKS. Исследование ее семантики открывает некоторые проблемы ее реализации в контексте интеллектуального графического терминала с расширенными арифметическими возможностями.

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Препринт Объединенного института ядерных исследований. Дубна 1982

Rudalics M. E11-82-782
Software Issues as Related to the Arithmetic Processing Capabilities of an Intelligent Graphics Terminal

The INSERT function and its implementation are among the main issues of disagreement between the two proposals for a graphic standard, CORE and GKS, respectively. An investigation of its semantics reveals some of the implementation problems in the context of an Intelligent Graphics Terminal with advanced arithmetic processing capabilities.

The investigation has been performed at the Laboratory of Computing Techniques and Automation, JINR.

esearch. Dubna 1982