Science and Engineering Research Council
Rutherford Appleton Laboratory
Chilton DIDCOT Oxon OX11 0QX

RAL-93-088

GKS-9x: An Experimental Application

Lj B Damnjanović and D A Duce

November 1993

GKS-9x: An Experimental Application

Lj.B. Damnjanović , D.A. Duce
Rutherford Appleton Laboratory
Chilton, Didcot, Oxon OX11 0QX, UK

October 28, 1993

1 Introduction

GKS-9x refers to the current state (Draft International Standard) of the revision of GKS which was published as an ISO standard for computer graphics programming in August 1985 [1]. In the revision process, GKS:1985 has been extended with new concepts and functionality [2]. Among them are the well-defined concept of an NDC picture, consisting of a sequence of output primitives which the application creates and workstations view, and the selection mechanism, used to select a subsequence of primitives in the NDC picture which are to be edited or displayed.

A part of the GKS-9x standard was implemented at Rutherford Appleton Laboratory to explore the feasibility of the new concepts as well as the behaviour of GKS-9x. The implementation was described and it was shown that the performance can be improved if the selection structure and the nameset representation optimizations are included [3, 4].

This paper explores the effects of the number of primitives manipulated in an application on the experimental GKS-9x implementation performance. It also identifies the major factors which increase the GKS-9x function execution times. Finally, the improvement in speed obtained when optimization was included in the traversal process is analyzed and discussed.

The original GKS-9x implementation [4] was extended with the SET VIEW function to investigate viewing mechanism introduced in GKS-9x, as it is a generalization of the GKS-3D and PHIGS mechanisms [3].

2 The Experimental Application

2.1 Overview

The main aim of the work reported in this paper was to explore how the number of primitives handled in an application affects the GKS-9x performance. Accordingly, an application was developed which could be instantiated to manipulate different number of primitives. Moreover, the application had two versions: interactive and non-interactive. The interactive version allows the user to exercise GKS-9x functions on-line with user defined selection

criteria, namesets and views in order to get better insight in the system behaviour. The non-interactive version is used to carry out performance measurements.

The application was made based on the *theatre* application described in [3, 4] which displays the seating plan of a theatre consisting of 1154 seats. This application can be instantiated to manipulate different numbers of theatre patterns, all with different coordinates. The implementation work was carried out using the ANSI C programming language.

The application exercises the following functions:

SET WORKSTATION SELECTION CRITERION
SET VIEW SELECTION CRITERION
SET VIEW
SET NDC PICTURE PRIMITIVE ATTRIBUTE
ADD SET OF NAMES TO NDC PICTURE
REMOVE SET OF NAMES FROM NDC PICTURE
DELETE PRIMITIVES
REORDER NDC PICTURE - old version
REORDER NDC PICTURE

The REORDER NDC PICTURE function moves the subsequence of primitives in the NDC picture which satisfy the source selection criterion to either the start or the end of the NDC picture subsequence which satisfy the reference selection criterion. However, the REORDER NDC PICTURE function had a different definition in the previous version of the GKS-9x revision. The REORDER NDC PICTURE - old version function moves the subsequence of primitives in the NDC picture which satisfy the selection criterion to either the start or the end of the NDC picture.

Functions SET VIEW and SET VIEW SELECTION CRITERION allow observation of visual effects of the viewing mechanism introduced in GKS-9x. In GKS-3D and PHIGS a single primitive can only be subjected to a single viewing operation. In GKS-9x, multiple view transformations can be defined, and a primitive can be displayed in any of the defined views. Selection criteria associated with the views determine which primitives appear in which views of the NDC picture.

This section describes the common parts of the two versions (interactive and non-interactive) of the application.

2.2 Theatre Model

As described in [4], the *theatre* application displays the seating plan of a theatre and allows seat reservations to be made and queried by manipulating namesets and selection criteria. This is considered a reasonable illustration of the power of the nameset and selection criterion mechanism. The seating plan comprises 1154 output primitives with different namesets, as shown below.

Nameset	*Number of primitives*
{AMPHITHEATRE, ROW(i), SEAT(j), X(m), Y(n)}	243
{BALCONY, ROW(i), SEAT(j), X(m), Y(n)}	94
{TIER, LEFT, ROW(i), SEAT(j), X(m), Y(n)}	16
{TIER, GRAND, ROW(i), SEAT(j), X(m), Y(n)}	17
{TIER, RIGHT, ROW(i), SEAT(j), X(m), Y(n)}	16
{BOX, SEAT(j), X(m), Y(n)}	12
{ORCHESTRA,STALLS,LEFT,ROW(i),SEAT(j),X(m),Y(n)}	168
{ORCHESTRA,STALLS,MIDDLE,LEFT,ROW(i),SEAT(i),X(m),Y(n)}	210
{ORCHESTRA,STALLS,MIDDLE,RIGHT,ROW(i),SEAT(j),X(m),Y(n)}	210
{ORCHESTRA,STALLS,RIGHT,ROW(i),SEAT(j),X(m),Y(n)}	168

The indices i and j are instantiated for each row and seat number. ROW(i) is a Roman numeral in the range I - XXI. SEAT(j) is an Arabic numeral in the range 1 - 26. This theatre seating plan is the basic set of output primitives used to create a model to be manipulated in the application. The basic *theatre* pattern is shown in Figure 1. It is possible to define a model with a different number of theatre patterns along x-axis and y-axis and to make the corresponding instance of the application. Two parameters stored in an include file `num_theatre.h` control this. They have to be set to the appropriate values. According to the number of theatre patterns defined, the application generates the required number of primitives, each with a different nameset, and GKS-9x stores them in an NDC picture. Two names, X(m), Y(n), are added to each primitive depending on the position of the particular theatre layout. One name reflects the position along the x-axis and the other the position along the y-axis. The range of these names is X1 - X20 and Y1 - Y20. For example, for the position in the lower left corner of the workstation display surface, names X1 and Y1 are added:

{ORCHESTRA, STALLS, RIGHT, IV, 3, X1, Y1}

There are 12 fill area primitives tagged by {BOX, SEAT(j), X(m), Y(n)}. All other 1142 primitives are polylines.

Boxes are placed on the sides: on the left tagged from 1-yellow to 6-cyan; on the right: from 7-yellow to 12-cyan. The function SET NDC PRIMITIVE ATTRIBUTE has only been implemented for polyline attributes and so cannot be applied to the fill area primitives tagged by {BOX, SEAT(j), X(m), Y(n)}.

2.3 Theatre Application

To build a model which is initially stored in the NDC picture, the following modules are used:

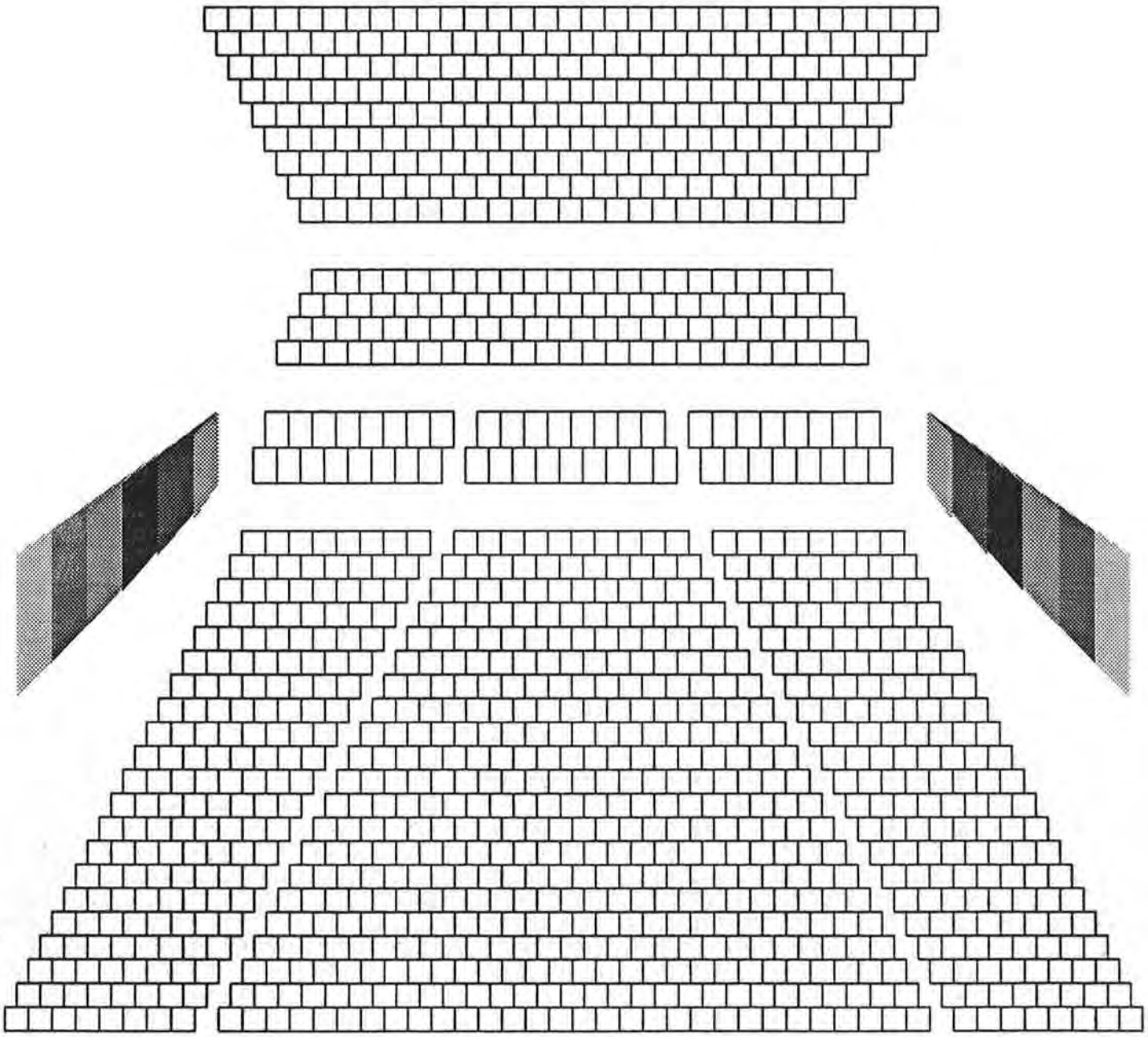

Figure 1: The seating plan of a theatre

```
theatre.c
nsi.c
theatre_I.c
copy_theatre.c
num_theatre.h
orcstalls_demo.h
points_mat_ext.h
```

A model can consist of several theatre layouts. For example, a model consisting of four theatre patterns is shown in Figure 2. The function `theatre` (`theatre.c` module) calls the function `nsi` (`nsi.c` module) to initialize namesets used to tag output primitives and then the function `seatplan` (`theatre_I.c` module) to generate all primitives necessary to create the required model.

The function `seatplan` calls two functions: `cp_scale_pnts` and `cp_shift_pnts`, stored in `copy_theatre.c` module, in order to initialize the arrays of type Ppoint to contain point values for all output primitives which are to be generated. Then the function `seatplan` generates all output primitives.

For simplicity, an example of a theatre layout consisting of three seats is used here to illustrate how a model is created in the application. The same applies for all 1154 output primitives.

Include file `orcstalls_demo.h` stores several two dimensional arrays of type Ppoint which contain point values for drawing a theatre layout pattern of maximum possible size on a workstation. Each array contains data for several seats. One dimension is the number of seats (3 in the following example), and the other is 5, defining the number of points per seat. The example shows an array containing points for 3 seats:

```
static Ppoint stallsmln14[3][5] = {

{{ 0.47, 0.09}, { 0.49, 0.09}, { 0.49, 0.07}, { 0.47, 0.07}, { 0.47, 0.09}},
{{ 0.48, 0.11}, { 0.50, 0.11}, { 0.50, 0.09}, { 0.48, 0.09}, { 0.48, 0.11}},
{{ 0.49, 0.13}, { 0.51, 0.13}, { 0.51, 0.11}, { 0.49, 0.11}, { 0.49, 0.13}}
};
```

Include file `num_theatre.h` contains the number of theatre patterns which are to be generated along x-axis and y-axis to build the model. For example, when 20 theatre patterns are required:

```
/* Number of theatres in a thm application */

#define NUM_X 4
#define NUM_Y 5
```

The function `cp_scale_pnts` copies all arrays from `orcstalls_demo.h` into four dimensional arrays stored in `points_mat_ext.h`. Two indices correspond to the number of theatre layouts along x-axis and y-axis. The other two correspond to indices of arrays in `orcstalls_demo.h`.

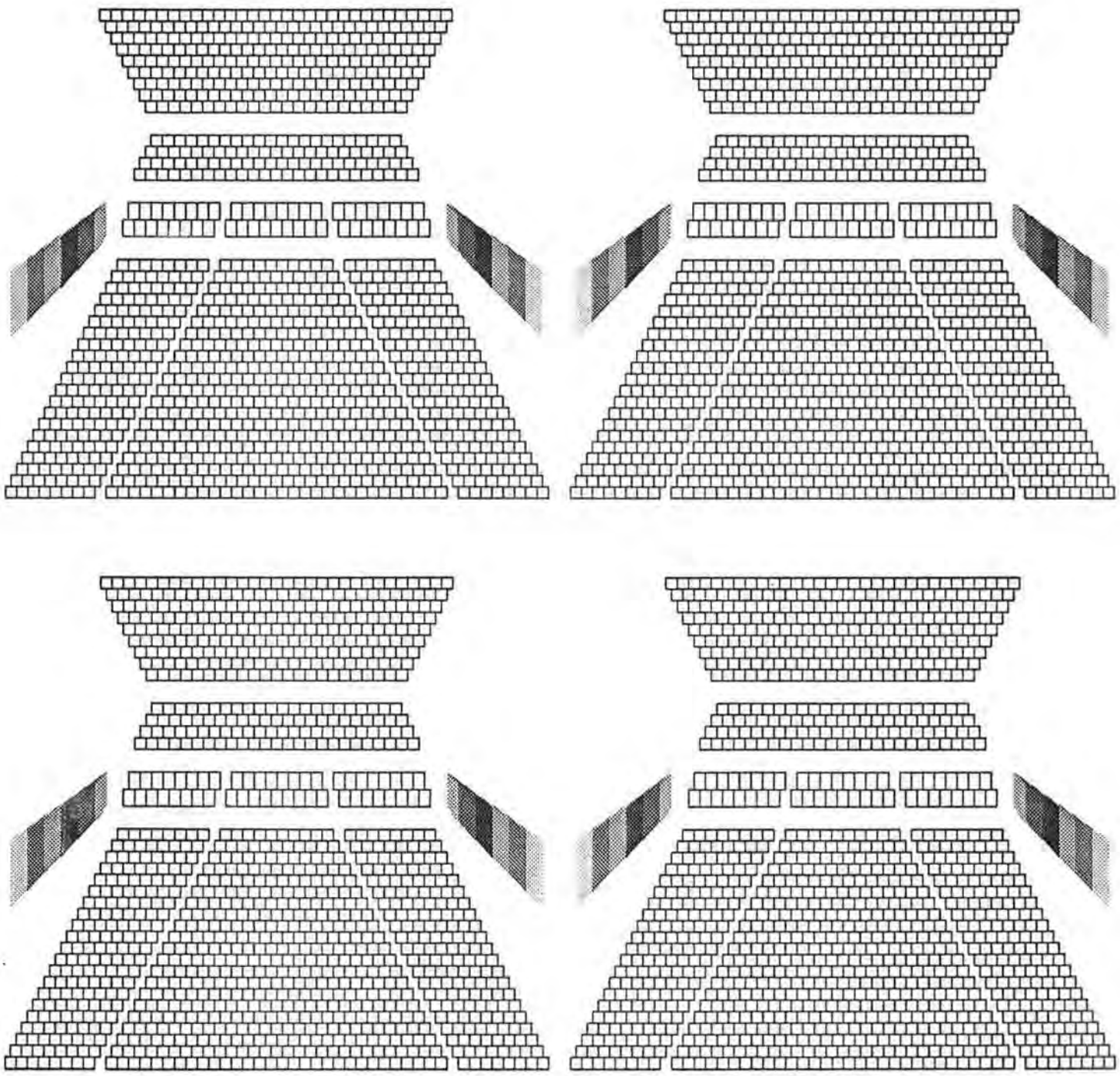

Figure 2: Theatre layout: a model consisting of four theatre patterns

The function `cp_scale_pnts` copies all points scaled with appropriate scale factor, which depends on the number of theatre patterns required, and is calculated as follows:

```
scal=1.;
fl = NUM_X;
fl1=NUM_Y;
if(NUM_X >= NUM_Y)
        scal = scal/fl;
else
        scal = scal/fl1;
```

The function `cp_scale_pnts` copies, for example, the array `stallsmln14[3][5]` as follows:

```
for(i=0; i<3; ++i) {
      for(j=0; j<5; ++j) {
            stallsmln14_I[0][0][i][j].x=scal * stallsmln14[i][j].x;
            stallsmln14_I[0][0][i][j].y=scal * stallsmln14[i][j].y;
      }
}
```

Therefore, the function `cp_scale_pnts` stores a scaled theatre layout (for displaying in the lower left corner of the picture) into four dimensional arrays.
The function `cp_shift_pnts` initializes all other four dimensional arrays. It copies points created by `cp_scale_pnts` NUM_X * NUM_Y -1 times to corresponding arrays' elements modifying coordinates by appropriate `shift_x` and `shift_y` values. For the previous example it is:

```
void cp_shift_pnts()
{
int i, j, m, n;
float shift_x, shift_y, x, y, x1, y1;
x = 1.;
x1 = NUM_X;
y = 1.;
y1 = NUM_Y;
x = x/x1;
y = y/y1;
for(m = 0; m<NUM_Y; ++m) {
     shift_y = y * m;
     for(n = 0; n<NUM_X; ++n) {
          shift_x = x * n;
          if(m == 0 && n == 0) ;
          else {
               for(i=0; i<3; ++i) {
                   for(j=0; j<5; ++j) {
                      stallsmln14_I[m][n][i][j].x =
                                   stallsmln14_I[0][0][i][j].x + shift_x;
                      stallsmln14_I[m][n][i][j].y =
                                   stallsmln14_I[0][0][i][j].y + shift_y;
```

```
                    }
                }
            }
        }
}
}
```

Function **seatplan** generates all output primitives after all four dimensional arrays have been initialized:

```
void seatplan()
{
#define NUM_POINTS5 5
Pint i, m, n, num_x, num_y;
Gpattr_value at;
cp_scale_pnts();
cp_shift_pnts();
num_x = NUM_X;
num_y = NUM_Y;

gadd_nameset(&ostalls_list);
gadd_nameset(&osmleft_list);

for(m = 0; m< NUM_Y; m++) {
        gadd_nameset(&y_list[m]);
        for(n = 0; n< NUM_X; n++) {
             gadd_nameset(&x_list[n]);
             gadd_nameset(&seatnum[13]);
             for(i = 0; i< (NUM_ORAWS-17); i++) {
                  gadd_nameset(&rawnum[i+1]);
                  plistm15[m][n][i].num_points = NUM_POINTS5;
                  plistm15[m][n][i].points = &stallsmln14_I[m][n][i][0];
                  gcreate_out_prim(SETOFPOLYLINES, &plistm15[m][n][i]);
                  gsubtr_nameset(&rawnum[i+1]);
             }
             gsubtr_nameset(&x_list[n]);
        }
        gsubtr_nameset(&y_list[m]);
}
}
```

All output primitives are tagged with different namesets in the way described earlier in this section.

2.4 GKS-9x

The implementation of a part of GKS-9x (described in [3, 4]) extended with the SET VIEW function is used here. This version of the implementation uses Sun PHIGS 2.0 to allow use of PHIGS and X Windows or X toolkits together.

SET VIEW is implemented without view scissor and with the following parameters:

```
void gset_view( Pint      ws_id,           /* workstation identifier */
                Pint      vwn,             /* view index             */
                Ppoint    *view_ref_point, /* view reference point   */
                Pvec      *view_up_vec,    /* view up vector         */
                GNDC_win  *win,            /* NDC window             */
                GLDC_vp   *vp              /* LDC viewport           */
)
```

The GKS-9x data types have been extended by the type `Gview`:

```
typedef struct {                        /* VIEW */
        Gselect         *view_sel;      /* viewing selection */
        Gview_rep       *view_rep;      /* view representation */
} Gview;
```

The GKS-9x data type `Gwssel_list` has been revised so that member `wsview[NUMVWT]` is of type `Gview` instead of type `Gselect` as described in [4].

```
typedef struct wsel *Gwssel_list_ptr;

typedef struct wsel {
        Pint            owsid;          /* workstation identifier  */
        Gselect         *wsvis;         /* visibility selection    */
        Gselect         *wshigh;        /* highlighting selection  */
        Gselect         *wsdetc;        /* detectability selection */
        Gview           *wsview[NUMVWT]; /* view representations    */
        Gpntwsf_list    *flagshead;     /* pointer to llwsf        */
        Gwssel_list_ptr nextwssel;      /*pointer to the next node */
} Gwssel_list;
```

To implement the SET VIEW function, the following PHIGS functions are used:

```
peval_view_ori_matrix
peval_view_map_matrix
pset_view_rep
pset_view_ind
```

To implement the 16 views allowed in GKS-9x, 16 entries in the PHIGS view representation table are used - one per view.

Two Motif drawing area widgets are used for two GKS-9x workstations. Workstations are of type `phigs_ws_type_drawable`.

As described in [4], the GKS-9x implementation can be compiled with or without PHIGS. The version with PHIGS is used for interactive applications, while the version without PHIGS (which also excludes SET VIEW function) is used for non-interactive applications.

The implementation of a part of GKS-9x (described in [3, 4]) includes `reorder_ndcp` and `reorder_prims`. The first corresponds to the former definition of REORDER NDC PICTURE. The second corresponds to the current definition of REORDER NDC PICTURE. However, this has been changed so that `reorder_ndcp` follows the current definition of REORDER NDC PICTURE.

3 Interactive Application

The interactive application, *thm*, displays the theatre layout on two GKS-9x workstations and allows sessions in which GKS-9x functions, selected interactively, are performed. A user can edit the NDC picture and change its visibility and appearance on workstations, in order to explore the behaviour of a GKS-9x system prototype.

To provide this, the application uses Motif and PHIGS together - Motif to create graphical user interface and Sun PHIGS 2.0 to produce GKS-9x graphics output. Two Motif drawing area widgets are used for two GKS-9x workstations.

The user interface provides easy selection of a function from a subset of GKS-9x functions.

A function is selected from a menu. A selection criterion has to be defined for all functions except SET VIEW. One more parameter has to be defined for some functions. The selected function is executed when correctly supplied with parameters. All available functions are executed with consequent redisplay of NDC picture. However, in some cases redisplay is not performed, for example, if SET PRIMITIVE ATTRIBUTE function is executed when the visibility selection criterion has value REJECTALL. When the execution of a function is completed, it is possible to select another one.

The *thm* main module contains the user interface implementation which is described in the following subsection. This section also describes how to use the *thm* application.

3.1 User Interface

The *thm* user interface is stored in the `theatre_motif_many.c` module. It contains the main function and a couple of other functions necessary to create a Motif interface. The main function calls `create_toplevel` function which creates all the Motif components. It opens GKS-9x, opens two workstations and calls `XtAppMainLoop` to wait for input and window events. Input events from the menus and text fields are handled by the widgets callback functions. Expose events are handled by the `redraw` function. It calls PHIGS function `redraw_all_structs`. This cause PHIGS to clear the workstation and redisplay all the posted structures. The GKS-9x implementation considered here creates and posts one

PHIGS structure for each open workstation. The structure contains all views of the visible output primitives.

The function `create_toplevel` creates a hierarchy of Motif widgets - the enclosing form, the menu bar, the menus, the text fields, two drawing areas with menu bars and menus and a large number of push-buttons. The hierarchy consists of over 100 widgets. The responses to input events from the menus or other widgets are carried out by 20 widget callback functions. They issue GKS-9x function calls, manage/unmanage other widgets, set appropriate variables or send error messages. This enables the application to change the image (display of NDC picture) in response to user selections from menus or other widgets.

3.2 Interaction

The *thm* application is an interactive program. It inputs user supplied character arrays representing selection criteria, namesets and points from Selection, Selection1, Nameset, Orientation and Mapping text fields. The application outputs character arrays representing error messages in Error text field.

A few functions stored in `getsel.c` and `crsel.c` modules parse an array of `Pchar` type representing a selection criterion and convert it to a selection criterion of type `Gselect` which is acceptable for GKS-9x functions.

Two functions stored in the `crnameset.c` module parse an array of `Pchar` type representing a nameset and convert it to a nameset of type `Gint_list`.

A few functions stored in the `view_transf.c` and `a_to_point.c` modules parse an array representing two points which are input from Orientation field, or an array representing four points input from Mapping field. In the first case they convert an array into two points of type `Ppoint`. In the second case they convert an array into a window of type `GNDC_win` and a viewport of type `GLDC_vp`.

In the GKS-9x implementation which is considered here, names are of `Pint` type. However, *thm* is an interactive application and selection criteria and namesets are entered as arrays of `Pchar` type. To obtain, for a character array representing a name, an integer, a hash function[5] is used. The function `nsi` which is used in the interactive application creates namesets in the following way:

```
static Pchar amphitheatre[]  = "AMPHITHEATRE";
        new_ar = (Pint*)malloc(SIZE_OF_INT);
        hash_tab(FIND_ENTER, amphitheatre, &n);
        *new_ar = n;
        amphitheatre_list.num_ints = 1;
        amphitheatre_list.ints = new_ar;
```

3.3 Starting the Interactive Application

When *thm* is started an X window is opened. It displays the main menu bar, Selection text field, Error text field and two Motif drawing area widgets used as two GKS-9x workstation

display surfaces. The NDC picture is displayed on both workstations (Figure 3) as the state of the GKS-9x system is:

NDC picture stores the predefined number of primitives.
Workstation 1 is open with initial values for all selection criteria.
Workstation 2 is open with initial values for all selection criteria.
View 0 is set to initial value.
Views 1 - 15 are set to different values.

The main menu bar is on the top of the window. It contains three buttons: Functions, Quit and Help.

Help - provides an explanation of how to use the application.
Quit - is used to exit the application.
Functions - provides a pull-down menu to choose a function to be performed.

3.4 Functions

To begin a session of exercising GKS-9x functions, press the Functions button. Then press a button, labelled by a function name, from the displayed menu which includes the following functions:

set workstation selection - workstation1
set workstation selection - workstation2
set view selection - workstation1 (view 0,...,view 15, view ALL)
set view selection - workstation2 (view 0,...,view 15, view ALL)
set view - workstation1 (view 1,...,view 15, view ALL)
set view - workstation2 (view 1,...,view 15, view ALL)
set ndc primitive attribute - colour (red, green, blue, cyan, yellow, magenta)
set ndc primitive attribute - type (solid, dashed, dotted, dotdash)
set ndc primitive attribute - width (thin, normal, thick)
add nameset to ndc picture
remove nameset from ndc picture
delete primitives
reorder ndc picture (old version) - front
reorder ndc picture (old version) - back
reorder ndc picture - front
reorder ndc picture - back

The next step, for all functions except set view, is to define a selection criterion in Selection text field . Therefore, when any available function, except set view is chosen, enter a selection criterion in the field labelled Selection. The chosen function will be executed or another field will appear. The latter case occurs when one of the following functions is chosen:

add nameset to ndc picture
remove nameset from ndc picture
reorder ndc picture

In the cases add/remove nameset to/from ndc picture, a field labelled Nameset appears between the fields Selection and Error. Enter a nameset in this field. The chosen function will be executed.

In the case of reorder ndc picture, a field labelled Selection1 appears between the fields Selection and Errors as shown in Figure 4. Enter a selection criterion in this field. The chosen function will be executed.

Fill area primitives (tagged by BOX) were constracted to partially overlap each other in the theatre layout in order to be suitable to exercise the REORDER NDC PICTURE function. The fill area primitive tagged by {BOX, 1} completely overlaps all other fill area primitives on the left side of the theatre, and is partially overlapped by them. The fill area primitive tagged by {BOX, 2} completely overlaps four fill area primitives tagged by: {BOX, 3}, {BOX, 4}, {BOX, 5} and {BOX, 6} and is partially overlapped by them, and so on. The fill area primitive tagged by {BOX, 6} (the last one in the sequence of the fill area primitives on the left) partially overlaps five fill area primitives and is completely overlapped by them. The similar is on the right side. When overlapped primitives are displayed they may obscure each other, or some parts of each other, depending on their positions in the sequence of primitives in the NDC picture. The order of fill area primitives follows the increase of value of the numeral by which they are tagged when the NDC picture is created. Therefore, the fill area primitive tagged by {BOX, 2} obscures the fill area primitive tagged by {BOX, 1}. The fill area primitive tagged by {BOX, 3} obscures the fill area primitives tagged by {BOX, 1} and {BOX, 2} and so on until the fill area primitive tagged by {BOX, 6} which obscures all others and itself is completely displayed.

In the theatre layout in Figure 4, a primitive labelled {BOX, 2} obscures completely primitives labelled by {BOX, 3}, {BOX, 4}, and {BOX, 5} and the partially primitive labelled {BOX, 1}, while itself is partially obscured by the primitive labelled {BOX, 6}. To achieve this, the reorder ndc picture - front function was chosen from the menu. Section 3.6 defines the syntax of specifying selection criteria. The source selection criterion, c{BOX, 2}, has been defined in the field Selection. The reference selection criterion, o(c{BOX, 5}, o(c{BOX, 3}, c{BOX, 4}), has been defined in the field Selection1. The primitive tagged {BOX, 2} has been moved to the position after the primitive tagged by {BOX, 5}.

When the set view function is chosen, a field labelled Orientation appears instead of the field labelled Selection as shown in Figure 5. Enter two points in this field which define a view reference point and a view up vector direction. Then a field labelled Mapping appears between the fields Orientation and Errors. Enter four points in this field. The first two define an NDC window and the last two an LDC viewport.

The procedure can be repeated from the beginning or from entering function parameters when previously selected function is assumed.
The message:

CHOOSE Function PLEASE !

is displayed in the Error field, when a function parameter has been defined but a function has not been chosen.

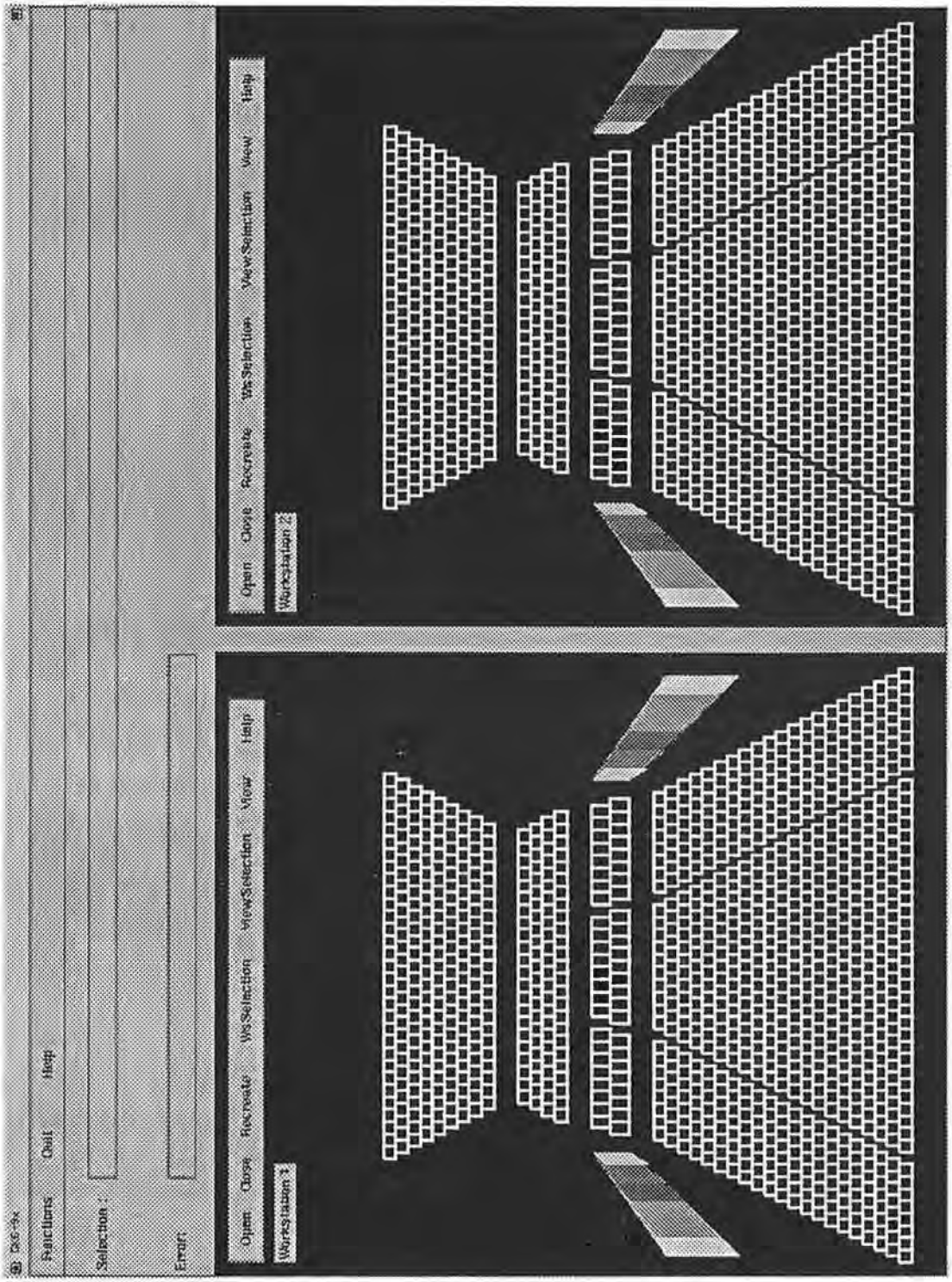
Functions
Quit
Help
Selection :
Error
Open
Close
Recreate
WsSelection
ViewSelection
View
Help
Workstation 3
Open
Close
Recreate
WsSelection
ViewSelection
View
Help
Workstation 2

Figure 3:

3.5 Nameset

A nameset is defined either as a part of a selection criterion or as an argument of add/remove nameset to/from ndc picture functions. In the later case, a nameset should be entered in the field labelled Nameset. Names contained in a nameset are enclosed in a pair of curved brackets and separated by commas. The following characters are allowed in names:

Uppercase letters	A - Z
Lowercase letters	a - z
Numbers	0 - 9
Underscore character	_

Examples:

{12, Blue, Balls}
{12_Blue_Balls}

3.6 Selection and Selection1

A selection criterion can be entered either in the Selection field or in the Selection1 field. A selection criterion is defined in terms of comparison and logical operations as follows:

s	SELECTALL
r	REJECTALL
c{}	CONTAINS
i{}	ISIN
e{}	EQUALS
n()	NOT
a()	AND
o()	OR

Curly brackets enclose a nameset. A nameset is an argument of the comparison operations: c, i and e. Brackets enclose arguments of the logical operations: n, o, and a. An argument of n is one selection criterion, while arguments of a and o are two selection criteria.

Examples:

c{12, Blue, Balls}
i{12, Blue, Balls}
e{12, Blue, Balls}
n({12, Blue, Balls})
o(e{12}, i{Blue, Balls})
a(c{12}, c{Blue, Balls})

Functions
Quit
Help
Selection :
c[BOX2]
Selection1 :
Error :
Open
Close
Recreate
WsSelection
View Selection
View
Help
Workstation 1
Open
Close
Recreate
WsSelection
View Selection
View
Help
Workstation 2

Figure 4:

3.7 Orientation

When the SET VIEW function with a workstation identifier and view index has been chosen from the Functions menu, view orientation and view mapping matrices have to be defined.

The specified view transformation on the workstation is defined by the orientation matrix and mapping matrix. For each view defined (except view 0) by view index, the application may redefine the origin and orientation of the NDC coordinate system. It should be defined such that it is the most appropriate for the view mapping. The *view reference point* defines the new origin to be used and a vector from this point called the *view up vector* specifies the new Y-direction of the axes. The view up vector is defined by one point, as the other one is always the origin. Effectively the view up vector defines the angle of rotation about the view reference point for objects contained in the NDC picture. Two points which define the view reference point and the view up vector are sufficient to calculate the orientation matrix.

Therefore, two points have to be supplied in the field labelled Orientation: the first of which defines the view reference point, and the second the view up vector. Both points are in the range [-7, +7]. Each point has to be defined in a pair of curly brackets, and their X and Y coordinates must be separated by a comma. For example in Figure 5, in field Orientation (for Workstation 2) the following points are defined:

{0.5, 0.5} {1, -1}

3.8 Mapping

The View Mapping transformation is a window to viewport mapping from NDC to LDC space. A window defines a rectangular area to be mapped onto a viewport which is another rectangular area. Each is defined by two points representing the lower left corner and the opposite one.

Therefore, four points have to be supplied in the field Mapping. The first two define the window in NDC space in the range [-7, +7], which is to be mapped onto the viewport in LDC space defined by the last two points (in the range [0, 1]). Each point has to be defined in a pair of curved brackets, and their X and Y coordinates must be separated by a comma. For example in Figure 5, in field Mapping (for Workstation 2) the following points are defined:

{-0.7, -0.7} {0.7, 0.7} {0, 0} {1, 1}

3.9 Errors

All error messages are reported in the field labelled Error. Here are some examples:

Number of left and right brackets must be equal !

Left bracket (is expected after logical operation !

A name contains a character which is not allowed !

Workstation viewport should be in range [0, 1] !

3.10 Workstation 1 and Workstation 2

Both Workstation 1 and Workstation 2 are open initially and display NDC picture. You can close each workstation, open it again, recreate the initial NDC picture and inspect a workstation's selection criteria and views by selecting one of the available buttons:

Open - opens the workstation.
Close - closes the workstation.
Recreate - recreate initial NDC Picture and display it on open workstations.
WsSelection - shows the current workstation DISPLAY selection criterion.
ViewSelection - shows the current view selection criteria.
View - shows the current view when a button (view identifier) from a displayed menu is selected.

An error message is reported (in the field labelled Error) when an attempt is made to open/close a workstation which has already been opened/closed.

4 Non-Interactive Application

To explore how GKS-9x function execution times vary for NDC pictures containing different numbers of primitives, an application *thm_time* was constructed. It differs from the *thm* application described earlier in the `theatre_motif_many.c` module. It was replaced in *thm_time* by the `theatre_time.c` module. This module contains the main function in which each GKS-9x function considered is executed 1000 times. It is given in Appendix 1. The GKS-9x functions considered are:

SET NDC PRIMITIVE ATTRIBUTE	`gset_ndc_prim_attr`
REMOVE SET OF NAMES FROM NDC PICTURE	`gremove_nameset_ndc_picture`
ADD SET OF NAMES TO NDC PICTURE	`gadd_nameset_ndc_picture`
REORDER NDC PICTURE	`greorder_ndcp`
SET WORKSTATION SELECTION CRITERION	`gset_ws_sel_crit`
SET VIEW SELECTION CRITERION	`gset_view_sel_crit`
DELETE PRIMITIVES	`gdel_prims`
CREATE OUTPUT PRIMITIVE	`gcreate_out_prim`

5 Performance

The execution times for a group of GKS-9x functions were measured for four selection criteria. Times were measured for six different NDC pictures (containing different number of primitives) for each selection criterion.

Functions
Quit
Help
Orientation:
Mapping:
Error:
Open
Close
Recreate
WsSelection
ViewSelection
View
Help
Workstation 1
Open
Close
Recreate
WsSelection
ViewSelection
View
Help
Workstation 2

Figure 5:

Times were measured using the Unix (TM) C library function *ftime*. The application was run on an overwise empty Sun 4/75 FGX-32 SPARCstation 2, with 32Mbyte memory (28 MIPS). Graphical output was not produced by the version of the implementation on which timing trials were made. One workstation was open.

The results are presented in several tables. There is one table per selection with six columns, each of which contains the execution times for different NDC pictures.

The execution times for `reorder_ndcp` functions were measured for both values of the position parameter (FRONT, BACK).

Times in tables 1 - 4 include NDC picture traversing time in all examples.

For the pair of functions: `gdel_prims` and `gcreate_out_prim`, a combined time is given in table 1 (includes two NDC picture traversing times). Tables 2-4 do not contain times for the `gdel_prims` function because the selection criteria used select more than one primitive and measuring these times would be inconvenient.

The execution times for all functions except `set_ws_sel_crit` and `set_view_sel_crit` were measured with initial values for all selection criteria.

The selection criterion SELECTION1 is :

CONTAINS{ORCHESTRA, STALLS, MIDDLE, LEFT, I, 1, X1, Y1}

SELECTION1 selects one primitive in the theatre layout placed in the lower left corner, as shown in Figure 6.

Table 1

Times (ms) for SELECTION1 for 6 different NDC pictures						
	number of primitives in NDC picture					
GKS function	1154	2308	6924	10386	16156	23080
gset_ndc_prim_attr	24.6	48.1	143.2	214.0	332.4	474.5
gadd_nameset _ndc_picture	24.7	48.4	143.4	213.9	332.3	473.6
gremove_nameset _ndc_picture	24.7	48.4	143.4	213.9	332.3	473.6
greorder_ndcp, FRONT	30.8	56.3	167.3	250.1	388.3	554.4
greorder_ndcp, BACK	42.0	71.0	186.9	273.1	418.4	591.7
gset_ws_sel_crit	13.2	26.1	78.2	117.2	205.0	260.6
gset_view_sel_crit	28.5	56.8	170.5	255.2	397.4	567.3
gdel_prims gcreate_out_prim	47.6	94.7	282.2	422.1	656.7	937.6

SELECTION2 is :

OR(
CONTAINS{AMPHITHEATRE, I, 1, X2, Y1},
CONTAINS{ORCHESTRA, STALLS, MIDDLE, LEFT, I, 1, X1, Y1})

SELECTION2 selects two primitives in the theatre layout. One primitive is the same selected by SELECTION1, and the other belongs to the theatre layout placed next to the one in the lower left corner along x-axes, as in Figure 7.

Table 2

Times (ms) for SELECTION2 for 6 different NDC pictures						
	number of primitives in NDC picture					
GKS function	1154	2308	6924	10386	16156	23080
gset_ndc_prim_attr	34.5	71.5	219.5	344.9	514.5	749.2
gadd_nameset _ndc_picture	34.8	75.9	219.9	330.7	514.9	753.6
gremove_nameset _ndc_picture	34.8	75.9	219.9	330.7	514.9	753.6
greorder_ndcp, FRONT	36.0	74.2	226.4	339.4	529.8	757.9
greorder_ndcp, BACK	49.1	88.7	246.0	374.9	560.2	795.8
gset_ws_sel_crit	20.6	44.1	138.6	209.9	326.9	468.7
gset_view_sel_crit	36.1	75.2	231.0	348.1	542.1	775.9

SELECTION3 is :

```
OR(
   CONTAINS{AMPHITHEATRE, X1, Y1},
   CONTAINS{STALLS, X1, Y1})
```

SELECTION3 selects 999 primitives in the theatre layout placed in the lower left corner, as in Figure 8.

Table 3

Times (ms) for SELECTION3 for 6 different NDC pictures						
	number of primitives in NDC picture					
GKS function	1154	2308	6924	10386	16156	23080
gset_ndc_prim_attr	46.0	87.3	252.0	375.0	581.0	831.0
gadd_nameset _ndc_picture	217.3	274.7	439.8	589.5	753.4	1020.3
gremove_nameset _ndc_picture	217.3	274.7	439.8	589.5	753.4	1020.3
greorder_ndcp, FRONT	57.5	100.5	270.8	395.5	606.6	859.4
greorder_ndcp, BACK	59.9	103.7	277.4	407.7	627.0	886.1
gset_ws_sel_crit	41.3	69.2	180.6	263.8	403.4	571.3
gset_view_sel_crit	43.8	86.8	259.3	389.0	628.2	863.6

SELECTION4 is :

```
OR(
   OR(
      OR(
         AND(CONTAINS{ORCHESTRA, STALLS, I, X1, Y1}, NOT(CONTAINS{MIDDLE}),
         CONTAINS{ORCHESTRA, STALLS, MIDDLE, I, X2, Y1}),
      OR(
         CONTAINS{AMPHITHEATRE, I, 1, X2, Y1},
         CONTAINS{ORCHESTRA,STALLS,MIDDLE,LEFT, I, 1, X1, Y1})),
   CONTAINS{TIER, GRAND, I, 1, X1, Y1})}
```

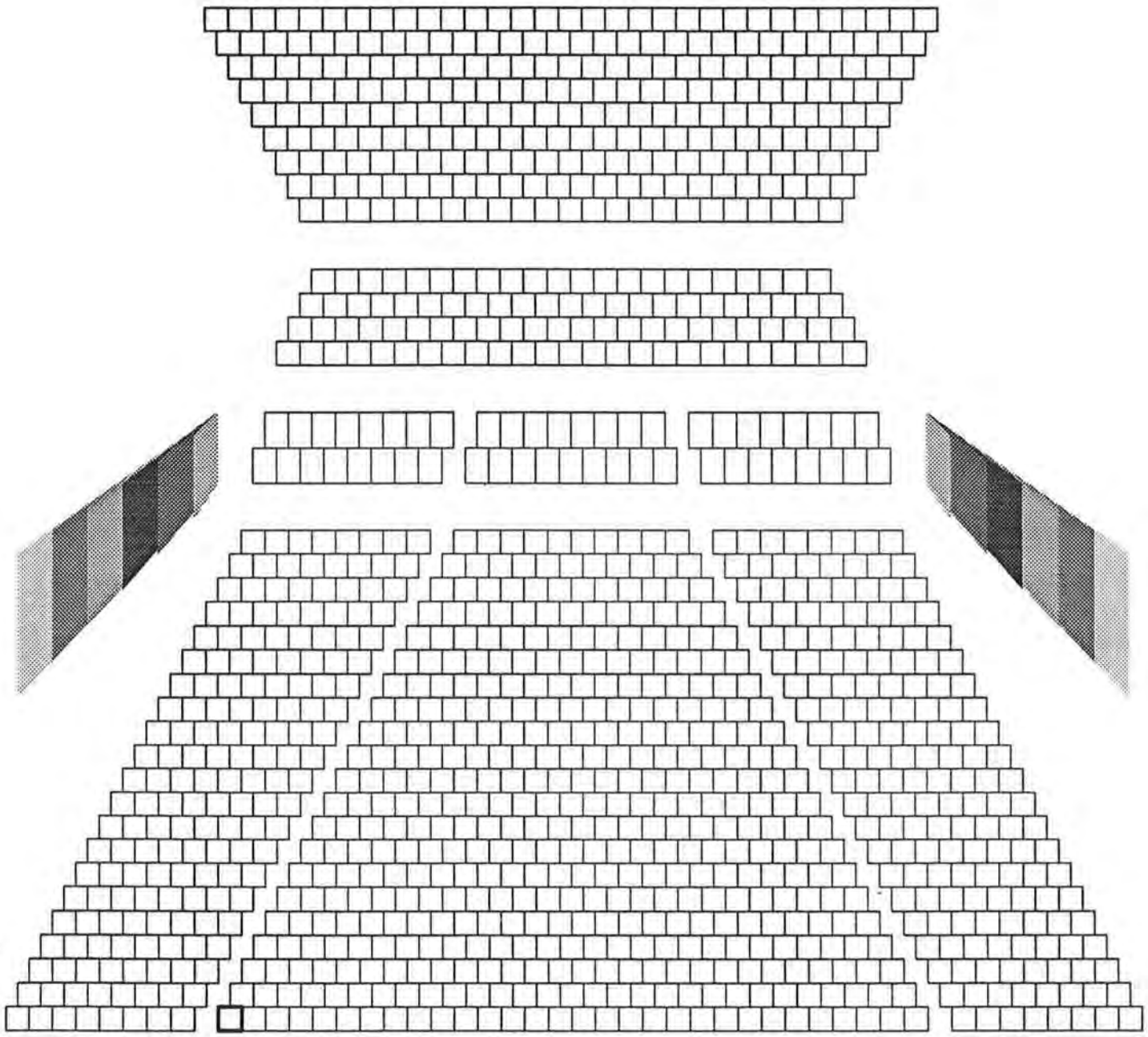

Figure 6: Theatre Layout - SELECTION1

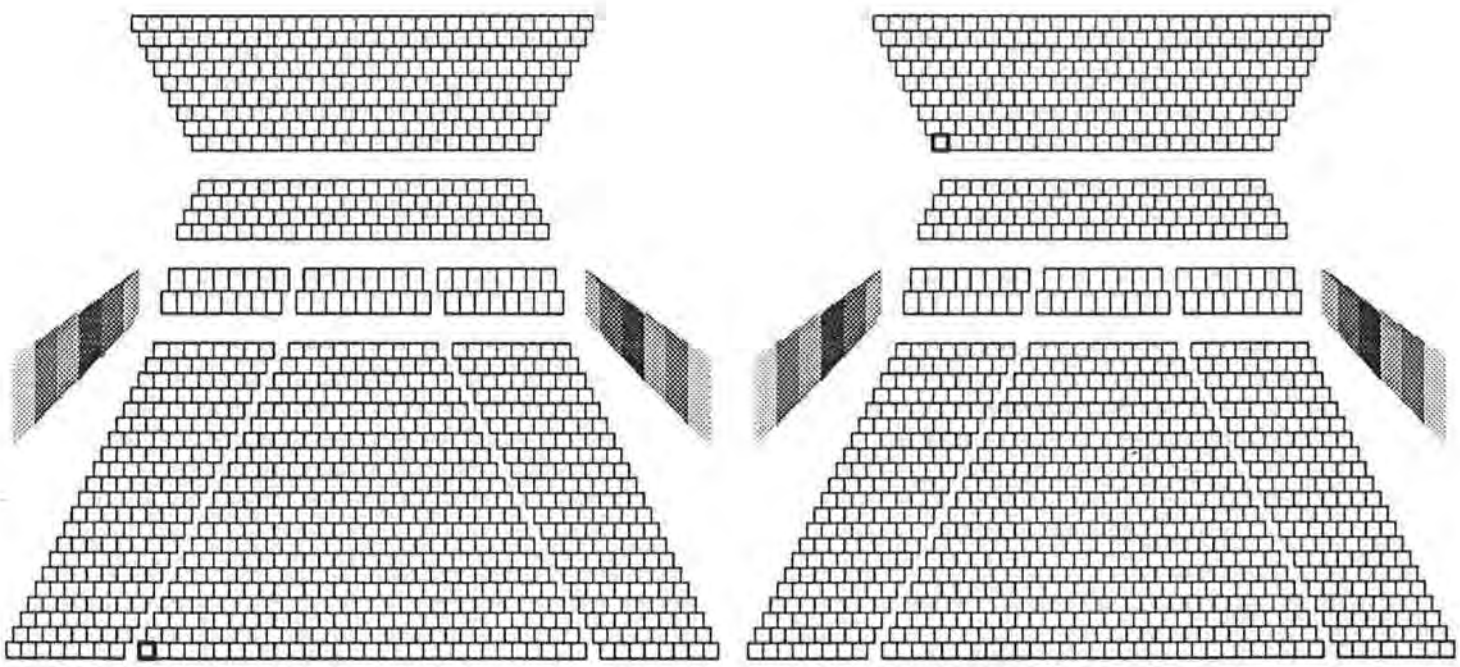

Figure 7: Theatre Layout - SELECTION2

SELECTION4 selects 19 primitives in the theatre layout. 18 primitives are selected in the theatre layout placed in the lower left corner, and one belongs to the theatre layout placed next to the one in the lower left corner along x-axes, as shown in Figure 9.

Table 4

Times (ms) for SELECTION4 for 6 different NDC pictures						
	number of primitives in NDC picture					
GKS function	1154	2308	6924	10386	16156	23080
gset_ndc_prim_attr	55.5	112.6	340.7	511.1	796.0	1137.3
gadd_nameset _ndc_picture	58.6	116.5	343.7	514.5	799.1	1141.0
gremove_nameset _ndc_picture	58.6	116.5	343.7	514.5	799.1	1141.0
greorder_ndcp, FRONT	57.2	115.2	347.1	522.1	810.5	1158.1
greorder_ndcp, BACK	70.1	129.9	367.1	546.0	841.3	1196.0
gset_ws_sel_crit	41.8	85.5	260.6	391.9	633.7	873.7
gset_view_sel_crit	57.0	116.2	352.7	529.1	825.7	1179.6

The time measurements described were repeated without NDC picture traversing. Results are presented in tables 5-8. Tables 5-8 contain two times per function in each column. The first is the function execution time without NDC picture traversing. The second is the NDC picture traversing time derived when the first time is subtracted from the corresponding time given in tables 1-4.

Table 5

Times (ms) for SELECTION1 for 6 different NDC pictures						
	number of primitives in NDC picture					
GKS function	1154	2308	6924	10386	16156	23080
gset_ndc_prim_attr	3.7	7.0	20.5	30.2	46.6	65.9
	20.9	41.1	122.7	183.8	285.8	408.6
gadd_nameset _ndc_picture	3.9	7.3	20.7	30.3	46.6	65.9
	20.8	41.1	122.7	183.6	285.7	407.7
gremove_nameset _ndc_picture	3.9	7.3	20.7	30.3	46.6	65.9
	20.8	41.1	122.7	183.6	285.7	407.7
greorder_ndcp, FRONT	8.0	15.6	44.5	66.4	102.4	145.5
	22.8	40.7	122.8	183.7	285.9	408.9
greorder_ndcp, BACK	21.2	30.1	64.1	89.5	132.5	182.5
	20.8	40.9	122.8	183.6	285.9	409.2
gset_ws_sel_crit	8.8	16.8	47.7	71.3	135.4	156.5
	4.4	9.3	30.5	45.9	69.4	104.1
gset_view_sel_crit	9.0	17.0	48.7	72.3	137.5	159.7
	19.5	39.8	121.8	182.9	259.9	407.6
gdel_prims gcreate_out_prim	6.9	13.2	37.2	55.2	85.1	120.9
	40.7	81.5	245.0	366.9	517.6	816.7

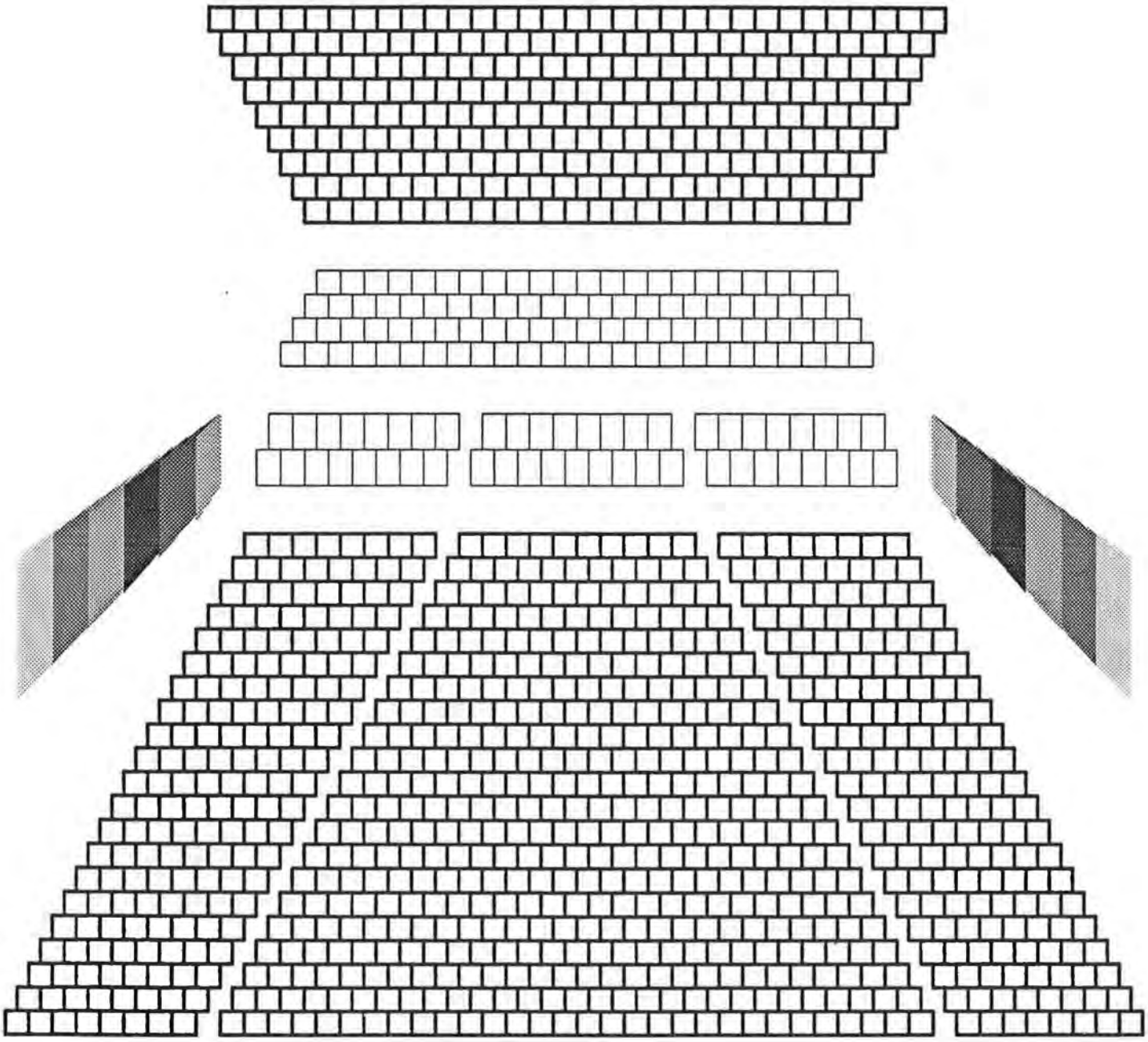

Figure 8: Theatre Layout - SELECTION3

Table 6

Times (ms) for SELECTION2 for 6 different NDC pictures						
	number of primitives in NDC picture					
GKS function	1154	2308	6924	10386	16156	23080
gset_ndc_prim_attr	13.9	30.9	97.0	151.3	228.7	327.5
	20.6	40.4	122.5	183.6	285.8	414.1
gadd_nameset						
_ndc_picture	14.4	35.0	97.8	147.2	229.1	328.4
	20.2	40.8	122.1	183.5	285.8	425.2
gremove_nameset						
_ndc_picture	14.4	35.0	97.8	147.2	229.1	328.4
	20.2	40.8	122.1	183.5	285.8	425.2
greorder_ndcp, FRONT	15.3	33.3	103.5	155.5	243.3	348.9
	20.7	40.9	122.9	183.9	286.5	409.0
greorder_ndcp, BACK	28.5	47.6	123.3	191.1	273.9	386.5
	20.6	41.1	122.7	183.8	286.3	427.3
gset_ws_sel_crit	16.2	34.7	108.0	163.7	254.5	370.0
	4.4	9.4	30.6	46.2	72.5	98.7
gset_view_sel_crit	16.5	35.2	109.2	165.2	257.1	368.4
	19.6	40.0	121.8	182.9	285.0	407.5

Table 7

Times (ms) for SELECTION3 for 6 different NDC pictures						
	number of primitives in NDC picture					
GKS function	1154	2308	6924	10386	16156	23080
gset_ndc_prim_attr	26.0	46.9	130.1	191.6	295.9	420.6
	20.0	40.4	121.9	183.4	281.1	410.4
gadd_nameset						
_ndc_picture	195.4	233.5	312.2	398.0	466.4	605.3
	21.9	41.2	127.6	191.5	287.0	415.0
gremove_nameset						
_ndc_picture	195.4	233.5	312.2	398.0	466.4	605.3
	21.9	41.2	127.6	191.5	287.0	415.0
greorder_ndcp, FRONT	36.7	59.0	147.6	211.6	319.7	449.4
	20.8	41.5	123.2	183.9	286.9	410.0
greorder_ndcp, BACK	39.2	62.8	155.0	224.3	340.8	476.9
	20.7	40.9	122.4	183.4	286.2	409.2
gset_ws_sel_crit	23.5	46.5	136.8	204.3	317.6	453.7
	17.8	22.7	43.8	59.5	85.8	117.6
gset_view_sel_crit	23.8	46.6	137.6	206.2	319.3	455.9
	20.0	40.2	121.7	182.8	308.9	407.7

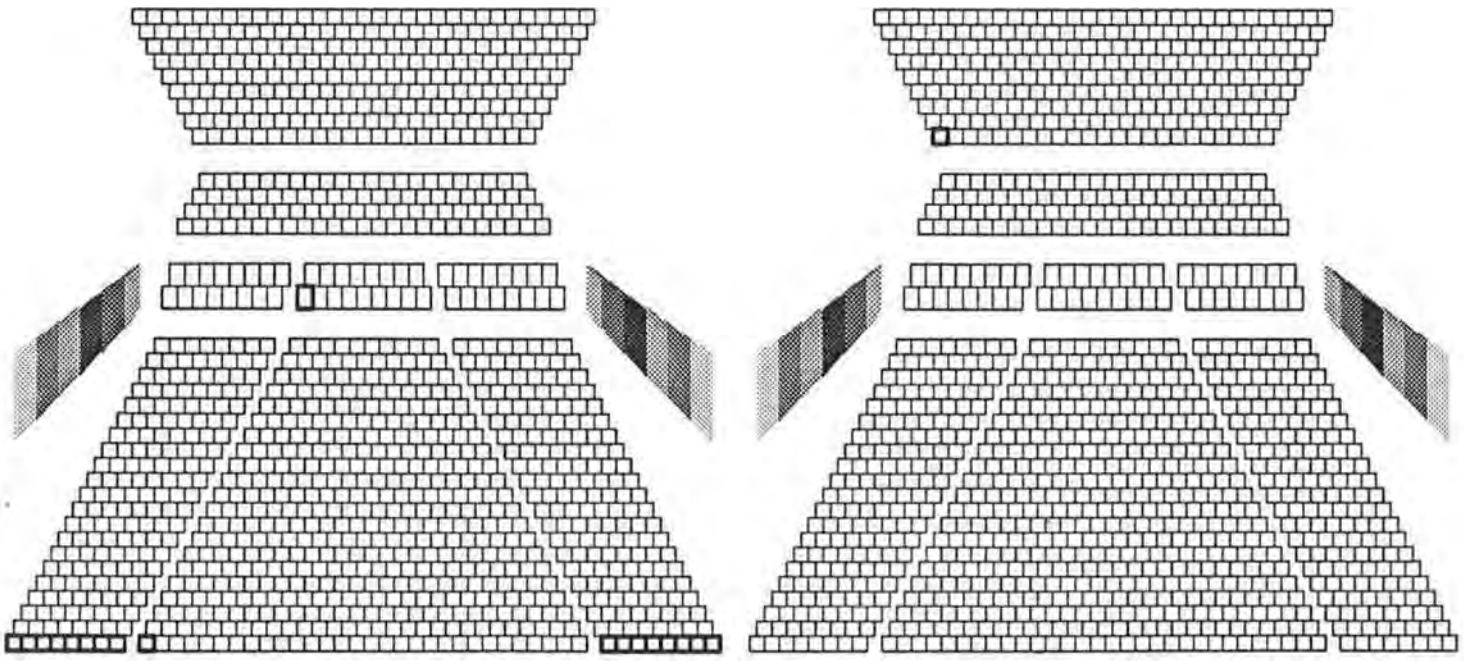

Figure 9: Theatre Layout - SELECTION4

Table 8

Times (ms) for SELECTION4 for 6 different NDC pictures						
	number of primitives in NDC picture					
GKS function	1154	2308	6924	10386	16156	23080
gset_ndc_prim_attr	35.2	71.9	217.8	327.8	510.2	730.2
	20.3	40.7	122.9	183.3	285.8	407.1
gadd_nameset _ndc_picture	38.4	75.4	221.4	331.4	514.0	740.6
	20.2	41.1	122.3	183.1	285.1	400.8
gremove_nameset _ndc_picture	38.4	75.4	221.4	331.4	514.0	740.6
	20.2	41.1	122.3	183.1	285.1	400.8
greorder_ndcp, FRONT	36.6	73.8	224.6	338.3	525.2	750.2
	20.6	41.4	122.5	183.8	285.3	407.9
greorder_ndcp, BACK	49.5	89.0	244.6	362.2	556.3	787.7
	20.6	40.9	122.5	183.8	285.0	408.3
gset_ws_sel_crit	37.8	76.0	230.4	346.1	561.0	782.2
	4.0	9.5	30.2	45.8	72.7	91.5
gset_view_sel_crit	37.8	76.2	231.4	346.7	540.9	805.0
	19.2	40.0	121.3	182.4	284.8	374.6

6 Discussion of Results

The dependency of execution times on the number of primitives for `gset_ndc_prim_attr` for each selection criterion given in tables 1-4 is graphically presented in Figure 10. The time measurements given in tables 5-8 are graphically presented in Figure 11.

Three components can be distinguished in the overall function execution time: selection time, data structures update time and traversing time.

It can be seen from tables 5-8 that traversing times are nearly the same for the corresponding NDC pictures for all functions except `gset_ws_sel_crit`. As an example traversing times for `gset_ndc_prim_attr` for SELECTION1 are presented by *t1* in Figure 12. The corresponding traversing times after optimization are presented by *to* and explained in section 7.

Table 5 showes that the traversing times for `gset_ws_sel_crit` are considerably less, because only one primitive is visible. These traversing times for SELECTION1 are presented by *t2* in Figure 12.

As the corresponding traversing times in the four cases considered in Figure 10 are nearly the same, it can be concluded that the differences in the slopes of the lines depend on the complexity of the corresponding selection criteria. The complexity of SELECTION1, SELECTION2 and SELECTION4 is significantly different with consequent different slopes of the corresponding lines. The complexity of SELECTION3 and SELECTION4 is the same. However, the corresponding line slopes are different because the selections select different numbers of primitives: SELECTION3 selects 2 primitives, while SELECTION4 selects 999

primitives. Therefore, the differences in slopes of the lines depend also on the data structures update time. The data structures update times vary with the type of the of the function as the type of the function determines how data structures have to be updated.

GKS-9x was designed to perform checking of whether a primitive satisfies a selection criterion or not, on a per nameset basis rather than on a per primitive basis. It was assumed that, on average, several primitives share a given nameset. Only one copy of a nameset is maintained in a system which is pointed to by all primitives that have it as the value of their nameset attribute. This avoids multiple computations of satisfaction of the selection criterion by the nameset as described in [4]. Furthermore, some namesets can be easily rejected based on the number of names they contain.

In the examples considered here very strong linearity (Figure 10 and 11) is the consequence of the structure of namesets associated with primitives (2.2). All primitives have different namesets in the NDC picture. Therefore, the number of namesets which potentially have to be checked against a selection criterion corresponds to the number of primitives in the NDC picture. In the case of SELECTION1, for example, when the NDC picture contains 1154 primitives 398 primitives will be rejected because their namesets contain less names than the nameset of SELECTION1 and 756 will be checked. In the case when the NDC picture contains 2308 (2*1154) primitives, 2*398 primitives will be rejected and 2*756 checked and so on. Therefore the number of namesets which have to be checked against the selection criterion grows proportionally with the number of primitives in the NDC picture which explains the strong linearity of the obtaioned results. Based on the results in tables 1-8 and this analysis, it can be concluded that the selection times grow linearly with the number of namesets which have to be compared to the selection criterion.

Results given in tables 5-8 show that traversing time does not depend on the GKS-9x function being executed, but depends only on the number of primitives stored in the NDC picture and on the number of visible primitives. The results presented in Figure 12 show that the traversing time grows linearly with the number of primitives in the NDC picture and that the slope of the line depends on the number of visible primitives. It depends on the number of primitives stored in the NDC picture because all primitives have to be traversed to select those that are visible. It depends on the number of visible primitives because they have to be traversed 16 times to determine the views of the NDC picture.

A simple selection criterion containing one comparison operation (like SELECTION1) can always be used for selecting a single primitive. It can be seen from table 5 that in such a case traversing time is considerably longer than selection time and update data structures time for some functions. Tables 5-8 show how this proportion changes for different selection criteria. Table 8 shows that even for a very complex selection criterion (consisting of 6 logical and 6 comparison operations) the traversing time is a considerable part of the overall execution time.

According to the GKS-9x DIS, the view table always contains 16 views regardless of whether an application uses them all or not. The view selection criterion for view 0 is set to SELECTALL and others to REJECTALL when a workstation is open and the corresponding workstation state list created. When this is implemented straightforwardly (as was done in the implemention considered here), visible primitives have to be traversed 16 times per single NDC picture traversing. However, as results showed that traversing time is a considerable part of the overall function execution time, it would be worth optimizing the average traversing time by considering only those views which an application uses.

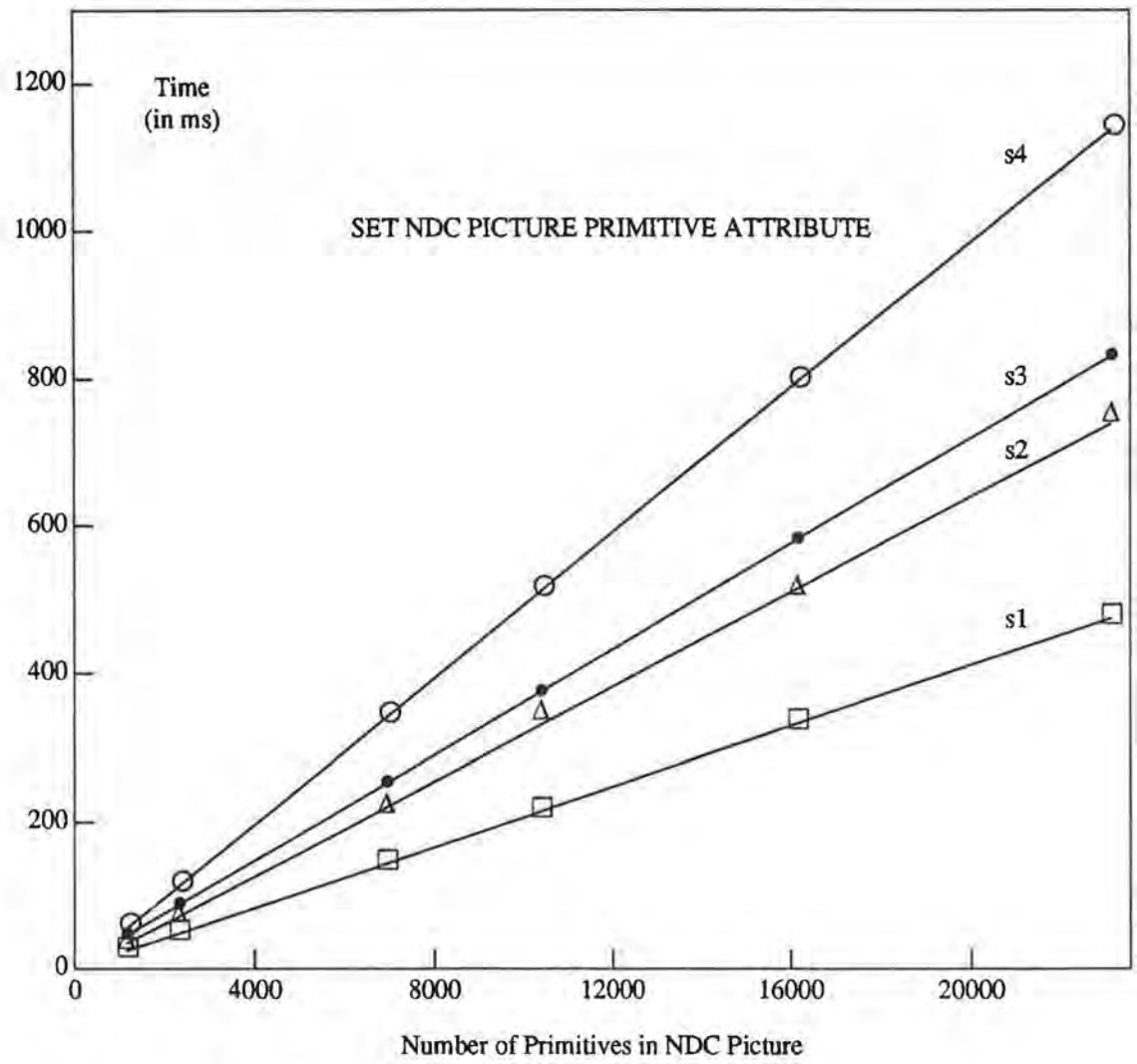
Time
(in ms)
SET NDC PICTURE PRIMITIVE ATTRIBUTE
s4
s3
s2
s1
1200
1000
800
600
400
200
0
0
4000
8000
12000
16000
20000
Number of Primitives in NDC Picture

Figure 10:

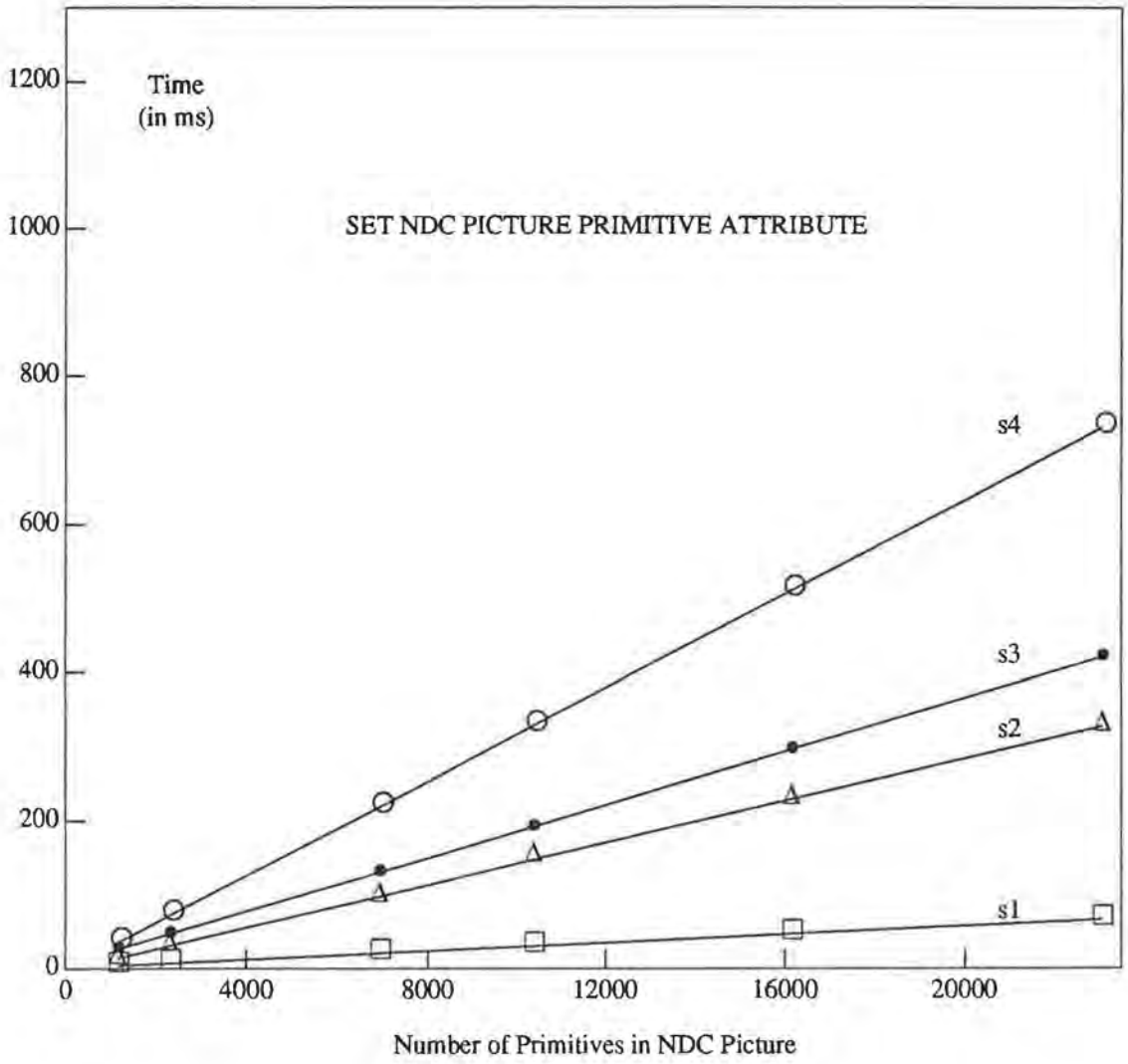
Time
(in ms)
SET NDC PICTURE PRIMITIVE ATTRIBUTE
1200
1000
800
600
400
200
0
0
4000
8000
12000
16000
20000
s4
s3
s2
s1
Number of Primitives in NDC Picture

Figure 11:

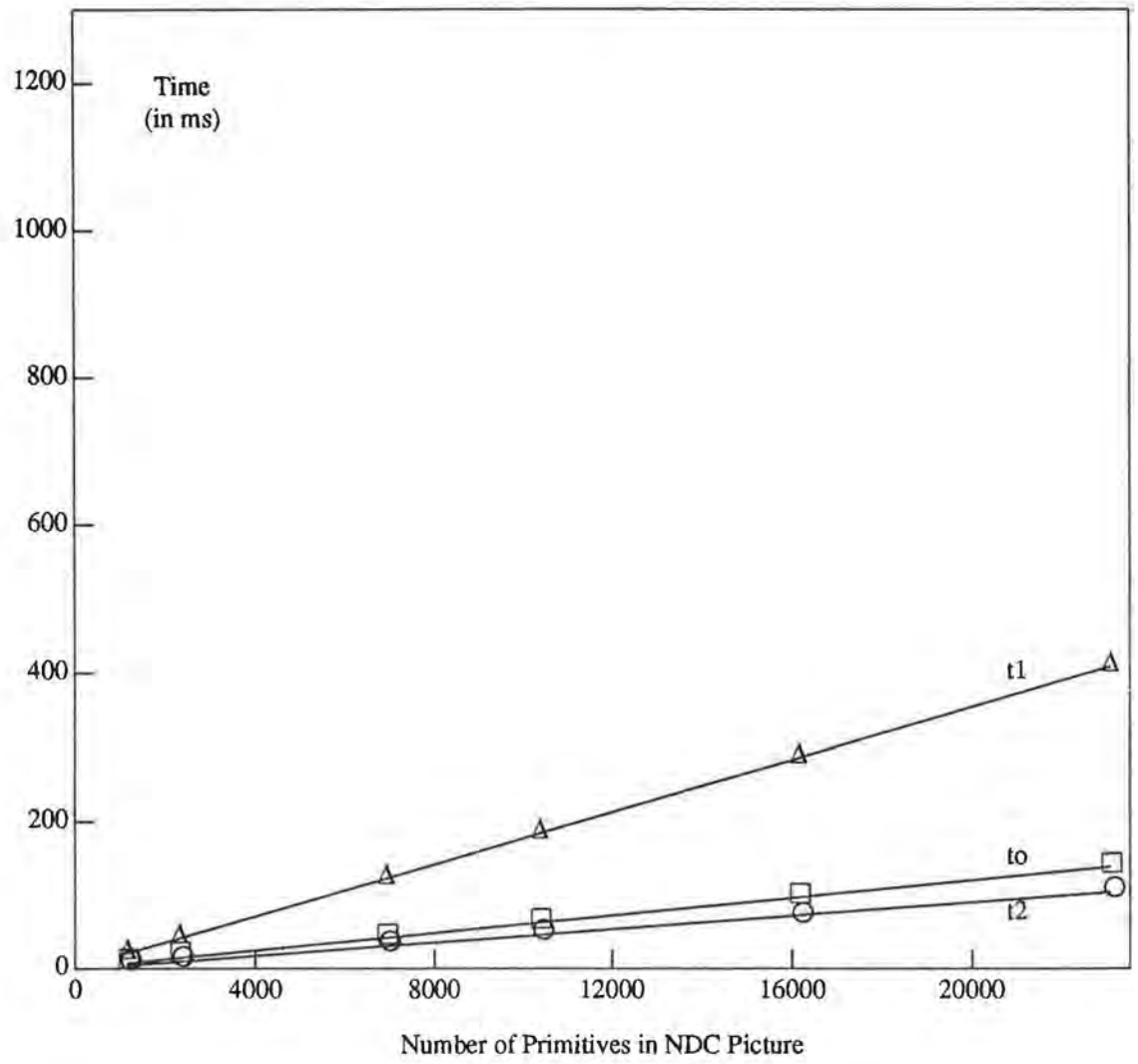
Time
(in ms)
1200
1000
800
600
400
200
0
0
4000
8000
12000
16000
20000
t1
to
t2
Number of Primitives in NDC Picture

Figure 12:

7 Optimization

One way to reduce the average traversing time is to determine dynamically for how many views the visible primitives have to be traversed for a single NDC picture traversing. To realize this, the data type **Gwssel_list** (2.4) has been extended by **numviews** member:

```
typedef struct wsel *Gwssel_list_ptr;

typedef struct wsel {
        Pint            owsid;          /* workstation identifier  */
        Gselect         *wsvis;         /* visibility selection    */
        Gselect         *wshigh;        /* highlighting selection  */
        Gselect         *wsdetc;        /* detectability selection */
        Gview           *wsview[NUMVWT]; /* view representations    */
        Pint            numviews;       /* number of used views    */
        Gpntwsf_list    *flagshead;     /* pointer to llwsf        */
        Gwssel_list_ptr nextwssel;      /*pointer to the next node */
} Gwssel_list;
```

When a workstation is open, the corresponding variable of type **Gwssel_list** is created and the member **numviews** is set to 1 to denote that only one view selection criterion is not REJECTALL. Whenever a view selection criterion is changed, its view index is compared to the corresponding **numviews**, which is accordingly set to the greater value. Therefore, **numviews** for a workstation has always the value of the greatest view index for which the selection criterion has been changed. With this optimization, visible primitives have to be traversed **numviews** rather than 16 times per a single NDC picture traversing.

Times were measured for SELECTION1 and SELECTION2 when this optimization was included in the GKS-9x implementation and the results are presented in Table 9 and Table 10. Tables 9 - 10 contain two times per function in each column. The first is the function execution time with NDC picture traversing time. The second is the NDC picture traversing time derived when the time without traversing (from Tables 5-6) is subtracted from the first time.

It can be seen that traversing time has been considerably reduced for all functions except for the **gset_ws_sel_crit**. The traversing time for **gset_ws_sel_crit** function with and without traversing is nearly the same because only one primitive is visible. Traversing times with optimization for **gset_ndc_prim_attr** for SELECTION1 is presented by *to* in Figure 12.

The optimization presented is very simple and does not give always good results. However, it was sufficient to show that the average traversing time can be substantially improved if each NDC picture traversing is performed for precomputed number of views (which can be less or equal 16), instead for the maximum number of views.

Table 9

Times (ms) for SELECTION1 for 6 different NDC pictures						
	number of primitives in NDC picture					
GKS function	1154	2308	6924	10386	16156	23080
gset_ndc_prim_attr	11.2	22.5	62.2	92.9	143.9	204.0
	7.2	17.5	41.7	62.7	97.3	138.1
gadd_nameset						
_ndc_picture	11.3	21.8	62.5	92.2	143.9	204.8
	7.4	14.5	41.8	62.6	97.3	138.9
gremove_nameset						
_ndc_picture	11.3	21.8	62.5	92.2	143.9	204.8
	7.4	14.5	41.8	62.6	97.3	138.9
greorder_ndcp, FRONT	15.1	29.7	86.3	129.3	199.7	285.2
	7.1	14.1	41.8	62.9	97.3	139.7
greorder_ndcp, BACK	28.4	44.0	105.7	153.3	230.1	322.3
	7.2	13.9	41.6	63.8	97.6	139.8
gset_ws_sel_crit	13.2	26.4	78.8	118.0	183.3	262.1
	4.4	9.6	31.1	46.7	47.9	105.6
gset_view_sel_crit	15.1	30.2	90.1	134.6	209.1	299.0
	6.1	13.2	41.4	62.3	71.6	139.3
gdel_prims						
gcreate_out_prim	20.7	40.9	120.8	195.3	279.5	398.8
	13.8	27.7	83.6	140.1	194.4	277.9

Table 10

Times (ms) for SELECTION2 for 6 different NDC pictures						
	number of primitives in NDC picture					
GKS function	1154	2308	6924	10386	16156	23080
gset_ndc_prim_attr	21.0	44.5	138.7	224.8	326.6	467.4
	7.1	13.6	41.7	63.5	97.9	139.9
gadd_nameset						
_ndc_picture	21.5	48.6	139.3	209.4	326.6	467.4
	7.1	13.6	41.5	62.2	97.5	139
gremove_nameset						
_ndc_picture	21.5	48.6	139.3	209.4	326.6	467.4
	7.1	13.6	41.5	62.2	97.5	139
greorder_ndcp, FRONT	22.5	47.1	145.1	218.6	342.0	520.2
	7.2	13.8	41.6	63.1	98.7	171.3
greorder_ndcp, BACK	35.6	61.9	164.9	242.7	372.7	526.3
	7.1	14.3	41.6	51.6	98.8	139.8
gset_ws_sel_crit	20.5	44.5	138.7	210.2	327.7	469.7
	4.3	9.8	30.7	46.5	73.2	99.7
gset_view_sel_crit	22.5	48.1	150.5	227.0	354.0	507.4
	6.0	12.9	41.3	61.8	96.9	139

8 General Discussion

Studies have been carried out at Rutherford Appleton Laboratory on the specification and implementation of the major new functionality in GKS-9x, as well as on the effects of an application complexity on the implementation performance. This was presented in [3, 4, 7] and here. Reservations had been expressed in the GKS Rapporteur Group about whether it was possible to implement the proposed GKS-9x functionality efficiently and whether the functionality would achieve the desired aims. The motivation for the work presented in the papers was to investigate these issues.

Prior to exploring approaches to implementing some of the new functionality, the authors prepared a formal description [7] of the key parts of the framework of GKS-9x which were relevant to the implementation issues which were later studied.

The experience gained in the implementation of a part of the GKS-9x and in exercising a prototype has been presented in [3, 4] and here. The implementation includes some of the major new concepts introduced in GKS-9x: NDC picture, namesets, selection criteria and multiple views, as the main aim of the work was to explore how the GKS-9x NDC picture, namesets and selection criteria could be implemented efficiently and to investigate the visual effects of multiple views.

A central aspect of the GKS-9x implementation was the provision of compact data structures which are dynamically created and destroyed to suit the current application requirements, and efficient execution time of the GKS-9x functions considered.

It has been shown that a comprehensive data type, selection criterion, can be implemented using standard C language type constructors.

Two frequently performed processes: selection of primitives and traversal were identified. An approach in the design and implementation of the GKS-9x data structures was presented which provides high efficiency for these processes.

In GKS-9x the NDC picture is defined as a sequence of output primitives. The order of appereance of output primitives in the NDC picture is important in that if two output primitives overlap, the second primitive in the sequence may obscure some part of the first.

The NDC picture is represented as a doubly linked list of primitives in the GKS-9x implementation considered here. It had to be a doubly linked list to provide the implementation of REORDER NDC PICTURE function, as described in [4].

GKS-9x provides a set of functions for editing the NDC picture. Selected primitives can be reordered or deleted or can have their attributes changed using this function. Each workstation is responsible for selecting the subset of the NDC picture to be viewed and rendering it. As the selection of primitives is a part of many GKS-9x functions, it is important to implement it efficiently.

A primitive is selected for some purpose if its nameset attribute satisfies the corresponding selection criterion. It was assumed that, on average, several primitives share a given nameset. Under this assumption, the GKS-9x implementation was designed to perform checking of whether a primitive satisfies a selection criterion or not, on a per nameset basis rather than on a per primitive basis. Only one copy of a nameset attribute is maintained in a system which is pointed to by all primitives that have it as the value of their nameset attribute. A list of the different namesets in use is maintained, together with lists of pointers to the primitives that have that nameset as the value of their nameset attribute. This avoids multiple computations of satisfaction of the selection criterion by the nameset as described in [3].

When a GKS-9x function for editing the NDC picture is performed it accesses a list of namesets to check which of them satisfy the selection criterion (which is the function's parameter) in order to edit the NDC picture. To provide a rapid way of accessing to the namesets which potentially satisfy the selection criterion, namesets containing the same number of names are chained together in linked lists of namesets. A linked list of keys contains pointers to linked lists of namesets. A nameset's key is the number of names it contains. Comparing a key to the number of names in a nameset of the selection criterion, the whole group of namesets in the corresponding list can be eliminated avoiding their complete checking as described in [4].

To further reduce the selection time selection criterion optimization and nameset representation optimization were introduced.

To explore the effects of optimizations on execution times, an application, *theatre*, was considered which comprises 1154 polylines, with different namesets. The execution times for selecting subsequences of output primitives were measured for groups of GKS-9x functions, for three versions of the implementation: without optimization, with selection optimization and with both selection optimization and nameset representation optimization. Times were measured for 13 selection criteria and the results were presented in several tables. It was shown that the avarage function execution time has been reduced substantially with optimizations. The two proposed ways of optimization are complimentary and can be generally applied to implementations of GKS-9x. The selection structure optimization reduces the number of cases when two namesets have to be compared, and therefore reduces the overall execution time, while the nameset representation optimization reduces the execution time of nameset comparison.

The NDC picture traversal is another frequently performed process, as it is assumed that the NDC picture has to be always displyed up-to-date.

Each workstation has associated a VISIBILITY selection criterion which is used for selecting the subset of the NDC picture to be viewed. The subset is further checked, by a set consisting of 16 view selection criteria, to determine which primitives appear in which views of the NDC picture. It was asssumed that the NDC picture will be traversed for display more frequently than the selection criteria are changed. Under this assumption, a flag was allocated for each nameset/selection criterion to indicate whether the nameset is accepted or rejected by that selection criterion. This allows traversal process to check only the corresponding flag to determine whether a primitive is visible or not, and a set of flags to decide how many views of each primitive selected are displayed on the workstation. The allocation of flags provided the minimum number of recomputations of whether a nameset satisfies a selection criterion or not. Only the modification of a selection criterion or a nameset leads to the recomputations for the nameset/selection criterion affected. Furthermore, the allocation of flags provided complete separation of two most frequently performed processes in GKS-9x which allows parallel execution of these two proceses.

The main aim of the further studies on GKS-9x implementation reported in this paper was to explore how the number of primitives handled in an application affects the GKS-9x performance. Accordingly, an application was developed which could be instantiated to manipulate different number of primitives. Moreover, the application had two versions: interactive and non-interactive. The non-interactive version was used to carry out performance measurements.

The execution times for a group of GKS-9x functions were measured for four selection criteria. Times were measured for six different NDC pictures (containing different number of primitives in the range of 1154 - 23080 primitives) for each selection criterion. Then

the time measurements were repeated without NDC picture traversing. The results showed that traversing times are nearly the same for corresponding NDC pictures for all functions, except SET WORKSTATION SELECTION CRITERION, because the number of visible primitives was the same. In the case of SET WORKSTATION SELECTION CRITERION the number of visible primitives was reduced with the consequent substantial reduction of traversing time. The results showed that the traversing time is a considerable part of the overall function execution time and that heavily depends on how many views of the NDC picture are displayed. According to the GKS-9x DIS, the view table always contains 16 views regardless of whether an application uses them all or not. The optimization of the traversing time was introduced. It was assumed that, on average, not all 16 views of the NDC picture are displayed. Therefore, the subset of visible primitives is checked by dynamically determined number of view selection criteria, instead by a set consisting of 16 view selection criteria, when optimisation was included. The execution times were measured again with optimisation included and showed considerable improvment.

The GKS-9x functions execution times were analyzed here to investigate their dependencies on various parameters. Three components of an execution time can be distinguished: selection time, data structures update time and traversing time.

It was concluded that GKS-9x function selection time, for a given selection criterion, depends on the complexity of the selection criterion and the number of namesets that have to be checked and does not depend on the type of function. The results showed that the selection time grows linearly with the number of namesets in system which have to be checked against the selection criteria and that the slope of the line depends on the complexity of the selection criterion.

However, the data structures update times vary with the type of the function as the type of a function determines how data structures have to be updated. For example, for the SET NDC PRIMITIVE ATTRIBUTE function, the results showed that the data structures update time grows linearly with the number of primitives for which data structures have to be updated. This does not hold for ADD NAMESET TO NDC PICTURE and REMOVE NAMESET FROM NDC PICTURE, which depend on the number of namesets which have to be revised.

The results showed that the traversing time grows linearly with the number of primitives in the NDC picture and that the slope of the line depends on the number of visible primitives. It depends on the number of primitives stored in the NDC picture because all primitives have to be traversed to select those that are visible. It depends on the number of visible primitives because they have to be traversed several times to determine the views of the NDC picture.

The results obtained validate the GKS-9x data structure design and data types implementation presented. Several benefits were demonstrated such as: complete separation of two frequently performed processes, minimization of the number of recomputations of satisfaction of the selection criterion by the nameset and the reduction of the execution time of nameset comparison.

The interactive version of the application, described here, allows the user to exercise GKS-9x functions on-line with user defined selection criteria, namesets and views in order to get better insight into the new functionality and system behaviour. It displays the theatre layout on two GKS-9x workstations and allows sessions in which GKS-9x functions, selected interactively, are performed. A user can edit the NDC picture, change its visibility and views on both workstations, in order to explore the effects of various GKS-9x functions. A function is selected from a menu. The number of primitives manipulated in an application (and therefore stored in the NDC picture) can be easily changed allowing simulation of very

comprehensive applications in order to explore the behaviour of the system when heavily loaded. The largest number of primitives tried was 23080.

9 Conclusions

In the first part of this paper an interactive application *thm* which allows exercising of the experimental implementation of GKS-9x with arbitrary user supplied selection criteria and views, was presented. The number of primitives manipulated in an application (and therefore stored in the NDC picture) can be easily changed allowing simulation of very comprehensive applications in order to explore the behaviour of the system when heavily loaded. The largest number of primitives tried was 23080.

In the second part of the paper an analoguous non-interactive application *thm_time*, suitable for measuring function execution times, was presented.

To explore the effects of the number of primitives on execution times, 6 instances of the *thm_time* application with different number of primitives manipulated were considered for a selection criterion. This was repeated for 4 different selection criteria.

Three components of an function execution time were identified and their dependences on the number of primitives, number of namesets and the complexity of the selection criteria were analyzed.

It has been shown that the performance can be improved if the traversing optimization is included.

Finally, measurements of GKS-9x implementation performance reported, validate assumptions made when GKS-9x was designed.

References

[1] Information processing systems - Computer graphics - Graphical Kernel System (GKS) functional description ISO 7942 ISO Central Secretariat (August 1985).

[2] K.W. Brodlie, D.A. Duce and F.R.A. Hopgood, The New Graphical Kernel System, Computer-Aided Design 23(4), pp. 312-318 (1991).

[3] L.B.Damnjanović, D.A.Duce and S.K.Robinson: GKS-9x: Some Implementation Considerations, Computer Graphics Forum, 12(3), Conference Issue, Barcelona, Spain Septembre 6-9, 1993, Ed. R.J. Hubbold and R. Juan. pp. 295-313.

[4] L.B.Damnjanović, D.A.Duce and S.K.Robinson: GKS-9x: Implementation of Selection, ERCIM Research Reports, ERCIM-93-R017 RAL.

[5] Information processing systems - Computer graphics - Programmer's Hierarchical Interactive Graphics System functional description ISO/IEC 9592: 1 (1989).

[6] J. Loubersac: VDM through Pictures, Bull Report RAD/DMA/92001, January 1992.

[7] L.B.Damnjanović, D.A.Duce: GKS-9x: A Specification of the Framework, RAL Report RAL-93-061.

Appendix 1

For illustration the *thm_time* main module is given for SELECTION1.

```
/*  cs11_time.c
*/

/* Definitions of include files and some global variables are omitted */

Gint_list swanlake_list;
Gint_list cs11_list;
Gselect  rall, selall;
Gselect cs11_sel;
Ppoint point11[] =
{
   { 0.20, 0.05}, { 0.22, 0.05}, { 0.22, 0.07}, { 0.20, 0.07}, { 0.20, 0.05}
};
static Gpoint_list p11_list;

main()
{
Gpattr_value at;
static Pint i,j, n,num,iter, iter1;
static Pchar  *k;

/* create hash table */
     hash_tab(CREATE_H, k, &n);

/* number of primitives: NUM_X and NUM_Y are the number of theatre patterns
   along X and Y axes respectively
*/
     num = 1154*NUM_X*NUM_Y;

/* number of iterations */
     iter = 1000;
     iter1 = iter/2;

/* nameset initialization */
     nsini();

/* initalization of selection criteria */
     selnsini();

/* open GKS */
     gopen_gks();
```

```
/* open WS1  */
     gopen_ws(WS1);

/* draw seating plan */

     gset_ws_sel_crit(WS1, VISIBILITY, &rall);
     seatplan();
     gset_ws_sel_crit(WS1, VISIBILITY, &selall);

/******** MEASURING TIME for SELECTION1 **************/

/*  CONTAINS{ORCHESTRA,STALLS,MIDDLE,LEFT,I,1,X1,Y1} */

/* SET NDC PICTURE PRIMITIVE ATTRIBUTE - type */
     at.type_f = THIN;
     start_rtimer();
     for(i=0; i<iter; ++i) {
          gset_ndc_prim_attr(&cs11_sel, LINEWIDTH, &at);
     }
     stop_rtimer();

/* SET NDC PICTURE PRIMITIVE ATTRIBUTE - colour */
     at.type_i = GREEN;
     start_rtimer();
     for(i=0; i<iter; ++i) {
          gset_ndc_prim_attr(&cs11_sel, PCOLINDEX, &at);
     }
     stop_rtimer();

/* ADD/REMOVE SET OF NAMES TO/FROM NDC PICTURE */
     start_rtimer();
     for(i=0; i<iter1; ++i) {
          gadd_nameset_ndc_scene(&swanlake_list, &cs11_sel);
          gremove_nameset_ndc_scene(&swanlake_list, &cs11_sel);
     }
     stop_rtimer();

/* SET WORKSTATION SELECTION CRITERION */
     start_rtimer();
     for(i=0; i<iter; ++i) {
          gset_ws_sel_crit(WS1, VISIBILITY, &cs11_sel);
     }
     stop_rtimer();
     gset_ws_sel_crit(WS1, VISIBILITY, &selall);

/* SET VIEW SELECTION CRITERION */
     start_rtimer();
     for(i=0; i<iter; ++i) {
```

```
        gset_view_sel_crit(WS1, 0, &cs11_sel);
    }
    stop_rtimer();
    gset_view_sel_crit(WS1, 0, &selall);

/* REORDER NDC PICTURE - front */
    start_rtimer();
    for(i=0; i<iter; ++i) {
        greorder_ndcp(&cs11_sel,&boxsel,FRONT);
    }
    stop_rtimer();

/* REORDER NDC PICTURE - back */
    start_rtimer();
    for(i=0; i<iter; ++i) {
        greorder_ndcp(&cs11_sel,&boxsel,BACK);
    }
    stop_rtimer();

/* DELETE PRIMITIVES */
    p11_list.num_points = 5;
    p11_list.points = point11;
    ap11_list.num_points = 5;
    ap11_list.points = apoint11;
    gadd_nameset(&cs11_list);
    start_rtimer();
    for(i=0; i<iter; ++i) {
        gdel_prims(&cs11_sel);
        gcreate_out_prim(SETOFPOLYLINES, &p11_list);
    }
    stop_rtimer();

/******** END OF MEASURING TIME **************/
}
```